LBL-24928 c.2

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

EARTH SCIENCES DIVISION

A Code to Compute Borehole Fluid Conductivity Profiles with Multiple Feed Points

F.V. Hale and C.F. Tsang

March 1988

LBL-24928 c.2

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

LBL-24928
NDC-2

A Code to Compute Borehole Fluid Conductivity Profiles With Multiple Feed Points

F.V. Hale and C.F. Tsang

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

March 1988

CONTENTS

ABSTRACT

It is of much current interest to determine the flow characteristics of fractures intersecting a wellbore in order to understand the hydrologic behavior of fractured rocks. Often inflow from these fractures into the wellbore is at very low rates. A new procedure (Tsang, 1987) has been proposed and a corresponding method of analysis developed to obtain fracture inflow parameters from a time sequence of electric conductivity logs of the borehole fluid.

The present report is a companion paper to Tsang (1987) giving the details of equations and computer code used to compute borehole fluid conductivity distributions. Verification of the code used and a listing of the code are also given.

1. INTRODUCTION

It is of much current interest to determine the flow characteristics of fractures intersecting a wellbore in order to provide data in the estimation of the hydrologic behavior of fractured rocks. Often inflow from these fractures into the wellbore is at very low rates. A new procedure (Tsang, 1987) has been proposed and a corresponding method of analysis developed to obtain fracture inflow parameters from a time sequence of electric conductivity logs of the borehole fluid. This is briefly described as follows.

Consider an uncased section of a wellbore that intersects a number of flowing fractures. In general, the flowing fractures contain fluids with different salinities and chemical compositions and hence different electric conductivities. Initially, the wellbore fluid is replaced by passing a tube to the well bottom and slowly injecting de-ionized water. This leaves the wellbore with a very low, residual conductivity. Next the well is produced at a flow rate Q, which is kept small to avoid turbulence in the wellbore, and electric conductivity logs of the borehole fluid are made at a series of times. Peaks in electric conductivity develop as fluid from each fracture inflow point enters the wellbore. At each fracture inflow point, the parameters characterizing the flow are: t_{oi}^F, the time when the fracture fluid emerges at the wellbore; x_i, the location of the inflow point; q_i^F the volumetric inflow rate; and $q_i^F C_i^F$, the solute mass inflow rate, where C_i^F is the concentration of chemicals in the fracture fluid. An inverse method is used to determine the above parameters by matching calculated electric conductivity distributions to the observed ones.

The present report is a companion paper to Tsang (1987) giving the details of equations and computer code used to compute borehole fluid conductivity distributions. Verification of the code used and a listing of the code are also given.

In Section 2 the analytical and numerical solutions to the governing equations are described. Section 3 presents a detailed discussion of the FORTRAN code. Section 4 describes the input and output files, and Section 5 contains three validation cases.

A IBM PC compatible diskette is available upon request from the authors. It contains the FORTRAN code and sample input files.

2. ANALYTICAL AND NUMERICAL SOLUTIONS

The principal equation governing borehole fluid electric conductivity variations is the equation for the transport of mass (or electrolyte concentration) in the borehole. However, additional consideration must be given to the temperature dependence of conductivity and the determination of conductivity as a function of electrolyte concentration.

2.1 Temperature Dependence of Conductivity

All calculations in the present work are made assuming a uniform temperature of 20 ° C throughout the borehole. Actual borehole temperatures generally vary with depth, so temperature corrections must be applied to field conductivity data to permit direct comparison with model output.

The effect of temperature on conductivity can be estimated using the following equation from NAGRA (1987):

$$\sigma(20^\circ C) = \frac{\sigma(T_x)}{1 + S(T_x - 20^\circ C)} \tag{1}$$

The value of S is estimated at 0.022.

An independent estimate of the effect of temperature on conductivity is given by Schlumberger (1984), which gives a formula (Chart Gen-9) that is equivalent to the equation above with S=0.024.

Generally temperature increases with depth below the land surface. If full temperature logs are available, these data can be used to correct the corresponding conductivities. However, if no complete logs are available, a simplifying assumption may be made that the temperature variation in the borehole is linear and can be modeled by:

$$T_x = Ax + B \tag{2}$$

where A and B are parameters determined by fitting any available temperature versus depth data. If the fit is unsatisfactory, other relationships with higher order terms must be used.

2.2 Conductivity as a Function of Concentration

The determination of electric conductivity as a function of electrolyte concentration is well known. A relatively simple equation for conductance given by Onsager (1926, 1927) includes terms for the dielectric constant, absolute temperature, viscosity and concentration, and the conductance at an extrapolated "infinite" dilution. For our model, however, a simple quadratic fit to measured values will be used.

Tables of equivalent conductances for small concentrations of electrolytes can be found in most electrochemistry reference books. Values of conductivity at the specified concentrations can be computed from these tables and fit with a quadratic approximation. In general, we expect that small concentrations will be found in the borehole fluid, and the curve will be nearly linear. It is important to keep in mind that these values are for small concentrations, and that the curve departs greatly from linear at high electrolyte concentrations.

Fracture fluids typically contain a variety of ions, the most common being Na^+, Ca^{2+}, Mg^{2+}, Cl^-, SO_4^{2-}, and HCO_3^-. If a hydrochemical analysis has been completed, various methods are available for computing an equivalent NaCl concentration for other ions. Schlumberger (1984) presents charts of multiplicative factors that convert various solutes to equivalent NaCl concentrations with respect to their effect on electric conductivity.

Once such a conversion is done, the following values taken from Shedlovsky et al. (1971) may then be used to relate total equivalent NaCl concentrations to equivalent conductances at 25 ° C:

Concentration, C		Equivalent Conductance, κ	Conductivity, σ
equiv/liter	kg/m^3	$S\ cm^2/mole$	$\mu S/cm$
0.	0.	126.45	0
0.0005	0.0292	124.50	62
0.001	0.0584	123.74	124
0.005	0.292	120.65	603
0.01	0.584	118.51	1,190
0.02	1.17	115.76	2,320
0.05	2.92	111.06	5,550
0.10	5.84	106.74	10,700

The conductivity σ can be calculated by multiplying the equivalent concentration C in equivalents per liter times the equivalent conductance κ with appropriate conversion factors:

$$\sigma \left[\frac{\mu\text{S}}{\text{cm}}\right] = \left(\text{C}\left[\frac{\text{equiv}}{\text{liter}}\right]\right)\left(1\left[\frac{\text{mole NaCl}}{\text{equiv NaCl}}\right]\right)\left(\kappa\left[\frac{\text{S cm}^2}{\text{mole}}\right]\right)\left(10^3\left[\frac{\text{liter } \mu\text{S}}{\text{cm}^3\text{ S}}\right]\right)$$

or, more concisely,

$$\sigma = 1{,}000 \text{ C } \kappa \tag{3}$$

where C is the equivalent concentration in equiv/l, κ is the equivalent conductance in S cm^2/mole, and σ is the conductivity in μS/cm. These values are tabulated above.

The data can be fit fairly well using a quadratic approximation:

$$\sigma = 2{,}075 \text{ C} - 45 \text{ C}^2 \tag{4}$$

where C is the concentration in kg/m^3 and σ is the conductivity in μS/cm at 25/(deC. The expression is accurate for a range of C up to 5 kg/m^3 and σ up to 10,000 μS/cm. For even lower C up to 1 kg/m^3 and σ up to 0.2 μS/cm, the second term may be neglected also:

$$\sigma = 2{,}075 \text{ C} \tag{5}$$

Graphs of the conductivity based on equation (3) and equation (4) are shown in Figure 2-1.

Although the experimental values are for 25 ° C, they may be used at 20 ° C if multiplied by 0.89 (based on equation (1) for the temperature dependence of conductivity). Thus the above relationship would be

$$\sigma = 1{,}850 \text{ C} - 40 \text{ C}^2 \ . \tag{6}$$

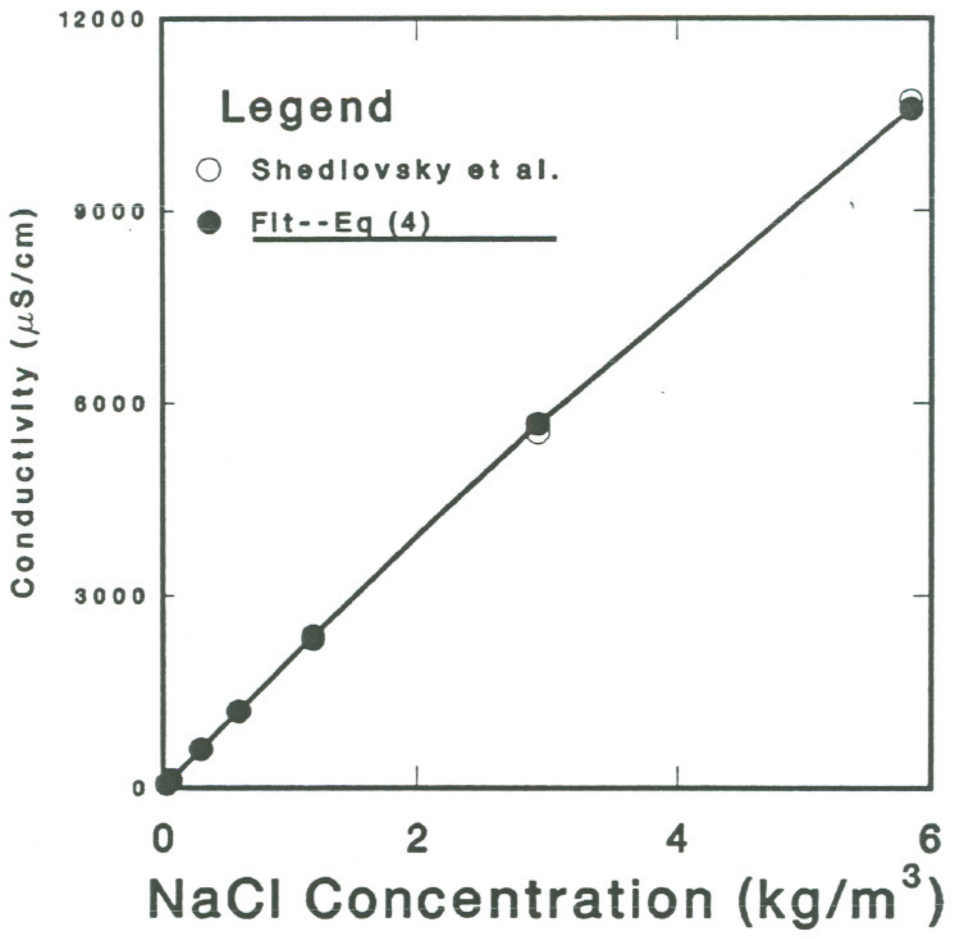

Figure 2-1. Conductivity vs. concentration.

