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

Permalink
https://escholarship.org/uc/item/02f5g3rg

Authors
Hale, F.V.
Tsang, C.F.

Publication Date
1988-03-01
A Code to Compute Borehole Fluid Conductivity Profiles with Multiple Feed Points

F.V. Hale and C.F. Tsang

March 1988

TWO-WEEK LOAN COPY

This is a Library Circulating Copy which may be borrowed for two weeks.
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
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: \( t_{ci} \), the time when the fracture fluid emerges at the wellbore; \( x_i \), the location of the inflow point; \( q_i^v \), 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):

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

(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:

\[ T_x = Ax + B \]  

(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⁺, Ca²⁺, Mg²⁺, Cl⁻, SO₄²⁻, and HCO₃⁻. 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:

<table>
<thead>
<tr>
<th>Concentration, C</th>
<th>Equivalent Conductance, κ</th>
<th>Conductivity, σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>equiv/liter</td>
<td>kg/m³</td>
<td>S cm²/mole</td>
</tr>
<tr>
<td>0.</td>
<td>0.</td>
<td>126.45</td>
</tr>
<tr>
<td>0.0005</td>
<td>0.0292</td>
<td>124.50</td>
</tr>
<tr>
<td>0.001</td>
<td>0.0584</td>
<td>123.74</td>
</tr>
<tr>
<td>0.005</td>
<td>0.292</td>
<td>120.65</td>
</tr>
<tr>
<td>0.01</td>
<td>0.584</td>
<td>118.51</td>
</tr>
<tr>
<td>0.02</td>
<td>1.17</td>
<td>115.76</td>
</tr>
<tr>
<td>0.05</td>
<td>2.92</td>
<td>111.06</td>
</tr>
<tr>
<td>0.10</td>
<td>5.84</td>
<td>106.74</td>
</tr>
</tbody>
</table>
The conductivity $\sigma$ can be calculated by multiplying the equivalent concentration $C$ in equivalents per liter times the equivalent conductance $\kappa$ with appropriate conversion factors:

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

or, more concisely,

$$\sigma = 1,000 C \kappa$$

where $C$ is the equivalent concentration in equiv/l, $\kappa$ is the equivalent conductance in S cm$^2$/mole, and $\sigma$ is the conductivity in $\mu$S/cm. These values are tabulated above.

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

$$\sigma = 2,075 C - 45 C^2$$

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

$$\sigma = 2,075 C$$

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

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

$$\sigma = 1,850 C - 40 C^2 .$$
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:

\[ \frac{\partial}{\partial x} \left( K_i \frac{\partial C}{\partial x} \right) - \frac{\partial}{\partial x} \left( C_i V_i \right) + S_i = \frac{\partial C_i}{\partial t} \]  

(7)

where

- \( C \) is the concentration (kg/m³)
- \( K \) is the dispersion coefficient (m²/sec)
- \( S \) is the source term (kg/m³·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:

\[ C(x, 0) = C_0(x) \]  

(8)

\[ C(x > x_{\text{max}}, t) = 0 \]  

(9)

\[ K = 0 \text{ for } x > x_{\text{max}} \text{ or } x < x_{\text{min}} \]  

(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 \( \Delta x \). 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:

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

(11)

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

If the length of the region of interest is not a multiple of $\Delta x$, 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 $t_0$ to time $t_M$ as specified by the input value $\Delta t$. If the duration is not a multiple of $\Delta t$, the ending time is adjusted accordingly. Each time step $k$ has duration $\Delta t$ and covers the period from $t_{k-1} = t_0 + (k-1)\Delta t$ to $t_k = t_0 + (k)\Delta t$.

The input time step $\Delta t$ 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:

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

(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,

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

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

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

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,
\[ \gamma_{i+1/2} = \frac{A K_{i+1/2}}{\Delta x} \]  

(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,

\[ q_i^f = \sum_j x(j, i) q_j^F , \]  

(19)

where

\[ x(j, i) = \begin{cases} 
1 & \text{if feedpoint } j \text{ flows into cell } i \\
0 & \text{otherwise}
\end{cases} \]

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:

\[ q_i = q_i^f + q_{i-1/2} . \]  

(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:

\[ R_{\max} = \frac{d Q / A}{3 \times 10^{-7} \text{m}^2/\text{sec}} , \]  

(21)

where \( d \) is the uniform diameter and \( A \) is the uniform cross-sectional area. This
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, \( \Delta x \), the finite-difference version of equation (7) may be written in terms of mass transfer (kg/sec) as follows:

\[
(C_i, k/\Delta t)(\Delta x) = (C_{i-1}, k) (q_{i-1/2}) + C_{i-1, k-1} (q_{i-1/2} - q_{i+1/2}) + \frac{1}{2} (C_{i-1, k-1} - C_{i+1, k-1}) (\gamma_{i-1/2}) - \frac{1}{2} (C_{i, k-1} - C_{i+1, k-1}) (\gamma_{i+1/2})
\]

where \( C_{i, k} \) is the average concentration of electrolyte in the fluid flowing from the feed points into cell \( i \) during time step \( k \), and \( \gamma_{i \pm 1/2} \) 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:

\[
(C_i, k/\Delta t)(\Delta x) = \sum \frac{1}{2} (C_{i-1, k} - C_{i+1, k-1}) (\gamma_{i \pm 1/2}) + C_{i-1, k-1} (q_{i-1/2} + \gamma_{i-1/2}) + C_{i, k-1} (-q_{i+1/2} - \gamma_{i-1/2} - \gamma_{i+1/2}) + C_{i+1, k-1} (\gamma_{i+1/2})
\]

Then an \( NxN \) banded matrix, \( T \), can be developed from the coefficients above to compute mass transfers. If \( C_k \) is the \( Nx1 \) matrix of concentrations at time \( k \), and \( F_k \) is the \( Nx1 \) matrix of mass flows from the feed points, \( F_{i, k} = C_{i, k} (q_{i}) \), the calculation can be stated as:

\[
(TC_{k-1} + F_k)(\Delta t/\Delta x) = C_k
\]

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, \( C_{i,k} \), is computed as follows:

\[
C_{i,k} = \sum_j x(j, i) C_j^F (q_j^F / q_i) (r/\Delta t),
\]

where

\[
x(j, i) = \begin{cases} 
1 & \text{if feedpoint } j \text{ flows into cell } i \\
0 & \text{otherwise}
\end{cases}
\]

and

\[
\tau = \begin{cases} 
0 & \text{if } t_k \leq t_j^F \\
t_k - t_j^F & \text{if } t_j^F < t_k < t_j^F + \Delta t \\
\Delta t & \text{if } t_j^F + \Delta t \leq t_k
\end{cases}
\]

where \( t_j^F \) is the time feed point \( j \) begins flowing. It is assumed that once a feed point begins flowing with concentration \( C_j^F \) at time \( t_j^F \) 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:

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

\[
M_{in,k} = \sum_{m=i}^{k} \sum_{i=1}^{N} \Delta t(C_{i,m})(q_i^F) \tag{27}
\]

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

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:

\[
| M_{s,k} - (M_{in,k} - M_{out,k}) | > 0.001 M_{in,k} \tag{29}
\]
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.

$$M_{in, M} = \sum_{k=1}^{M} \sum_{i=1}^{N} \Delta t(C_{i, k}^f)$$

$$M_F = \sum_j (t_{M} - t_j^F)q_j^F C_j^F$$

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

$$| M_{in, M} - M_F | > 0.0001 M_F .$$
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 6.1 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 3.1 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 coefficients for converting electrolyte concentra-
tion in kg/m³ 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 aver-
age 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, includ-
ing 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 concentra-
tion 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.