2.3 Governing Equation for Borehole Flow with Sources

The differential equation for mass or solute transport in a borehole is:

$$\frac{\partial}{\partial x}\left(K_i \frac{\partial C}{\partial x}\right) - \frac{\partial}{\partial x}\left(C_i V_i\right) + S_i = \frac{\partial C_i}{\partial t} \tag{7}$$

where

C is the concentration (kg/m^3)
K is the dispersion coefficient (m^2/sec)
S is the source term ($kg/m^3 \cdot sec$), and
V is the fluid velocity (m/sec)

This parabolic partial differential equation is solved numerically using the finite difference method (FDM). The following initial and boundary conditions are also specified:

$$C(x, 0) = C_o(x) \tag{8}$$

$$C(x > x_{max}, t) = 0 \tag{9}$$

$$K = 0 \text{ for } x > x_{max} \text{ or } x < x_{min} \tag{10}$$

The first condition allows for the specification of initial electrolyte concentrations in the borehole. The second condition implies that there is no electrolyte in the borehole fluid flowing from below the area of interest. If there is a background concentration in the fluid flowing from the borehole bottom, this value should be added to all of the resulting concentrations. The third condition indicates that dispersion does not take place across the specified boundaries of the area of interest. In general, advection will be the dominant process at the boundaries. If dispersion is dominant for a particular problem, the boundaries should be extended in order to prevent improper trapping of electrolyte.

Discretization in Space

Uniform, one-dimensional spacing of nodes is used. It is assumed that the borehole has uniform diameter d, and that the region of interest is divided into N equal length cells of length Δx. Position values indicate depth in the borehole, and thus x is zero at the surface and increases downward. The flow within the borehole is generally upward, and the cell index i increases downstream (upward, toward the surface). Thus cells 1 and N are located at the bottom and top of the region, respectively, and node x_i is upstream of and at a greater depth than node x_{i+1}. In general, node i is located at:

$$x_i = x_{max} - (i-1/2)\Delta x \tag{11}$$

with boundaries of $x_{max} - (i-1)\Delta x$ upstream (at a depth below the node), and $x_{max} - (i)\Delta x$ downstream (at a depth above the node).

If the length of the region of interest is not a multiple of Δx, then the upper limit is adjusted accordingly and a message is produced.

Note that because all cells are assumed to have the same geometry, flow rates are directly proportional to linear velocities.

Discretization in Time

The problem duration is divided into M equal periods from time t_o to time t_M as specified by the input value Δt. If the duration is not a multiple of Δt, the ending time is adjusted accordingly. Each time step k has duration Δt and covers the period from $t_{k-1}=t_o+(k-1)\Delta t$ to $t_k=t_o+(k)\Delta t$.

The input time step Δt will be adjusted if it is greater than one half of the smallest time required for a cell to empty. Thus, the following condition must be met:

$$\Delta t \leq \frac{A\Delta x}{2Q} \tag{12}$$

where $A\Delta x$ is the uniform cell volume (A being the uniform cross-sectional area) and Q is the total flow rate out of the region of interest.

Methods of Computing the Dispersion Coefficient

Within the code, three methods are available for determining the dispersion coefficient for use at the interface between each pair of cells, $K_{i\pm1/2}$: constant, velocity scaled, and velocity squared scaled. The first approach is used to model dispersion due to molecular diffusion; the second, velocity dependence, is an approximation for porous medium transport; and the last, velocity-squared dependence, corresponds to Taylor dispersion for flow in a pipe. With each method, the dispersion coefficients at the two adjacent cells to an interface are computed, then the harmonic mean is used at the interface. Because no dispersion occurs across the region boundaries, $K_{1/2}$ and $K_{N+1/2}$ are defined to be zero.

With the constant method, the input dispersion parameter, K_o, is used for all the cell interfaces and equation (7) simplifies to

$$K\frac{\partial^2 C}{\partial x^2} - \frac{\partial}{\partial x}\left(C_i V_i\right) + \frac{C_i^f q_i^f}{A\Delta x} = \frac{\partial C_i}{\partial t} \tag{13}$$

where C_i^f is the concentration, if any, of fracture fluid flowing into cell i, q_i^f is the fracture fluid inflow rate, if any, into cell i, and $A\Delta x$ is the uniform cell volume (A being the cross-sectional area).

The other two methods involve a spatial variation of K, so the governing equation may be rewritten as:

$$\frac{\partial}{\partial x}\left(K_i \frac{\partial C}{\partial x}\right) - \frac{\partial}{\partial x}\left(C_i V_i\right) + \frac{C_i^f q_i^f}{A\Delta x} = \frac{\partial C_i}{\partial t} \tag{14}$$

The velocity scaled methods use a reference dispersion coefficient K_o defined as the dispersion coefficient at a depth where the flow velocity is equal to the mean velocity or the mean velocity squared,

$$\overline{V^n} = \frac{\min(V_i^n) + \max(V_i^n)}{2}, \tag{15}$$

where V_i^n is the fluid flow velocity at node i raised to the first or second power (n=1 or 2). Then the dispersion coefficient for node i is given by

$$K_i = K_o\left(\frac{V_i^n}{\overline{V^n}}\right). \tag{16}$$

Note that since the cells have a uniform volume the velocities are proportional to the flow rates, and the actual calculations are based on q_i rather than v_i ($q_i = v_i A$, where A is the uniform cross-sectional area).

The dispersion coefficient at the interface between two cells is the harmonic mean:

$$K_{i\pm 1/2} = 2\left(\frac{1}{K_i} + \frac{1}{K_i \pm 1}\right)^{-1} \tag{17}$$

For all cells with no flow (e.g. upstream from the first feed point), the dispersion coefficient of the first cell with nonzero flow is used.

The dispersion coefficients are then adjusted for the problem geometry,

$$\gamma_{i+1/2} = \frac{A\,K_{i+1/2}}{\Delta x} \tag{18}$$

where A is the uniform cross-sectional area.

Calculation of Flow Rates

Flow rates are assumed to be constant with time, i.e. a steady state flow is assumed. The flow rates into the bottom and out of the top of the region, as well as the flow rate for each of the feed points is included in the problem description.

The flow rate into cell i from the feed points is calculated as,

$$q_i^f = \sum_j \chi(j,i)\, q_j^F\ , \tag{19}$$

where

$$\chi(j,i) = \begin{cases} 1 \text{ if feedpoint j flows into cell i} \\ 0 \text{ otherwise} \end{cases}$$

and q_j^F is the flow rate of feed point j.

The total flow into each cell, q_i, is then computed as the sum of the flows into the cell:

$$q_i = q_i^f + q_{i-1/2}\ . \tag{20}$$

where q_i^f is the flow into cell i from the feed points, and $q_{i-1/2}$ is the flow into cell i from the adjacent upstream cell, i-1. Note that $q_{1/2}$ is the flow into the bottom of the region.

The downstream flow to the next cell, $q_{i+1/2}$, is equal to q_i, and it is assumed that there is no leakage from the borehole. The flow from the downstream boundary cell, $q_{N+1/2}$, flows out of the system, and this should equal the total flow, Q. If this is not the case, the total flow is set to $q_{N+1/2}$, and an informative message is produced.

After the flow rates have been computed, the maximum Reynold's number is also computed:

$$R_{max} = \frac{d\cdot Q/A}{3\times 10^{-7}\text{m}^2/\text{sec}}\ , \tag{21}$$

where d is the uniform diameter and A is the uniform cross-sectional area. This is included as a message in the code print out for information only. A Reynold's number over 2000 may indicate that the flow in the borehole is turbulent rather than laminar.

Calculations During Each Time Step

The mass transfer during time step k is due to flow from the feed points (the source of electrolyte), advection, and dispersion. The numerical solution of the partial differential equation (7) can be summarized as follows.

After multiplying by the uniform cell volume, $A\Delta x$, the finite-difference version of equation (7) may be written in terms of mass transfer (kg/sec) as follows:

$$\begin{aligned}(C_{i,\,k}/\Delta t)\,(A\Delta x) \;=\; & C^f_{i,\,k}\,(q^f_i) \\ & + C_{i-1,\,k-1}\,(q_{i-1/2}) \;-\; C_{i,\,k-1}\,(q_{i+i/2}) \\ & + (C_{i-1,\,k-1}-C_{i,\,k-1})(\gamma_{i-1/2}) - (C_{i,\,k-1}-C_{i+1,\,k-1})(\gamma_{i+1/2})\end{aligned} \qquad (22)$$

where $C^f_{i,\,k}$ is the average concentration of electrolyte in the fluid flowing from the feed points into cell i during time step k, and $\gamma_{i\pm1/2}$ is the dispersion coefficient at the interface between two adjoining cells adjusted for the problem geometry. The first line of the right-hand side is the source term, the second line is the advection term, and the third line is the dispersion term. Upstream weighting is used in the advective terms.

This equation can be rewritten by collecting coefficients of the different cell concentrations as:

$$\begin{aligned}(C_{i,\,k}/\Delta t)\,(A\Delta x) \;=\; & C^f_{i,\,k}(q^f_i) \\ & + C_{i-1,\,k-1}\,(q_{i-1/2}+\gamma_{i-1/2}) \\ & + C_{i,\,k-1}\,(-q_{i+i/2} - \gamma_{i-1/2} - \gamma_{i+1/2}) \\ & + C_{i+1,\,k-1}\,(\gamma_{i+1/2})\end{aligned} \qquad (23)$$

Then an NxN banded matrix, **T**, can be developed from the coefficients above to compute mass transfers. If $\mathbf{C}_k$ is the Nx1 matrix of concentrations at time k, and $\mathbf{F}_k$ is the Nx1 matrix of mass flows from the feed points, $F_{i,\,k} = C^f_{i,\,k}(q^f_i)$, the calculation can be stated as:

$$(\mathbf{T}\mathbf{C}_{k-1} + \mathbf{F}_k)\,(\Delta t/A\Delta x) = \mathbf{C}_k \qquad (24)$$

At the beginning of each time step the weighted average of concentration of all the feed point flow into each cell during the time step, $C^f_{i,\,k}$, is computed as follows:

$$C^f_{i,\,k} = \sum_j \chi(j,\,i)\; C^F_j (q^F_j / q^f_i)\, (\tau/\Delta t)\,, \tag{25}$$

where

$$\chi(j,\,i) = \begin{cases} 1 \text{ if feedpoint j flows into cell i} \\ 0 \text{ otherwise} \end{cases}$$

and

$$\tau = \begin{cases} 0 & \text{if } t_k \le t^F_j \\ t_k - t^F_j & \text{if } t^F_j < t_k < t^F_j + \Delta t \\ \Delta t & \text{if } t^F_j + \Delta t \le t_k \end{cases}$$

where t^F_j is the time feed point j begins flowing. It is assumed that once a feed point begins flowing with concentration C^F_j at time t^F_j it will continue to do so until the end of the problem.