<table>
<thead>
<tr>
<th>UNIT NUMBER</th>
<th>INPUT/OUTPUT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>INPUT</td>
<td>Problem description</td>
</tr>
<tr>
<td>6</td>
<td>OUTPUT</td>
<td>Messages</td>
</tr>
<tr>
<td>7</td>
<td>OUTPUT</td>
<td>Computed conductivity data</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>LINE</th>
<th>NAME</th>
<th>UNITS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>XTOP</td>
<td>m</td>
<td>Top of study area, surface is zero and positions increase downward, adjusted</td>
</tr>
<tr>
<td></td>
<td>XBOT</td>
<td>m</td>
<td>Bottom of study area</td>
</tr>
<tr>
<td></td>
<td>DELX</td>
<td>m</td>
<td>Cell length</td>
</tr>
<tr>
<td></td>
<td>DIAM</td>
<td>cm</td>
<td>Borehole diameter (uniform)</td>
</tr>
<tr>
<td>2</td>
<td>QEXTI(I)</td>
<td>m³/sec</td>
<td>Flow in at bottom</td>
</tr>
<tr>
<td></td>
<td>QEXTO(ILIM)</td>
<td>m³/sec</td>
<td>Flow out at top, adjusted if necessary to fit total inflow</td>
</tr>
<tr>
<td>3</td>
<td>IFLIM</td>
<td>none</td>
<td>Number of feed points</td>
</tr>
<tr>
<td>3.1</td>
<td>XIN(I)</td>
<td>m</td>
<td>Position of feed point</td>
</tr>
<tr>
<td>I=1,</td>
<td>QIN(I)</td>
<td>m³/sec</td>
<td>Feed point flow rate</td>
</tr>
<tr>
<td></td>
<td>CIN(I)</td>
<td>kg/m³</td>
<td>Feed point fluid concentration</td>
</tr>
<tr>
<td></td>
<td>TR(I)</td>
<td>hr</td>
<td>Feed point flow start time</td>
</tr>
<tr>
<td>4</td>
<td>TSTART</td>
<td>hr</td>
<td>Problem start time</td>
</tr>
<tr>
<td></td>
<td>TEND</td>
<td>hr</td>
<td>Problem end time, may be recomputed to fit TSTART and DELT</td>
</tr>
<tr>
<td></td>
<td>DELT</td>
<td>min</td>
<td>Time step, may be reduced to the maximum for flow rates and geometry</td>
</tr>
<tr>
<td></td>
<td>ILPT</td>
<td>none</td>
<td>Number of print times</td>
</tr>
<tr>
<td>4.1</td>
<td>PT(I)</td>
<td>hr</td>
<td>Time at which concentration curve is to be printed, actual curve is at next</td>
</tr>
<tr>
<td>I=1,</td>
<td></td>
<td></td>
<td>discrete time</td>
</tr>
<tr>
<td></td>
<td>ILPT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ITYPDK</td>
<td>none</td>
<td>Type code for dispersion coefficient</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 means DK is constant over all cells</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 means DK is the value for the mean velocity,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( \bar{q} = (\min(q_i) + \max(q_i))/2, K_i = DK(q_i/\bar{q}) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 means DK is the value for mean vel. squared,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>( q^2 = (\min(q_i^2) + \max(q_i^2))/2, K_i = DK(q_i^2/q^2) )</td>
</tr>
<tr>
<td></td>
<td>DK</td>
<td>m²/sec</td>
<td>Dispersion coefficient</td>
</tr>
</tbody>
</table>
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):

\[ C(x,t) = qC_0(4\pi D)^{-1/2} \int_0^t (t-r)^{-1/2} \exp\left[-x^2/4D(t-r)\right] \, dr \]

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 \( qC_0 = 1 \) 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:

<table>
<thead>
<tr>
<th>LINE NO.</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-9.975, 9.975, .05, 112.838</td>
</tr>
<tr>
<td>2</td>
<td>0, 1.e-5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3.1</td>
<td>0, 1.e-5, 1.e5, 0</td>
</tr>
<tr>
<td>4</td>
<td>0,4.5e-3,5.e-5,4</td>
</tr>
<tr>
<td>4.1</td>
<td>2.778e-4</td>
</tr>
<tr>
<td>4.2</td>
<td>1.111e-3</td>
</tr>
<tr>
<td>4.3</td>
<td>2.500e-3</td>
</tr>
<tr>
<td>4.4</td>
<td>4.444e-3</td>
</tr>
<tr>
<td>5</td>
<td>1,0.25</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0,1,0</td>
</tr>
</tbody>
</table>
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):

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

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:

<table>
<thead>
<tr>
<th>LINE NO.</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-149.875, 0.125, .25, 112.838</td>
</tr>
<tr>
<td>2</td>
<td>0, 1.923e-5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3.1</td>
<td>0, 1.923e-5, 1, 0</td>
</tr>
<tr>
<td>4</td>
<td>0.1444.8,15,2</td>
</tr>
<tr>
<td>4.1</td>
<td>722.4</td>
</tr>
<tr>
<td>4.2</td>
<td>1444.8</td>
</tr>
<tr>
<td>5</td>
<td>1,1.923e-5</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>6.1</td>
<td>0,1</td>
</tr>
<tr>
<td>7</td>
<td>0,1,0</td>
</tr>
</tbody>
</table>
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:

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

<table>
<thead>
<tr>
<th>LINE NO.</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>750, 1490, 5, 14</td>
</tr>
<tr>
<td>2</td>
<td>0, 1.e-5</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>3.1</td>
<td>1440, 0.820e-6, 0.530, 18</td>
</tr>
<tr>
<td>3.2</td>
<td>1300, 0.440e-6, 0.520, 18</td>
</tr>
<tr>
<td>3.3</td>
<td>1215, 0.480e-6, 0.520, 18</td>
</tr>
<tr>
<td>3.4</td>
<td>1200, 0.200e-6, 0.550, 18</td>
</tr>
<tr>
<td>3.5</td>
<td>1188, 0.590e-6, 0.470, 18</td>
</tr>
<tr>
<td>3.6</td>
<td>1085, 0.170e-6, 0.530, 18</td>
</tr>
<tr>
<td>3.7</td>
<td>1048, 0.510e-6, 0.530, 18</td>
</tr>
<tr>
<td>3.8</td>
<td>918, 0.210e-6, 13.000, 9</td>
</tr>
<tr>
<td>3.9</td>
<td>843, 6.900e-6, 1.200, -12</td>
</tr>
<tr>
<td>4</td>
<td>-13,60,30,5</td>
</tr>
<tr>
<td>4.1</td>
<td>13.05</td>
</tr>
<tr>
<td>4.2</td>
<td>27.20</td>
</tr>
<tr>
<td>4.3</td>
<td>31.63</td>
</tr>
<tr>
<td>4.4</td>
<td>38.68</td>
</tr>
<tr>
<td>4.5</td>
<td>57.40</td>
</tr>
<tr>
<td>5</td>
<td>1.0.5e-3</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>73, 1870, -40</td>
</tr>
</tbody>
</table>
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


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


Appendix A--FORTRAN Source Code
borehole flow model
june 1987
lawrence berkeley laboratory, earth sciences division

pt is print times
dimension pt(25)

common /segs/ contains segment data used during time step
common s are used to speed up processing and avoid
parameter passing during time steps
cxt is concentration array at two times (i-1, i) (kg/m**3)
cfrac is concentration in fracture flow into segment
(kg/m**3)
gexto is flow out of the system from each segment (m**3/sec)
gexti is flow into the system at each segment (m**3/sec)
gfrac is fracture flow into each segment (m**3/sec)
qnxt is flow from one segment to the next (up) (m**3/sec)
qtot is total flow through a segment
xs is segment midpoint
dks is harmonic mean dispersion coefficient between
segments i and i+1

common /segs/ cxt(1000,2),cfrac(1000),
gexto(1000),gexti(1000),gfrac(1000),qnxt(1000),
qtot(1000),xs(1000),dks(1000)

common /fracs/ contains fracture data used during time step
qin is flow rate at each fracture (m**3/sec)
cin is concentration in flow at each fracture (kg/m**3)
xin is position of fracture (m)
ifseg is the segment in which the fracture is found
t0 is the starting time for solute flow in the fracture (sec)

common /fracs/ qin(100),cin(100),xin(100),ifseg(100),t0(100)

common /step/ is used for additional time step data
delt is time step (sec, input)
ilim is number of segments (computed)
vol is segment volume (m**3, computed)
gmout is grand total mass out of the system (kg)
gmtna is actual grand total mass into the system (kg)
t is time at end of time step (sec, computed)
iout is general output unit number

ic is time at beginning of time step (sec, computed)
ilim is number of fractures (computed)
iprt is concentration output unit number
itog is toggle between two sets of C(x,t)
tm0 is initial mass in system
c

common /step/ delt,ilim,imax,vol,gtmout,gtmina,
    t,iout,tp,iflim,iprt,itog,tm0

C

C*** offset 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/ offset,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*** imx is maximum number of segments
C*** imax 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(l),cin(l),xin(l),ifseg(l),t0(l),ifmax,iflim,
    xtop,xbot,delx,iin,iout)
call flows(qexto(l),qexti(l),qfrac(l),qnext(l),qtot(l),
    qin(l),ifseg(l),ilim,iflim,iout)
call tandr(qtot(l),ilim,diam,area,vol,dtmax,rnmax,iout)
call rdtime(tstart,tend,delt,pt(l),ilpt,impt,itstep,itmax,
    dtmax,iin,iout)
call chkt(t0(l),ilim,tstart,tend,iout)
call rdk(dks(l),qtot(l),ilim,area,delx,iin,iout)
call rdc0(cx(l,1),ilim,ilim,xtop,xbot,delx,vol,tm0,iin,iout)
call rdcorr(offset,coefa,coefb,iin)
iprod=itstep*ilim
write(iout,910)iprod

*** loop over each time step
     do 1000 i=1,itstep

*** compute interval time limits
     t=tstart+i*delt
     tp=t-delt

*** compute fracture inflows
     call ctfrac

*** compute mass transfer
     call tstep

*** check for output requests
     do 600 j=1,ilpt
         if (t.ge.pt(j)) then
             call cppt
             pt(j)=2.*tend
         endif
     600 continue

*** end of loop over time steps
     continue

*** check for final print

     needp=0
     do 1500 j=1,ilpt
         if (pt(j).lt.(2.*tend)) needp=1
     1500 continue
     if (needp.ne.0) call cppt

*** compare discreet and continuous mass calculations

     call gtin(tend,cin(1),qin(1),t0(1),iflim,gtmina,iout)

*** end of problem

     write(iout,900)
stop

c 900 format(/' *** PROBLEM FINISHED ***')
910 format(/' Problem size parameter (segments*time steps) is',i15)
end

c*** subroutine rdhole

c*** read borehole parameters

c*** this subroutine is executed only once before time steps

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, converted to m)
c*** area is cross-sectional area of hole (m**2)
c*** vol is segment volume (m**3)
c*** nlim is number of segment over analysis region

c*** input stream--one free format record

c*** xtop, xbot, delx, diam

c subroutine rdhole(xtop,xbot,delx,diam,area,vol,nlim,imax,
                 xs,iin,iout)

c dimension xs(imax)
data 'pi/3.141592654/

c*** interactive prompt disabled

c write(iout,900)
read(iin,*)xtop,xbot,delx,diam

c*** convert diameter in cm to m

c diam=diam/100.

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

- 34 -
endif
ilim=xint
if (ilim.gt.imax) then
  write(iout,990)imax
  stop
endif
xtopi=xtop
xtop=xbot-(ilim*delx)
deltop=abs(xtop-xtopi)
write(iout,955)xbot,xtopo

*** if delta-x does not fit over the interval within 1% of
*** a delta-x, then make a note
if (deltop.gt.(0.01*delx)) then
  write(iout,960)ilim,delx
  write(iout,970)deltop,xtop
else
  write(iout,960)ilim,delx
endif

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 compute segment midpoints
*** note--sign reversed for graphing

do 100 i=1,ilim
  xs(i)=xbot+(0.5-i)*delx
100 continue
return
c 900 format(' Enter top and bottom of region (m), delta-x (m), and'  
  ' diameter (cm)')
  940 format(' Top of region is below bottom of region--aborting')
  950 format(' delta-x is greater than region size--aborting')
  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')