At each time step k a check is made to verify that the total mass in the cell at the beginning of the time step is greater than or equal to the total mass to be transported out of the cell during the time step. If this condition is not met, an error message is printed and the run is stopped.

Conservation of mass is verified during each time step by comparing the total mass in the system at the end of the time step, $M_{s,\,k}$ with the difference of the cumulative mass flows into and out of the system, $M_{in,\,k}$ and $M_{out,\,k}$, respectively:

$$M_{s,\,k} = \sum_{i=1}^{N} (A\Delta x)\, C_{i,\,k} \tag{26}$$

$$M_{in,\,k} = \sum_{m=1}^{k} \sum_{i=1}^{N} \Delta t (C^f_{i,\,m})(q^f_i) \tag{27}$$

$$M_{out,\,k} = \sum_{m=1}^{k} \sum_{i=1}^{N} \Delta t (C_{N,\,m-1})(q_{N+1/2}) \tag{28}$$

A conservation of mass error is indicated (and the run stopped) if the difference between the actual mass in the system and the value expected based on mass in and out is greater than 0.1% of the total input mass:

$$\mid M_{s,\,k} - (M_{in,\,k} - M_{out,\,k}) \mid \; > 0.001\, M_{in,\,k} \tag{29}$$

At the end of the problem, the total of mass which has moved into the system from the feed points during the discrete calculation, $M_{in,\,M}$, is compared with the mass which would have flowed in during a continuous calculation, M_F. If these numbers differ by more than 0.01% of the continuous value, a message indicating the magnitude of the discretization error in mass inflow is indicated.

$$M_{in,\,M} = \sum_{k=1}^{M} \sum_{i=1}^{N} \Delta t(C^f_{i,\,k}) \tag{30}$$

$$M_F = \sum_{j} (t_M - t_j^F) q_j^F C_j^F \tag{31}$$

where t_j^F is the time at which feed point j begins flowing at rate q_j^F and concentration C_j^F. Flow continues until the end of the problem.

A message occurs if

$$| \; M_{in,\,M} - M_F \; | > 0.0001 \; M_F \, . \tag{32}$$

3. DESCRIPTION OF FORTRAN CODE

The main program performs initialization of data areas, sequences the preprocessing, controls the time step loop, and controls conductivity output. Calls are made to the input and preprocessing routines in the following order: RDHOLE, QEXTS, RDFRAC, FLOWS, TANDR, RDTIME, CHKT, RDK, RDC0, and RDCORR. An informative message indicating the problem size (the product of the number of time step and the number of segments) is then produced.

The time step loop consists of calculation of the starting and ending times of the step, a call to subroutine CTFRAC to compute the fracture inflow concentrations, and a call to subroutine TSTEP to compute mass transport. Following this a call to subroutine CPRT is made if conductivity output has been requested for a time contained in the time step.

At the end of the time interval, a final conductivity output is made if necessary, and the discrete and continuous calculations for the total mass flowing into the system from the fractures are compared by a call to the subroutine GTIN.

3.1 Subroutines

Subroutine CHKT

This subroutine verifies that the inflow starting times lie within the problem time limits. If not, a warning message is produced.

Subroutine CPRT

At each requested print time, this routine produces output consisting of the time, and lines containing three pairs of positions and conductivities.

Subroutine CTFRAC

During each time step the average concentration of all fracture fluid flowing into each segment is calculated. The mass contribution from each fracture during the time step is computed, and this is averaged over the total fracture flow into the segment for the interval. Because the flow is steady-state, only the concentrations may vary with time.

Subroutine FLOWS

The steady-state fluid flow through the region is determined by this routine. Total fracture flow into each segment, total flow into each segment, and total downstream flow out of each segment is computed. In addition, the total flow out of the top of the region is computed, and this is compared to the input value. An informational message is produced if there is an inconsistency.

Subroutine GTIN

At the conclusion of the problem this routine compares the accumulated mass in the system from the discrete time steps with the mass expected from a continuous calculation. If there is a discrepancy of more than 0.01% a message is produced.

Function ICONVX

This function determines the segment containing a given position.

Subroutine QEXTS

This subroutine reads the external flow data into the bottom and out of the top of the region described on input line 2.

Subroutine RDC0

If any initial concentrations are to be specified, they are read by subroutine RDC0 before the time steps begin. The number of specified initial concentrations is input on line 6, and a position and initial concentration are listed on each of the 6.i input lines.

Subroutine RDCORR

The coefficients to convert electrolyte concentration in kg/m^3 to conductivity in $\mu S/cm$ are input on line 7. The three coefficients for the second degree fit are entered, linear term first. If no conversion is desired, the coefficients 0,1,0 can be used.

Subroutine RDFRAC

The fracture data on input lines 3 and 3.i are read by this subroutine. This includes the number of fractures, and the position, flow rate, concentration, and starting time for each fracture.

The number of fractures is compared against the program maximum and the positions of the fractures are verified to lie within the segment limits. An error message is produced if any inconsistencies are detected.

Subroutine RDHOLE

This is an input routine executed once before the main loop. It reads line 1 of the input file containing the top and bottom positions of the area of interest, the segment length, and the uniform borehole diameter.

After reading the data, the subroutine verifies the relative locations of the top and bottom, adjusts the top to make it an multiple of the segment length from the bottom. The cross-sectional area and uniform segment volume are then computed, as well as the midpoint location of each segment.

Informational output from the routine consists of a message reporting the number of segments used, and the uniform diameter, cross-sectional area and volume of each segment.

Subroutine RDK

The dispersion coefficient and the flag indicating the method to be used in computing the coefficient values for each segment are contained on input line 5 which is read by this routine. After reading the values, this routine computes a dispersion coefficient for each segment based on the selected method: constant, velocity scaled, or velocity-squared scaled.

Subroutine RDTIME

This subroutine reads the problem time limits, time step, and print times on input records 4 and 4.i. The relationship between the starting and ending times is verified, and the ending time is adjusted if necessary to make the total time interval a multiple of the time step. If the time step is greater than the maximum time step allowed by advection, it is adjusted downward and a message is produced. In addition, a message indicating the number of time step is printed. If the number of time steps exceeds the program maximum, an error message is produced and execution halted.

Subroutine TANDR

The maximum time step based on advection and the maximum Reynold's number are computed by this routine after the steady-state flow field has been determined.

Subroutine TSTEP

This is the main mass transport calculation executed during each time step. The mass in each segment at the beginning of the time step, the changes in mass of each segment during the time step, and the mass in the segment at the end of the time step are computed. In addition, conservation of mass checks are performed.

3.2 Common Blocks

The program has four named common blocks, CORR, FRACS, SEGS and STEP.

Common block CORR contains the coeffiecents for converting electrolyte concentration in kg/m^3 to conductivity in μS/cm. These terms are derived from a second degree polynomial fit to experimental data, and are read by subroutine RDCORR.

Common block FRACS contains the arrays describing the fractures, including the fracture flow rates, concentrations, positions, segment locations, and starting times.

Common block SEGS contains the arrays describing each segment, including the concentration at the beginning and end of the time step, the fracture inflow average concentration, external flows into and out of the segment, fracture flow into the segment, downstream flow to the next segment, total flow into the segment, the position of the segment and the dispersion coefficient between the segment and the next downstream segment.

Common block STEP contains other variables used during the time steps, including the step duration, number of segments, maximum number of segments allowed, uniform segment volume, cumulative mass out of the system, cumulative mass into the system, time at the beginning and end of the time step, number of fractures, conductivity output unit number, and toggle index for the concentration array.

4. INPUT AND OUTPUT GUIDE

Input and Output Files

The model uses one input and two output files. The input file contains the problem description consisting of borehole geometry, top and bottom borehole flows, feed point flows, timing parameters, dispersion parameters, and initial concentrations. One of the output files consists of messages produced by the model, while the other contains position and conductivity pairs for each borehole cell at the requested output times. An interactive version of the code is also available which uses terminal input and output rather than the problem description and message files. The following table summarizes the input and output files and indicates their FORTRAN unit numbers.

Table 4-1. Input and Output Files.

UNIT NUMBER	INPUT/OUTPUT	DESCRIPTION
5	INPUT	Problem description
6	OUTPUT	Messages
7	OUTPUT	Computed conductivity data

Problem Description

The problem description is entered in free format, with values being separated by spaces or commas. The number of lines in the problem description will be variable depending on the number of feed points, number of times at which conductivity output is desired, and number of initial concentrations specified. The following table provides a detailed description of each line of the input.

Table 4-2. Problem description input guide.

LINE	NAME	UNITS	DESCRIPTION
1	XTOP	m	Top of study area, surface is zero and positions increase downward, adjusted if necessary to fit XBOT and DELX
	XBOT	m	Bottom of study area
	DELX	m	cell length
	DIAM	cm	Borehole diameter (uniform)
2	QEXTI(1)	m^3/sec	Flow in at bottom
	QEXTO(ILIM)	m^3/sec	Flow out at top, adjusted if necessary to fit total inflow
3	IFLIM	none	Number of feed points
3.I I=1, IFLIM	XIN(I)	m	Position of feed point
	QIN(I)	m^3/sec	Feed point flow rate
	CIN(I)	kg/m^3	Feed point fluid concentration
	T0(I)	hr	Feed point flow start time
4	TSTART	hr	Problem start time
	TEND	hr	Problem end time, may be recomputed to fit TSTART and DELT
	DELT	min	Time step, may be reduced to the maximum for flow rates and geometry
	ILPT	none	Number of print times
4.I I=1, ILPT	PT(I)	hr	Time at which concentration curve is to be, printed, actual curve is at next nearest discrete time
5	ITYPDK	none	Type code for dispersion coefficient 1 means DK is constant over all cells 2 means DK is the value for the mean velocity, $\overline{q}=(\min(q_i)+\max(q_i))/2$, $K_i=DK(q_i/\overline{q})$ 3 means DK is the value for mean vel. squared, $\overline{q^2}=(\min(q_i^2)+\max(q_i^2))/2$, $K_i=DK(q_i^2/\overline{q^2})$
	DK	m^2/sec	Dispersion coefficient

LINE	NAME	UNITS	DESCRIPTION
6	ICON	none	Number of initial concentrations
6.I I=1, ICON	X(I) CO(I)	m kg/m^3	Position of initial concentration Initial concentration
7	OFSET	$\mu S/cm$	constant term for converting concentration in kg/m^3 to conductivity in $\mu S/cm$
	COEFA	$(\mu S \cdot m^2)/100kg$	linear coefficient for converting concentration to conductivity
	COEFB	$(\mu S \cdot m^5)/10000kg^2$	quadratic coefficient for converting concentration to conductivity

Output Conductivity Data

The conductivity output file consists of pairs of position and conductivity values. Three pairs are output on each line of the file. In addition, the time in hours is output at the start of each set of values.

The output subroutine CPRT converts concentration to conductivity just prior to output by means of the coefficients in common CORR. If the coefficients 0,1,0 are entered, then the output values are equivalent to the concentrations.

5. CODE VALIDATION

5.1 Short Time Single Source

For short times with electrolyte inflow localized near the inflow points, the concentration is given by Fischer et al. (1979):

$$C(x,t) = qC_o(4\pi D)^{-1/2} \int_0^t (t-\tau)^{-1/2} \exp[-x^2/4D(t-\tau)]\, d\tau$$

The model results are compared with the analytical solution for short times (1, 4, 9 and 16 seconds) from a single fracture inflow at x=0 with $qC_o = 1$ and $D = 0.25$. The results, plotted together in Figure 5-1, are indistinguishable. Note that the lower boundary is not felt, so there is symmetry with respect to x=0.

The input problem description is:

LINE NO.	DATA
1	-9.975, 9.975, .05, 112.838
2	0, 1.e-5
3	1
3.1	0, 1.e-5, 1.e5, 0
4	0,4.5e-3,5.e-5,4
4.1	2.778e-4
4.2	1.111e-3
4.3	2.500e-3
4.4	4.444e-3
5	1,0.25
6	0
7	0,1,0

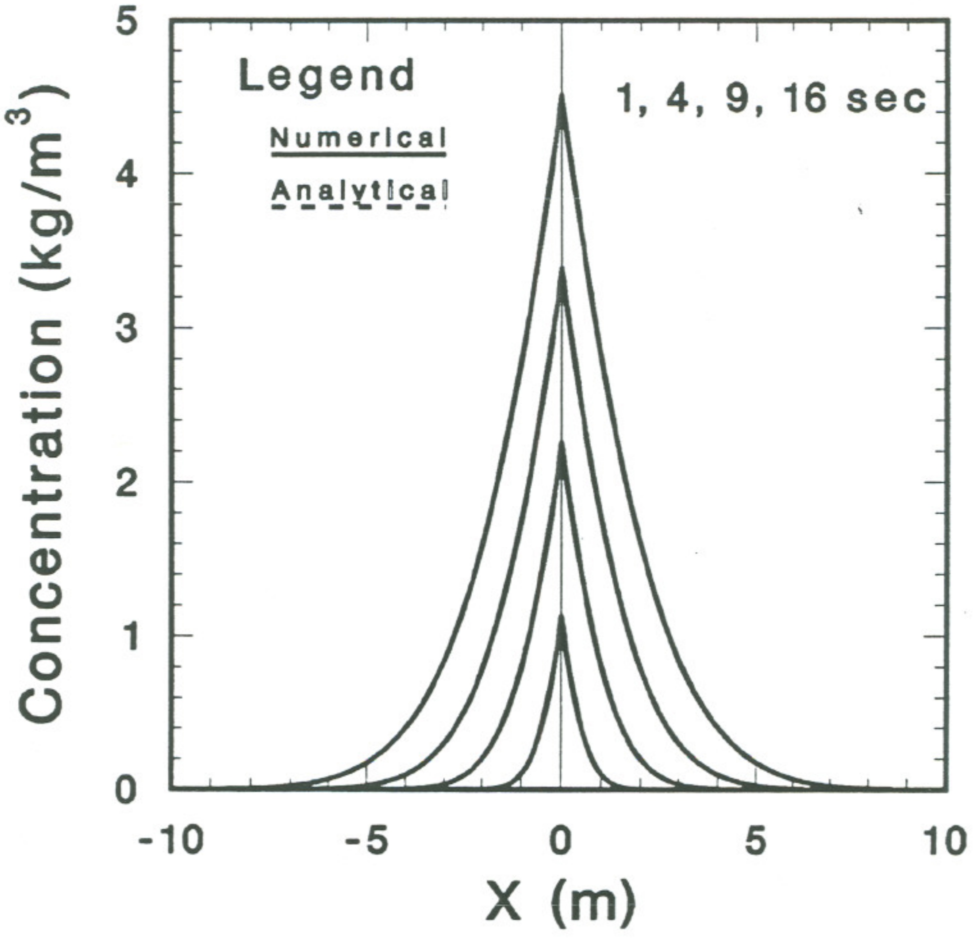

Figure 5-1. Short times, single source.

5.2 Constant Input Concentration Under Constant Overall Flow

When the inflow is well established and the constant wellbore flow dominates the transport of electrolyte, the following analytical solution is available, Javandel et al. (1984):

$$C/C_o = \frac{1}{2}\mathrm{erfc}\left(\frac{x-vt}{2(Kt)^{1/2}}\right) + \left(\frac{v^2t}{\pi K}\right)^{1/2} \exp\left[-\frac{(x-vt)^2}{4Kt}\right]$$
$$-\frac{1}{2}\left(1+\frac{vx}{K}+\frac{v^2t}{K}\right)\exp[vx/K]\ \mathrm{erfc}\left(\frac{x+vt}{2(Kt)^{1/2}}\right)$$

The model results are compared with the analytical solution for a constant input concentration at x=0 under constant overall flow in Figure 5-2; the agreement is excellent.

The input problem description is:

LINE NO.	DATA
1	-149.875, 0.125, .25, 112.838
2	0, 1.923e-5
3	1
3.1	0, 1.923e-5, 1, 0
4	0,1444.8,15,2
4.1	722.4
4.2	1444.8
5	1,1.923e-5
6	1
6.1	0,1
7	0,1,0

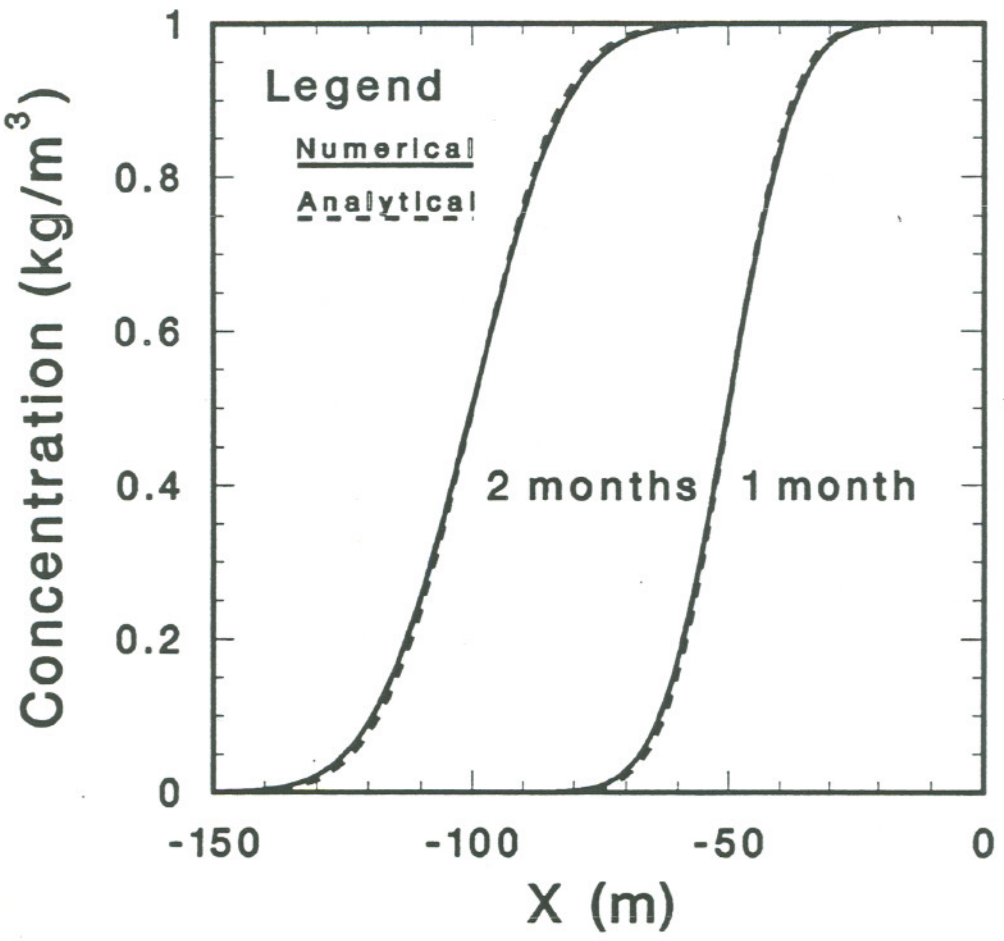

Figure 5-2. Constant overall flow.

5.3 Comparison with PT Results

PT is a verified and validated code which has been used to study heat and mass flows. PT was was developed at Lawrence Berkeley Laboratory and uses the Integral Finite Difference (IFD) method, Bodvarsson (1982).

The model PT was run to match field data for a borehole with nine fracture inflow points by using heat to mimic concentration. The comparison between the results from this model and those from PT are shown in Figures 5-3. The variation of the match from peak to peak is due to a different mesh.