- 35 -
subroutine qexts
read external (non-fracture) flows
input stream (free format)
flow in at bottom and out at top (m**3/sec)
number of additional inflows (inactive)
for each inflow: position (m) and rate (m**3/sec)
number of additional outflows (inactive)
for each outflow: position (m) and rate (m**3/sec)
this subroutine is executed only once before time steps

subroutine qexts(qexto,qexti,xtop,xbot,delx,ilim,imax,iin,iout)
dimension qexto(imax),qexti(imax)
do 10 i=1,ilim
  qexto(i)=0.
  qexti(i)=0.
10 continue
top and bottom flows
interactive prompt disabled
write(iout,900)
read(iin,*)qexti(1),qexto(ilim)
additional non-fracture inflows (disabled)
write(iout,920)
read(iin,*)n
if (n.gt.0) then
  write(iout,930)
do 100 i=1,n
    read(iin,*)xin,qin
    if ((xin.gt.xbot .or. xin.lt.xtop)) then
      write(iout,935)xin
      goto 100
    endif
    ixin=iconvx(xtop,xbot,delx,xin)
    qexti(ixin)=qexti(ixin)+qin
100 continue
endif
additional non-fracture outflows (disabled)
write(iout,950)
read(iin,*)n
if (n.gt.0) then
write(iout,960)
do 200 i=1,n
read(iin,*)xout,qout
if ((xout.lt.xbot .or. xout.gt.xtop)) then
  write(iout,935)xout
goto 200
endif
ixout=iconvx(xtop,xbot,delx,xout)
qexto(ixout)=qexto(ixout)+qout
continue
endif

*** write summary of non-fracture external flows (disabled) ***
write(iout,980)
do 300 i=1,ilim
if ((qexti(i).ne.0.) .or. (qexto(i).ne.0.)) then
  xb=xbot+(l-i)*delx
  xt=xb-delx
  write(iout,990)i,xt,xb,qext1(i),qexto(1)
endif
continue
return

format(j' Enter flow into bottom and out of top of region',
  (m**3j/sec))
920 format(Enter number of additional non-fracture external',
  ' inflows')
930 format(Enter position (m) and rate (m**3/sec) of each inflow')
935 format(*** NOTE *** flow position outside of region ',f10.4,
  -- ignored')
950 format(Enter number of additional non-fracture external',
  ' outflows')
960 format(Enter position (m) and rate (m**3/sec) of each outflow')
980 format(/Summary of external non-fracture flows'/
  Segment Top Bottom In (m**3/sec)',
  Out (m**3/sec'))
990 format(i6,2fll.4,2e15.5)
end

*** function iconvx
*** convert x position to segment index
*** segment 1 begins at xbot
*** segment ilim ends at xtop
*** this function is used outside of time steps
function iconvx(xtop,xbot,delx,x)
idxup=(xbot-x)/delx+1. 
if (x.eq.xtop) idxup=idxup-1
iconvx=idxup
return
end

subroutine rdfrac
*** read flow from fractures
*** input stream -- free format
*** number of fractures
*** one record for each fracture --
*** position (m), flow rate (m**3/sec), concentration of
*** solute (kg/m**3), and solute flow start time (h)
*** this subroutine is executed once before time steps

subroutine rdfrac(qin,cin,xin,ifseg,t0,ifmax,iflim,
    xtop,xbot,delx,iin,iout)

dimension qin(ifmax),cin(ifmax),xin(ifmax),ifseg(ifmax),t0(ifmax)

*** interactive prompt disabled

read fracture data

write(iout,900)
read(iin,*)iflim
if (iflim.gt.ifmax) then
    write(iout,910)ifmax
    stop
endif

j=iflim

*** interactive prompt disabled

write(iout,920)
do 100 i=1,j
    read(iin,*)xin(i),qin(i),cin(i),t0(i)
100

*** locate fracture on segments
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))

*** convert t0 to seconds

    t0(i)=t0(i)*3600.

  - 38 -
100 continue

write fracture summary (disabled)

write(iout,940)
do 200 i=1,ilim
write(iout,950)xin(i),qin(i),cin(i),t0(i),ifseg(i)
200 continue

return

format(/' Enter number of fractures flowing into borehole')
900 format(' Maximum number of fractures is ',i4,' -- aborting')
920 format(' Enter position (m), flow rate (m**3/sec),',/
930 format(' solute concentration (kg/m**3), and starting'/
940 format(' *** NOTE *** fracture position outside of region ',f10.4,
950 format(' -- ignored')

format(/' Summary of fracture input'/
   format(/' Position',8x,'Rate',6x,'Concentration',
   format(/' Start time Seg.')
end

subroutine flows
*** calculate flows through each segment
*** this subroutine is executed once before the time steps

subroutine flows(qexto,qexti,qfrac,qnxt,qtot,
   qin,ifseg,ilim,iflim,iout)

dimension qexto(ilim),qexti(ilim),qfrac(ilim),qnxt(ilim),
   qtot(ilim)
dimension qin(iflim),ifseg(iflim)