The input problem description is:

LINE NO.	DATA
1	750, 1490, 5, 14
2	0, 1.e-5
3	9
3.1	1440, 0.635e-6, 2.283, 18
3.2	1300, 0.223e-6, 2.283, 18
3.3	1215, 0.486e-6, 2.283, 18
3.4	1200, 0.206e-6, 2.283, 18
3.5	1188, 0.530e-6, 2.283, 18
3.6	1085, 0.175e-6, 2.283, 18
3.7	1048, 0.517e-6, 2.283, 18
3.8	918, 0.219e-6, 56.755, 9
3.9	843, 7.010e-6, 4.994,-12
4	-13,60,30,5
4.1	13.05
4.2	27.20
4.3	31.63
4.4	38.68
4.5	57.40
5	1,0.75e-3
6	0
7	0,1,0

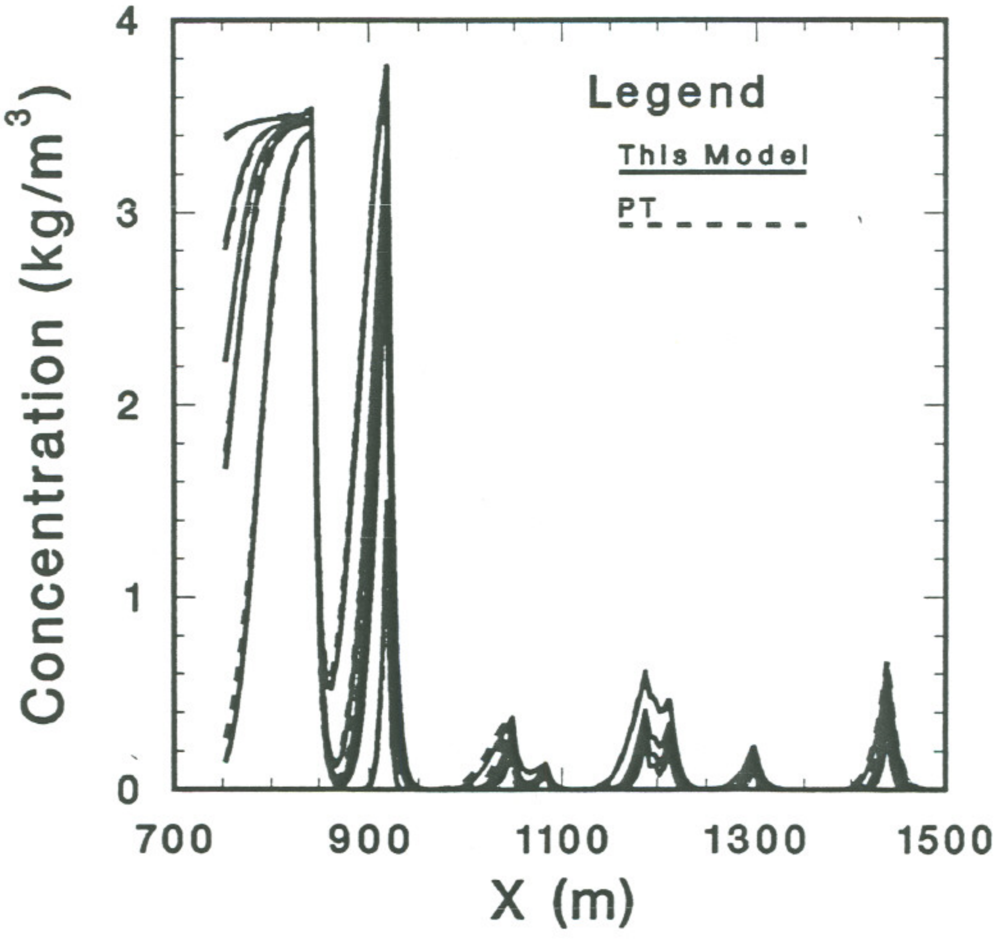

Figure 5-3. Comparison with PT.

5.4 Comparison with Field Data

The model was run to match field data for a borehole with nine fracture inflow points. The comparison between the results from one attempt at matching the field data are shown in Figure 5-3. This is an initial attempt at fitting the field data, and obviously not the best fit. The fracture inflow mass near 1300 meters should be decreased, and that near 900 m increased.

The input problem description is:

LINE NO.	DATA
1	750, 1490, 5, 14
2	0, 1.e-5
3	9
3.1	1440, 0.620e-6, 0.530, 18
3.2	1300, 0.440e-6, 0.520, 18
3.3	1215, 0.480e-6, 0.520, 18
3.4	1200, 0.200e-6, 0.550, 18
3.5	1188, 0.590e-6, 0.470, 18
3.6	1085, 0.170e-6, 0.530, 18
3.7	1048, 0.510e-6, 0.530, 18
3.8	918, 0.210e-6, 13.000, 9
3.9	843, 6.900e-6, 1.200,-12
4	-13,60,30,5
4.1	13.05
4.2	27.20
4.3	31.63
4.4	38.68
4.5	57.40
5	1,0.5e-3
6	0
7	73, 1870, -40

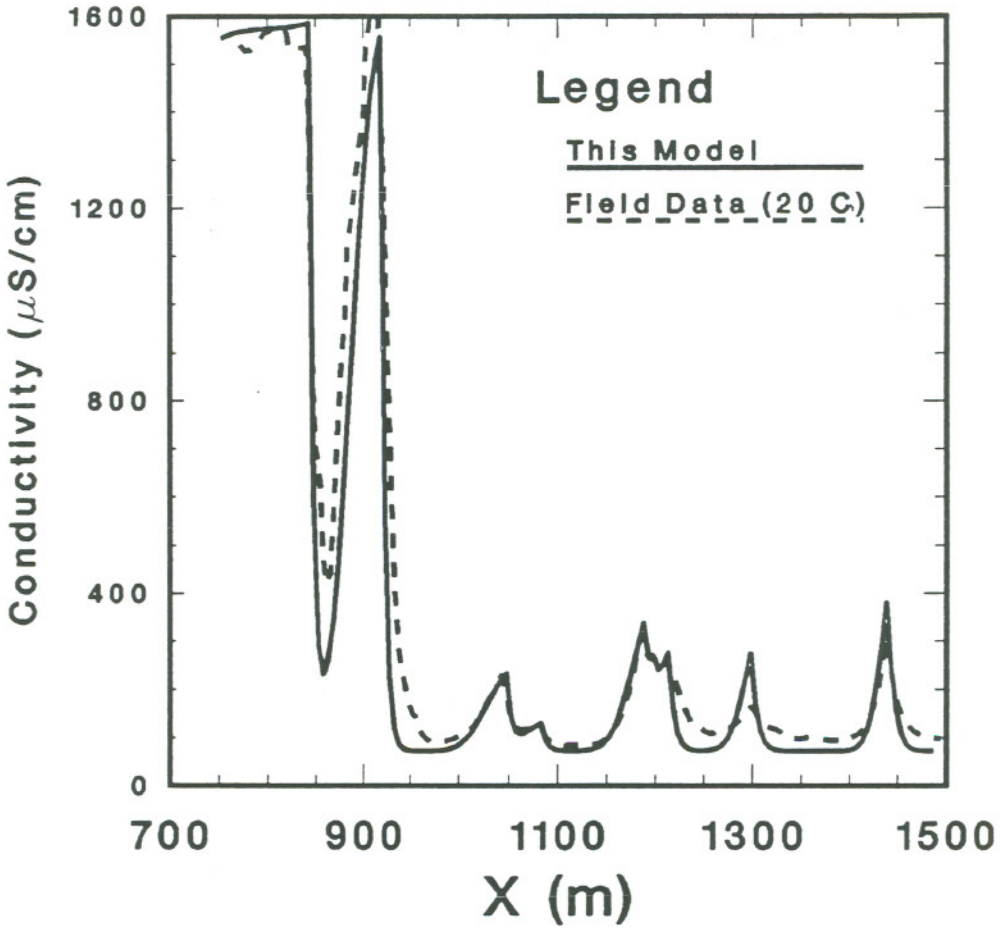

Figure 5-4. An attempt to match field data.

6. ACKNOWLEDGEMENTS

Discussions and cooperation with NAGRA personnel, especially P. Hufschmied, M. Thury, S. Vomvoris and P. Zuidema are much appreciated. Assistance in computation and graphing from C. Doughty is gratefully acknowledged. We would also like to thank S. Vomvoris and C. Doughty for reviewing and commenting on the manuscript and software. The work is supported by the Repository Technology Division, Office of Civilian Radioactive Waste Management, U.S. Department of Energy, through U.S. Department of Energy Contract No. DE-AC03-76SF00098.

7. REFERENCES

BODVARSSON, G.S., Mathematical modeling of the behavior of geothermal systems under exploitation, Ph.D. thesis, University of California at Berkeley, Department of Materials Sciences and Mineral Engineering, 1982.

FISCHER, H.B., LIST, E.J., KOH, R.C.Y., IMBERGER, J., BROOKS, N.H., *Mixing in Inland and Coastal Waters,* Academic Press, New York, 1979.

JAVANDEL, I., DOUGHTY, C., TSANG, C.F., *Groundwater Transport,* Water Resources Monograph No. 10, American Geophysical Union, Washington, D.C., 1984.

NAGRA, Private communication from Dr. Peter Hufschmied, NAGRA, Baden, Switzerland, 1987.

ONSAGER, L., *Physik. Z.,* **27,** 388, 1926.

ONSAGER, L., *Physik. Z.,* **28,** 277; *Trans. Faraday Soc.,* **23,** 341, 1927.

SHEDLOVSKY, T., SHEDLOVSKY, L., Conductometry, in *Techniques of Chemistry, Vol. 1: Physical Methods of Chemistry, Part IIA: Electrochemical Methods* A. Weissberger, ed., Wiley-Interscience, New York, 1971.

SCHLUMBERGER, Ltd., Log Interpretation Charts, New York, 1984.

TSANG, C.F., A Borehole Fluid Conductivity Logging Method for the Determination of Fracture Inflow Parameters, LBL-23096, NDC-1, Lawrence Berkeley Laboratory, Berkeley, CA, 1987.

Appendix A--FORTRAN Source Code

```
c***   borehole flow model
c***   june 1987
c***   lawrence berkeley laboratory, earth sciences division
c
c***   pt is print times
c
      dimension pt(25)
c
c***   common /segs/ contains segment data used during time step
c***      commons are used to speed up processing and avoid
c***      parameter passing during time steps
c***   cxt is concentration array at two times (i-1, i) (kg/m**3)
c***   cfrac is concentration in fracture flow into segment
c***      (kg/m**3)
c***   qexto is flow out of the system from each segment (m**3/sec)
c***   qexti is flow into the system at each segment (m**3/sec)
c***   qfrac is fracture flow into each segment (m**3/sec)
c***   qnxt is flow from one segment to the next (up) (m**3/sec)
c***   qtot is total flow through a segment
c***   xs is segment midpoint
c***   dks is harmonic mean dispersion coefficient between
c***      segments i and i+1
c
      common /segs/ cxt(1000,2),cfrac(1000),
     .              qexto(1000),qexti(1000),qfrac(1000),qnxt(1000),
     .              qtot(1000),xs(1000),dks(1000)
c
c***   common /fracs/ contains fracture data used during time step
c***   qin is flow rate at each fracture (m**3/sec)
c***   cin is concentration in flow at each fracture (kg/m**3)
c***   xin is position of fracture (m)
c***   ifseg is the segment in which the fracture is found
c***   t0 is the starting time for solute flow in the fracture (sec)
c
      common /fracs/ qin(100),cin(100),xin(100),ifseg(100),t0(100)
c
c***   common /step/ is used for additional time step data
c***   delt is time step (sec, input)
c***   ilim is number of segments (computed)
c***   vol is segment volume (m**3, computed)
c***   gtmout is grand total mass out of the system (kg)
c***   gtmina is actual grand total mass into the system (kg)
c***   t is time at end of time step(sec, computed)
c***   iout is general output unit number
c***   tp is time at beginning of time step (sec, computed)
c***   iflim is number of fractures (computed)
c***   iprt is concentration output unit number
c***   itog is toggle between two sets of C(x,t)
c***   tm0 is initial mass in system
```

```
c
      common /step/ delt,ilim,imax,vol,gtmout,gtmina,
     .              t,iout,tp,iflim,iprt,itog,tm0
c
c***   ofset is the background conductivity in S/m
c***   coefa is the linear term for converting concentrations
c***       in kg/m**3 to conductivity in S/m (at 20 C)
c***   coefb is the second degree term for converting
c***       concentrations to conductivity (at 20 C)
c
      common /corr/ ofset,coefa,coefb
c
c***   iin is input unit number
c***   itmax is maximum number of time steps
c***   impt is maximum number of print times
c***   imax is maximum number of segments
c***   ifmax is maximum number of fractures
c
      data iin/5/,itmax/10000/,impt/25/,ifmax/100/
c
c***   input and initialization
c
      imax=1000
      iout=6
      iprt=7
      gtmout=0.
      gtmina=0.
      itog=1
      do 100 i=1,imax
         cfrac(i)=0.
  100 continue
c
c***   note:  all input and initialization subroutines pass data
c***          by parameters.  all subroutines within the time step
c***          loop pass data by commons for speed
c
      call rdhole(xtop,xbot,delx,diam,area,vol,ilim,imax,xs,iin,iout)
      call qexts(qexto(1),qexti(1),xtop,xbot,delx,ilim,imax,iin,iout)
      call rdfrac(qin(1),cin(1),xin(1),ifseg(1),t0(1),ifmax,iflim,
     .            xtop,xbot,delx,iin,iout)
      call flows(qexto(1),qexti(1),qfrac(1),qnxt(1),qtot(1),
     .           qin(1),ifseg(1),ilim,iflim,iout)
      call tandr(qtot(1),ilim,diam,area,vol,dtmax,rnmax,iout)
      call rdtime(tstart,tend,delt,pt(1),ilpt,impt,itstep,itmax,
     .            dtmax,iin,iout)
      call chkt(t0(1),iflim,tstart,tend,iout)
      call rdk(dks(1),qtot(1),ilim,area,delx,iin,iout)
      call rdc0(cxt(1,1),imax,ilim,xtop,xbot,delx,vol,tm0,iin,iout)
      call rdcorr(ofset,coefa,coefb,iin)
      iprod=itstep*ilim
      write(iout,910)iprod
c
c***   loop over each time step
c
      do 1000 i=1,itstep
c
c***   compute interval time limits
c
         t=tstart+i*delt
         tp=t-delt
c
c***   compute fracture inflows
c
         call ctfrac
c
c***   compute mass transfer
c
         call tstep
c
c***   check for output requests
c
         do 600 j=1,ilpt
            if (t.ge.pt(j)) then
               call cprt
               pt(j)=2.*tend
            endif
  600    continue
c
c***   end of loop over time steps
c
 1000 continue
c
c***   check for final print
c
      needp=0
      do 1500 j=1,ilpt
         if (pt(j).lt.(2.*tend)) needp=1
 1500 continue
      if (needp.ne.0) call cprt
c
c***   compare discreet and continuous mass calculations
c
      call gtin(tend,cin(1),qin(1),t0(1),iflim,gtmina,iout)
c
c***   end of problem
c
      write(iout,900)
c
```

```
      stop
c
  900 format(//' *** PROBLEM FINISHED ***')
  910 format(/' Problem size parameter (segments*time steps) is',i15)
      end
c
c***   subroutine rdhole
c***   read borehole parameters
c***   this subroutine is executed only once before time steps
c
c***   xtop is top of analysis region (m)
c***        xtop may be adjusted during discretization
c***   xbot is bottom of analysis region (m)
c***     note: x increases downward, zero at surface
c***   delx is segment length (m)
c***   diam is borehole diameter (input in cm, coverted to m)
c***   area is cross-sectional area of hole (m**2)
c***   vol is segment volume (m**3)
c***   ilim is number of segment over analysis region
c
c***   input stream--one free format record
c***       xtop, xbot, delx, diam
c
      subroutine rdhole(xtop,xbot,delx,diam,area,vol,ilim,imax,
     .                  xs,iin,iout)
c
      dimension xs(imax)
      data pi/3.141592654/
c
c***   interactive prompt disabled
c
c     write(iout,900)
      read(iin,*)xtop,xbot,delx,diam
c
c***   convert diameter in cm to m
c
      diam=diam/100.
c
c***   check geometry
c
      if (xbot.lt.xtop) then
         write(iout,940)
         stop
      endif
      xdist=xbot-xtop
      xint=xdist/delx
      if (xint.lt.1) then
         write(iout,950)
         stop
      endif
      ilim=xint
      if (ilim.gt.imax) then
         write(iout,990)imax
         stop
      endif
      xtopi=xtop
      xtop=xbot-(ilim*delx)
      deltop=abs(xtop-xtopi)
c     write(iout,955)xbot,xtop
c
c***   if delta-x does not fit over the interval within 1% of
c***   a delta-x, then make a note
c
      if (deltop.gt.(0.01*delx)) then
         write(iout,960)ilim,delx
         write(iout,970)deltop,xtop
      else
         write(iout,960)ilim,delx
      endif
c
c***   calculate cross-sectional area and volume of a segment
c
      rad=diam/2.
      area=pi*rad*rad
      vol=area*delx
      write(iout,980)diam,area,vol
c
c***   compute segment midpoints
c***   note--sign reversed for graphing
c
      do 100 i=1,ilim
         xs(i)=xbot+(0.5-i)*delx
  100 continue
c
      return
c 900 format(' Enter top and bottom of region (m), delta-x (m), and'
c    .        ' diameter (cm)')
  940 format(' Top of region is below bottom of region--aborting')
  950 format(' delta-x is greater than region size--aborting')
c 955 format(/' Analysis region is from ',f10.4,' to ',f10.4,' (m)')
  960 format(' Region has been divided into ',i4,' segments with',
     .        ' length ',f10.4,' (m)')
  970 format(' *** NOTE *** Top of region has been moved by ',
     .       f10.4,' (m) to ',f10.4,' (m)')
  980 format(' Each region has diameter            ',f12.5,' (m)'/
     .        '                  cross-sectional area ',f12.5,' (m**2)'/
     .        '              and volume                ',f12.5,' (m**3)')
  990 format(' Maximum number of segments is ',i4,' -- aborting')
```

```
      end
c
c***   subroutine qexts
c***   read external (non-fracture) flows
c***   input stream (free format)
c***        flow in at bottom and out at top (m**3/sec)
c***        number of additional inflows  (inactive)
c***           for each inflow: position (m) and rate (m**3/sec)
c***        number of additional outflows  (inactive)
c***           for each outflow: position (m) and rate (m**3/sec)
c***   this subroutine is executed only once before time steps
c
      subroutine qexts(qexto,qexti,xtop,xbot,delx,ilim,imax,iin,iout)
c
      dimension qexto(imax),qexti(imax)
c
      do 10 i=1,ilim
         qexto(i)=0.
         qexti(i)=0.
   10 continue
c
c***   top and bottom flows
c***   interactive prompt disabled
c
c     write(iout,900)
      read(iin,*)qexti(1),qexto(ilim)
c
c***   additional non-fracture inflows (disabled)
c
c     write(iout,920)
c     read(iin,*)n
c     if (n.gt.0) then
c        write(iout,930)
c        do 100 i=1,n
c           read(iin,*)xin,qin
c           if ((xin.gt.xbot .or. xin.lt.xtop)) then
c              write(iout,935)xin
c              goto 100
c           endif
c           ixin=iconvx(xtop,xbot,delx,xin)
c           qexti(ixin)=qexti(ixin)+qin
c 100    continue
c     endif
c
c***   additional non-fracture outflows (disabled)
c
c     write(iout,950)
c     read(iin,*)n
c     if (n.gt.0) then
```

```
c         write(iout,960)
c         do 200 i=1,n
c            read(iin,*)xout,qout
c            if ((xout.lt.xbot .or. xout.gt.xtop)) then
c               write(iout,935)xout
c               goto 200
c            endif
c            ixout=iconvx(xtop,xbot,delx,xout)
c            qexto(ixout)=qexto(ixout)+qout
c 200     continue
c      endif
c
c***    write summary of nonfracture external flows (disabled)
c
c      write(iout,980)
c      do 300 i=1,ilim
c         if ((qexti(i).ne.0.) .or. (qexto(i).ne.0.)) then
c            xb=xbot+(1-i)*delx
c            xt=xb-delx
c            write(iout,990)i,xt,xb,qexti(i),qexto(i)
c         endif
c 300 continue
      return
c
c 900 format(/' Enter flow into bottom and out of top of region',
c    .          ' (m**3/sec)')
c 920 format(' Enter number of additional non-fracture external',
c    .          ' inflows')
c 930 format(' Enter position (m) and rate (m**3/sec) of each inflow')
c 935 format(' *** NOTE *** flow position outside of region ',f10.4,
c    .          ' -- ignored')
c 950 format(' Enter number of additional non-fracture external',
c    .          ' outflows')
c 960 format(' Enter position (m) and rate (m**3/sec) of each outflow')