*** initialize segment flows

do 100 i=1,ilim
   qfrac(i)=0.
100 continue

*** compute flow into segments from fractures
*** it is possible to have more than one fracture in a single
*** segment

do 200 i=1,ilim
   qfrac(ifseg(i))=qfrac(ifseg(i))+qin(i)
200 continue

*** compute flow up to next segment and total flow
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)
    endif

*** adjust input value of outflow at top to match total flow
    if (i.eq.iliim) 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
return

900 format(/' *** NOTE *** Outflow at top changed by ',el5.5, .
         . to ',el5.5)
end

subroutine tandr
compute maximum time step and Reynold's number
this subroutine is executed once before the time steps

subroutine tandr(qtot,ilim,diam,area,vol,dtmax,rnmax,iout)
dimension qtot(ilim)

qtmax=abs(qtot(l))
do 100 i=2,ilim
    if (abs(qtot(i)).gt/qtmax) qtmax=abs(qtot(i))
100 continue

datax=0.5*vol/qtmax
vmax=qtmax/area
rnmax=vmax*diam/0.3e-6
dtm=dtmax/60.
write(iout,900)dtm,vmax,rnmax
return

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

subroutine rdtime
 **** read time parameters
 **** input stream--
 ****   start time (hrs), end time (hrs), delta time (min),
 ****   number of prints
 ****   then, print times (hrs)
 **** note: all times are converted to seconds
 **** this subroutine is executed only once before time steps

subroutine rdtime(tstart,tend,delt,pt,ilpt,impt,itstep,itmax, 
  dtmax,iin,iout)

dimension pt(impt)

**** read in time parameters
**** interactive prompt disabled

write(iout,900)
read(iin,*)tstart,tend,delt,ilpt 
tstart=tstart*3600.
tend=tend*3600.
delt=delt*60.

if (tend.lt.tstart) then
  write(iout,910) stop
endif
if (ilpt.gt.impt) then
  write(iout,915)impt
endif

**** read in print-time parameters
**** interactive prompt disabled

write(iout,917)
read(iin,*)(pt(i),i=1,ilpt)
do 100 i=1,ilpt
   pt(i)=pt(i)*3600.
100 continue
c*** check for maximum delta-t
if (delt.gt.dtmax) then
delt=dtmax
dtm=delt/60.
    write(iout,920)dtm
endif

c*** compute number of time steps
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*** adjust ending time if needed
tendx=tend
tend=tstart+itstep*delt
diff=abs(tend-tendx)
tsh=tstart/3600.
teh=tend/3600.
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 900 format(/' Enter start (h), end (h), delta (m) and no. of prints')
c 915 format(/' Maximum number of print times is ',i4,' -- aborting')
c 917 format(/' Enter print times (h)')
c 910 format(/' End time before start time -- aborting')
c 920 format(/' *** NOTE *** time step changed to maximum of ',
         . f15.5,' (min)')
c 925 format(/' Maximum number of time steps is ',i6,' -- aborting')
c 930 format(/' Beginning and ending times are (hrs):',2f15.5)
c 940 format(/' *** NOTE *** Ending time changed by ',f15.5,' to ',
         . f15.5,' (hrs)')
c 950 format(/' Using ',i6,' time steps of ',f15.5,' (min)')
end
subroutine ctfrac
prepare fracture concentrations for time step
this subroutine is executed during each time step,
and it loops over each fracture

subroutine ctfrac

common /segs/ cxt(1000,2),cfrac(1000),
gexto(1000),gexti(1000),qfrac(1000),qnxt(1000),
gtot(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

reset values first

do 10 i=1,iflim
   cfrac(ifseg(i))=0.
10 continue

compute mass inflow during time step
more than one fracture may be in a segment

do 100 i=1,iflim
   if (t.le.t0(i)) goto 100
   is=ifseg(i)
   if (tp.ge.t0(i)) then

inflow of mass flow during entire time step (kg/sec)
cfrac(is) = cfrac(is) + cin(i)*qin(i)/qfrac(is)
else
inflow of mass during part of time step (kg/sec)
cfrac(is) = cfrac(is) +
   cin(i)*qin(i)*((t-t0(i))/delt)/qfrac(is)
endif
100 continue

return
end

subroutine rdk
read dispersion parameter
this subroutine is only executed once before time steps
input stream--(one record)
type code, parameter
at the end of this routine, dks has units m**3/sec

subroutine rdk(dks,qtot,ilim,area,delx,iin,iout)