c 980 format(/' Summary of external non-fracture flows'/
c    .         /' Segment    Top       Bottom    In (m**3/sec)',
c    .          '  Out (m**3/sec)')
c 990 format(i6,2f11.4,2e15.5)
      end
c
c***    function iconvx
c***    convert x position to segment index
c***    segment 1 begins at xbot
c***    segment ilim ends at xtop
c***    this function is used outisde of time steps
c
      function iconvx(xtop,xbot,delx,x)
c
      idxup=(xbot-x)/delx+1.
      if (x.eq.xtop) idxup=idxup-1
      iconvx=idxup
      return
      end
c
c***   subroutine rdfrac
c***   read flow from fractures
c***   input stream -- free format
c***      number of fractures
c***      one record for each fracture --
c***         position (m), flow rate (m**3/sec), concentration of
c***         solute (kg/m**3), and solute flow start time (h)
c***   this subroutine is exectued once before time steps
c
      subroutine rdfrac(qin,cin,xin,ifseg,t0,ifmax,iflim,
     .           xtop,xbot,delx,iin,iout)
c
      dimension qin(ifmax),cin(ifmax),xin(ifmax),ifseg(ifmax),t0(ifmax)
c
c***   interactive prompt disabled
c***   read fracture data
c
c     write(iout,900)
      read(iin,*)iflim
      if (iflim.gt.ifmax) then
         write(iout,910)ifmax
         stop
      endif
c
      j=iflim
c
c***   interactive prompt disabled
c
c     write(iout,920)
      do 100 i=1,j
         read(iin,*)xin(i),qin(i),cin(i),t0(i)
c
c***   locate fracture on segments
c
         if ((xin(i).gt.xbot) .or. (xin(i).lt.xtop)) then
            write(iout,930)xin(i)
            iflim=iflim-1
            goto 100
         endif
         ifseg(i)=iconvx(xtop,xbot,delx,xin(i))
c
c***   convert t0 to seconds
c
         t0(i)=t0(i)*3600.
```

```
  100 continue
c
c***    write fracture summary (disabled)
c
c      write(iout,940)
c      do 200 i=1,iflim
c         write(iout,950)xin(i),qin(i),cin(i),t0(i),ifseg(i)
c 200 continue
      return
c
c 900 format(/' Enter number of fractures flowing into borehole')
  910 format(' Maximum number of fractures is ',i4,' -- aborting')
c 920 format(' Enter position (m), flow rate (m**3/sec),'/
c    .       '      solute concentration (kg/m**3), and starting'/
c    .       '      time (h) of solute flow for each fracture'/)
  930 format(' *** NOTE *** fracture position outside of region ',f10.4,
     .       ' -- ignored')
c 940 format(/' Summary of fracture input'/
c    .       /'  Position',8x,'Rate',6x,'Concentration',
c    .       ' Start time   Seg.')
c 950 format(f11.4,e15.5,e15.5,f15.4,i5)
      end
c
c***    subroutine flows
c***    calculate flows through each segment
c***    this subroutine is executed once before the time steps
c
      subroutine flows(qexto,qexti,qfrac,qnxt,qtot,
     .           qin,ifseg,ilim,iflim,iout)
c
      dimension qexto(ilim),qexti(ilim),qfrac(ilim),qnxt(ilim),
     .          qtot(ilim)
      dimension qin(iflim),ifseg(iflim)
c
c***    initialize segment flows
c
      do 100 i=1,ilim
         qfrac(i)=0.
  100 continue
c
c***    compute flow into segments from fractures
c***    it is possible to have more than one fracture in a single
c***    segment
c
      do 200 i=1,iflim
         qfrac(ifseg(i))=qfrac(ifseg(i))+qin(i)
  200 continue
c
c***    compute flow up to next segment and total flow
```

```
c
      do 300 i=1,ilim
         if (i.eq.1) then
            qtot(i)=qexti(i)+qfrac(i)
            qnxt(i)=qtot(i)-qexto(i)
         else
            qtot(i)=qexti(i)+qnxt(i-1)+qfrac(i)
c
c***   adjust input value of outflow at top to match total flow
c
            if (i.eq.ilim) then
               qe = qexto(i)
               qexto(i)=qtot(i)
               dq = abs(qe-qexto(i))
               if (dq .gt. (0.01*qe)) then
                  write(iout,900)dq,qexto(i)
               endif
            else
               qnxt(i)=qtot(i)-qexto(i)
            endif
         endif
  300 continue
c
      return
c
  900 format(/' *** NOTE *** Outflow at top changed by ',e15.5,
     .       ' to ',e15.5)
      end
c
c***   subroutine tandr
c***   compute maximum time step and Reynold's number
c***   this subroutine is executed once before the time steps
c
      subroutine tandr(qtot,ilim,diam,area,vol,dtmax,rnmax,iout)
c
      dimension qtot(ilim)
c
      qtmax=abs(qtot(1))
      do 100 i=2,ilim
         if (abs(qtot(i)).gt.qtmax) qtmax=abs(qtot(i))
  100 continue
c
c***   maximum time step is time minimum time to flush one-half
c***   of a segment.  note that this time may still be too large
c***   if dispersion is significant
c
      dtmax=0.5*vol/qtmax
      vmax=qtmax/area
      rnmax=vmax*diam/0.3e-6
      dtm=dtmax/60.
      write(iout,900)dtm,vmax,rnmax
      return
c
  900 format(/' Maximum time step (min) is         ',e15.5
     .       /'   (based on advection only)'
     .       /' Maximum linear velocity (m/sec) is ',e15.5
     .       /' Maximum Reynold''s number is         ',e15.5)
      end
c
c***   subroutine rdtime
c***   read time parameters
c***   input stream--
c***      start time (hrs), end time (hrs), delta time (min),
c***         number of prints
c***      then, print times (hrs)
c***    note: all times are converted to seconds
c***   this subroutine is executed only once before time steps
c
      subroutine rdtime(tstart,tend,delt,pt,ilpt,impt,itstep,itmax,
     .                  dtmax,iin,iout)
c
      dimension pt(impt)
c
c***   read in time parameters
c***   interactive prompt disabled
c
c     write(iout,900)
      read(iin,*)tstart,tend,delt,ilpt
      tstart=tstart*3600.
      tend=tend*3600.
      delt=delt*60.
c
      if (tend.lt.tstart) then
         write(iout,910)
         stop
      endif
      if (ilpt.gt.impt) then
         write(iout,915)impt
      endif
c
c***   read in print-time parameters
c***   interactive prompt disabled
c
c     write(iout,917)
      read(iin,*)(pt(i),i=1,ilpt)
      do 100 i=1,ilpt
         pt(i)=pt(i)*3600.
  100 continue
```

```
c
c***   check for maximum delta-t
c
      if (delt.gt.dtmax) then
         delt=dtmax
         dtm=delt/60.
         write(iout,920)dtm
      endif
c
c***   compute number of time steps
c
      itstep=(tend-tstart)/delt
      tendx=tstart+itstep*delt
      diff=tendx-tend
      if (diff.lt.0.) itstep=itstep+1
      if (itstep.gt.itmax) then
         write(iout,925)itmax
         stop
      endif
c
c***   adjust ending time if needed
c
      tendx=tend
      tend=tstart+itstep*delt
      diff=abs(tend-tendx)
c
c     tsh=tstart/3600.
      teh=tend/3600.
c     write(iout,930)tsh,teh
      if (diff.gt.(0.01*delt)) then
         dh=diff/3600.
         write(iout,940)dh,teh
      endif
      dtm=delt/60.
      write(iout,950)itstep,dtm
      return
c
c 900 format(/' Enter start (h), end (h), delta (m) and no. of prints')
  915 format(/' Maximum number of print times is ',i4,' -- aborting')
c 917 format(' Enter print times (h)')
  910 format(/' End time before start time -- aborting')
  920 format(/' *** NOTE ***  time step changed to maximum of ',
     .        f15.5,' (min)')
  925 format(/' Maximum number of time steps is ',i6,' -- aborting')
c 930 format(/' Beginning and ending times are (hrs):',2f15.5)
  940 format(' *** NOTE *** Ending time changed by ',f15.5,' to ',
     .        f15.5,' (hrs)')
  950 format(' Using ',i6,' time steps of ',f15.5,' (min)')
      end
```

```
c
c***   subroutine ctfrac
c***   prepare fracture concentrations for time step
c***   this subroutine is executed during each time step,
c***      and it loops over each fracture
c
      subroutine ctfrac
c
      common /segs/ cxt(1000,2),cfrac(1000),
     .              qexto(1000),qexti(1000),qfrac(1000),qnxt(1000),
     .              qtot(1000),xs(1000),dks(1000)
      common /fracs/ qin(100),cin(100),xin(100),ifseg(100),t0(100)
      common /step/ delt,ilim,imax,vol,gtmout,gtmina,
     .              t,iout,tp,iflim,iprt,itog,tm0
c
c***   reset values first
c
      do 10 i=1,iflim
         cfrac(ifseg(i))=0.
   10 continue
c
c***   compute mass inflow during time step
c***   more than one fracture may be in a segment
c
      do 100 i=1,iflim
         if (t.le.t0(i)) goto 100
         is=ifseg(i)
         if (tp.ge.t0(i)) then
c
c***   inflow of mass flow during entire time step (kg/sec)
c
            cfrac(is) = cfrac(is) + cin(i)*qin(i)/qfrac(is)
         else
c
c***   inflow of mass during part of time step (kg/sec)
c
            cfrac(is) = cfrac(is) +
     .                  cin(i)*qin(i)*((t-t0(i))/delt)/qfrac(is)
         endif
  100 continue
c
      return
      end
c
c***   subroutine rdk
c***   read dispersion parameter
c***   this subroutine is only executed once before time steps
c***   input stream--(one record)
c***      type code, parameter
```

```
c***   at the end of this routine, dks has units m**3/sec
c
      subroutine rdk(dks,qtot,ilim,area,delx,iin,iout)
c
c***   frac holds the multipliers for schemes 2 and 3
c
      dimension dks(ilim),qtot(ilim)
      dimension frac(1000)
c
c***   read dispersion parameter
c***   interactive prompt disabled
c
c     write(iout,900)
      read(iin,*)itypdk,dk
c
c***   if itypdk=1, value is constant over all segments
c***   if itypdk=2, value is for mean velocity
c***   if itypds=3, value is for mean velocity squared
c
      dksi=dk*area/delx
      if (itypdk.eq.1) then
         do 100 i=1,ilim
            dks(i)=dksi
  100    continue
      else
c
c***   find first nonzero, minimum, maximum and mean velocities
c
         qfnz=0.
         qmin=1.e20
         qmax=0.
         if (itypdk.eq.2) then
            do 200 i=1,ilim
               if ((qfnz.eq.0.).and.(qtot(i).ne.0.)) qfnz=qtot(i)
               if (qtot(i).gt.qmax) qmax=qtot(i)
               if (qtot(i).lt.qmin) qmin=qtot(i)
  200       continue
         else
            do 210 i=1,ilim
               qtoti2=qtot(i)*qtot(i)
               if ((qfnz.eq.0.).and.(qtoti2.ne.0.)) qfnz=qtoti2
               if (qtoti2.gt.qmax) qmax=qtoti2
               if (qtoti2.lt.qmin) qmin=qtoti2
  210       continue
         endif
         qavg=(qmin+qmax)/2.
         if (qavg.eq.0.) then
            write(iout,910)
            stop
         endif
c
         do 250 i=1,ilim
            if (qtot(i).ne.0.) then
               qtoti=qtot(i)
               if (itypdk.eq.3) qtoti=qtoti*qtoti
c
c***   use first nonzero flow for zero flow area
c
            else
               qtoti=qfnz
            endif
            frac(i)=qtoti/qavg
  250    continue
c
c***   compute harmonic mean of scaled dispersion parameter
c
         j=ilim-1
         do 300 i=1,j
            qx=2./( (1./frac(i)) + (1./frac(i+1)) )
            dks(i)=dksi*qx
  300    continue
      endif
      return
c
c 900 format(/' Enter dispersion parameter (m**2/sec)')
  910 format(/' *** NOTE *** average flow rate is zero -- aborting')
      end
c
c***    subroutine tstep
c***    calculate concentrations at next time step
c***    this is the main mass transport time step routine
c***    executed for each time step, and loops over each segment
c
      subroutine tstep
c
      common /segs/ cxt(1000,2),cfrac(1000),
     .              qexto(1000),qexti(1000),qfrac(1000),qnxt(1000),
     .              qtot(1000),xs(1000),dks(1000)
      common /step/ delt,ilim,imax,vol,gtmout,gtmina,
     .              t,iout,tp,iflim,iprt,itog,tm0
c
c***    tm1  -- initial mass in segment
c***    tm2  -- final mass in segment
c***    dmal -- rate change in mass due to advection from left (+ = in)
c***    dmar -- rate change in mass due to advection to right (- = out)
c***    dmdl -- rate change in mass due to diffusion with left (+ = in)
c***    dmdr -- rate change in mass due to diffusion with right (- = out)
c***    dmf  -- rate change in mass due to fracture inflow (+ = in)
```

```
c***    dme  -- rate change in mass due to external outflow (- = out)
c
c
c***    calculate mass changes
c***    gtm is total mass in system at end of time step
c
      gtm=0.
c
c***    generalized outflow disabled
c
      dme=0.
c
c***    loop over each segment
c
      itogx=3-itog
      do 100 i=1,ilim
         ipl=i+1
         cxti=cxt(i,itog)
         tml=cxti*vol
         if (cfrac(i).ne.0.) then
            dmf=cfrac(i)*qfrac(i)
         else
            dmf=0.
         endif
c
c***    generalized outflow disabled
c***    outflow considered only at top segment
c
c        dme=-cxti*qexto(i)
c
         if (i.gt.1) then
            iml=i-1
c
c***    independent calculation of mass transfer with left side
c***    disabled--using complement of right side values from below
c
c           if (qnxt(iml).gt.0.) then
c              dmal=cxt(iml,itog)*qnxt(iml)
c           else
c              dmal=cxti*qnxt(iml)
c           endif
c           dmdl=(cxt(iml,itog)-cxti)*dks(iml)
            dmal=-dmar
            dmdl=-dmdr
         else
            dmal=0.
            dmdl=0.
         endif
         if (i.lt.ilim) then
```

```
c
c***     note:  full upstream weighting is used.  without
c***            upstream weighting, these equations would
c***            use (cxti+cxt(ipl,itog))/2. in place of
c***            cxti or cxt(ipl,itog)--that is, the
c***            concentration at the interface would be
c***            the average instead of the upstream value
c
               if (qnxt(i).gt.0.) then
                  dmar=-cxti*qnxt(i)
               else
                  dmar=-cxt(ipl,itog)*qnxt(i)
               endif
               dmdr=(cxt(ipl,itog)-cxti)*dks(i)
            else
               dmar=0.
               dmdr=0.
c
c***     outflow considered only at top
c
               dme=-cxti*qexto(i)
            endif
c
c***     check for segment mass limitations
c***     if more mass is being moved out of a segment than is in
c***     the segment, write an error message and stop.  This problem
c***     can be fixed by reducing the time step or increasing the
c***     segment length
c
            summo=0.
            if (dme.lt.0.) summo=summo+dme
            if (dmal.lt.0.) summo=summo+dmal
            if (dmdl.lt.0.) summo=summo+dmdl
            if (dmar.lt.0.) summo=summo+dmar
            if (dmdr.lt.0.) summo=summo+dmdr
            if (abs(summo*delt).gt.tml) then
               fact=abs(tml/(summo*delt))
               write(iout,910)fact
               stop
            endif
c
c***     compute new mass in segment
c
            tm2=tml+(dmf+dme+dmal+dmdl+dmar+dmdr)*delt
            cxt(i,itogx)=tm2/vol
c
c***     check for conservation of mass
c***     gtmina is total inflow from fractures
c***     gtout is sum of external outflows
```

```
c
         gtmout=gtmout-(dme*delt)
         gtmina=gtmina+(dmf*delt)
         gtm=gtm+tm2
  100 continue
c
c***   end of loop over segments
c
c***   emass is expected total mass in system (total in - total out)
c***   gtm is actual total mass is system
c***   if these numbers differ by more than 0.1% of total input mass,
c***      signal a mass conservation error.  this requires that the
c***      numerical accuracy of all the discrete mass calculations be
c***      at least three digits (99.9%).
c***   this is executed once each time step
c
      if (gtmina.ne.0.) then
         emass=(tm0+gtmina)-gtmout
         errm=gtm-emass
         errp=(100.*errm)/(tm0+gtmina)
         if (abs(errp).gt.0.1) then
            write(iout,900)errp
            stop
         endif
      endif
c
c***   toggle row pointer
c
      itog=itogx
      return
c
  900 format(' *** NOTE *** mass conservation error ',f10.5,' (%)')
  910 format(' *** NOTE *** time step too big for advection '/
     .       '              and dispersion transport -- aborting'/
     .       '              multiply time step by factor of ',f10.5/
     .       '              or increase segment length')
      end
c
c***   subroutine cprt
c***   print out concentration arrays
c***   this subroutine is executed whenever concentration arrays
c***   are requested during the time steps, or at the end of the
c***   problem
c***   output format is:
c***     "  xxx.xxx"        (time in hours, 3 decimal places, quoted)
c***    xxx ccc xxx ccc xxx ccc  (three sets of x, S(x,t) on a line,
c***                              for as many lines as needed)
c***   this format is for ISSCO plotting software, but can be
c***   modified for any other plotting package
```

```
c***   note:  units of output conductivity is S/m
c
      subroutine cprt
c
      dimension sigma(1000)
      common /segs/ cxt(1000,2),cfrac(1000),
     .              qexto(1000),qexti(1000),qfrac(1000),qnxt(1000),
     .              qtot(1000),xs(1000),dks(1000)
      common /step/ delt,ilim,imax,vol,gtmout,gtmina,
     .              t,iout,tp,iflim,iprt,itog,tm0
      common /corr/ ofset,coefa,coefb
c
c***   output time in hours and concentrations with ascending
c***   values (note negative increment in loop)
c
      th=t/3600.
      write(iprt,900)th
      isig=1
      if (isig.eq.1) then
         do 100 i=1,ilim
            ci=cxt(i,itog)
            sigma(i)=(ofset + ci*coefa + ci*ci*coefb)
  100    continue
         write(iprt,910)(xs(i),sigma(i),i=ilim,1,-1)
      else
         write(iprt,910)(xs(i),cxt(i,itog),i=ilim,1,-1)
      endif
      write(iout,920)th
      return
c
  900 format(' "',f15.3,'"')
  910 format(3(f10.3,e15.5))
  920 format(/' Printing conductivity at time ',e15.5,' (hrs)')
      end
c
c***   subroutine gtin
c***   calculate expected grand total mass into the system
c***       for entire problem, and compare with discretized mass flow
c***   this subroutine is only executed once at the end of the problem
c**       to check on mass inflow discretization
c
      subroutine gtin(tend,cin,qin,t0,iflim,gtmina,iout)
c
      dimension cin(iflim),qin(iflim),t0(iflim)
c
c***   compute continuous mass inflow
c
      gtmin=0.
      do 100 i=1,iflim
```

```
            dt=tend-t0(i)
            if (dt.gt.0) gtmin = gtmin + (dt*qin(i)*cin(i))
  100 continue
c
c***   compute difference between continuous and discreet
c***   mass inflows
c
      aerr=gtmina-gtmin
      rerr=(100.*aerr)/gtmin
c
c***   if discretization error in total mass into system is
c***   greater than 0.01%, write an information message
c
      if (abs(rerr).gt.0.01) write(iout,900)gtmin,gtmina,aerr,rerr
      return
c
  900 format(/' Expected mass into system: ',e15.5,' (kg)'
     .       /' Discrete mass into system: ',e15.5,' (kg)'
     .       /' Discretization error:      ',e15.5,' (kg)'
     .       /' Percent error:             ',e15.5,' (%)')
      end
c
c***   subroutine chkt
c***   check fracture flow times to make sure they fit into problem
c
      subroutine chkt(t0,iflim,tstart,tend,iout)
c
      dimension t0(iflim)
c
      do 100 i=1,iflim
         if ((t0(i).lt.tstart) .or. (t0(i).gt.tend)) then
            th=t0(i)/3600.
            write(iout,900)i,th
         endif
  100 continue
      return
c
  900 format(/' *** NOTE *** fracture ',i3,' t0 at ',f15.5,' (hrs)'/
     .       '      is outside of problem time limits')
      end
c
c***   subroutine rdc0
c***   read initial concentrations
c***   input records:
c***      ic0n n=number of initial values specified
c***      x, c0  pairs of x values and initial concentrations
c
      subroutine rdc0(cxt,imax,ilim,xtop,xbot,delx,vol,tm0,iin,iout)
c
```

```
      dimension cxt(imax,2)
c
      do 100 i=1,ilim
         cxt(i,1)=0.
  100 continue
      tm0=0.
c
      read(iin,*)ic0n
      if (ic0n .gt. 0) then
         do 500 i=1,ic0n
            read(iin,*)x,c0
            if ((x.gt.xbot) .or. (x.lt.xtop)) then
               write(iout,900)x
            else
               iseg=iconvx(xtop,xbot,delx,x)
               if (cxt(iseg,1).eq.0.) then
                  cxt(iseg,1)=c0
                  tm0=tm0 + vol*c0
               else
                  write(iout,910)x
               endif
            endif
  500    continue
      endif
c
      return
c
  900 format(/' *** NOTE *** initial concentration position',
     .       ' invalid--ignored'
     .       /'                    position is ',f15.5)
  910 format(/' *** NOTE *** multiple initial concentrations',
     .       ' for one segment'
     .       /' second value ignored--position is ',f15.5)
      end
c
c***   subroutine rdcorr
c***   read correlation between concentration and conductivity:
c***
c***   ofset is the background conductivity in S/m
c***   coefa is the linear term for converting concentrations
c***       in kg/m**3 to conductivity in S/m (at 20 C)
c***   coefb is the second degree term for converting
c***       concentrations to conductivity (at 20 C)
c
      subroutine rdcorr(ofset,coefa,coefb,iin)
c
      read(iin,*)ofset,coefa,coefb
      return
      end
```