**frac** holds the multipliers for schemes 2 and 3

dimension dks(ilim),qtot(ilim)
dimension frac(1000)

**read dispersion parameter**
**interactive prompt disabled**

write(iout,900)
read(iin,*)itypdk,dk

if itypdk=1, value is constant over all segments
if itypdk=2, value is for mean velocity
if itypdk=3, value is for mean velocity squared

\[ dksi = dk * area / delx \]

if (itypdk.eq.1) then
  do 100 i=1,ilim
   dks(i)=dksi
  continue
else
  find first nonzero, minimum, maximum and mean velocities

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)
  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
  continue
endif
qavg=(qmin+qmax)/2.
if (qavg.eq.0.) then
  write(iout,910)
  stop

endif

   do 250 i=l,ilim
      if (qtot(i).ne.0.) then
         qtoti=qtot(i)
         if (itypdk.eq.3) qtoti=qtoti*qtoti
      else
         qtoti=qfnz
      endif
      frac(i)=qtoti/qavg
      continue

 250 

compute harmonic mean of scaled dispersion parameter

j=ilim-1
   do 300 i=1,j
      qx=2./( (1./frac(i)) + (1./frac(i+1)) )
      dks(i)=dksi*qx
      continue
300 

endif
return

900 format(/' Enter dispersion parameter (m**2/sec)'
910 format(/' *** NOTE *** average flow rate is zero -- aborting'
end

subroutine tstep
 calculate concentrations at next time step
 this is the main mass transport time step routine
 executed for each time step, and loops over each segment

subroutine tstep

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

ml -- initial mass in segment
m2 -- final mass in segment
m1 -- rate change in mass due to advection from left (+ = in)
m2 -- rate change in mass due to advection to right (- = out)
m1 -- rate change in mass due to diffusion with left (+ = in)
m2 -- rate change in mass due to diffusion with right (- = out)
m2 -- rate change in mass due to fracture inflow (+ = in)
dme -- rate change in mass due to external outflow (- = out)
calculate mass changes
gtm is total mass in system at end of time step
gtm=0.
generalized outflow disabled
dme=0.

loop over each segment

ito gx=3-ito
do 100 i=1,ilim
    ipl=i+l
    cxt i=cxt(i,ito)
    tml=cxti*vol
    if (cfrac(i).ne.0.) then
        dmf=cfrac(i)*qfrac(i)
    else
        dmf=0.
    endif

generalized outflow disabled
outflow considered only at top segment
dme--cxti*qexto(i)

if (i.gt.l) then
    iml=i-l
independent calculation of mass transfer with left side
disabled--using complement of right side values from below

if (qnxt(iml).gt.0.) then
    dmal=cxt(iml,ito)*qnxt(iml)
else
    dmal=cxti*qnxt(iml)
endif
dmdl=(cxt(iml,ito)-cxti)*dks(iml)
dmar=dma

dmdr
else
    dmal=0.
dmdl=0.
endif
if (i.lt.ilim) then
note: full upstream weighting is used. Without upstream weighting, these equations would use \((c_{xti} + c_{xt(ipl,itog)})/2.\) in place of \(c_{xti}\) or \(c_{xt(ipl,itog)}\)—that is, the concentration at the interface would be the average instead of the upstream value.

```fortran
if (qnxt(i).gt.0.) then
dmar = -cxti*qnxt(i)
else
  dmar = -cxt(ipl,itog)*qnxt(i)
endif

dmr = (cxt(ipl,itog)-cxti)*dks(i)
else
dmar = 0.
dmr = 0.
```

outflow considered only at top

```fortran
dme = -cxti*qexto(i)
endif
```

check for segment mass limitations

```fortran
if more mass is being moved out of a segment than is in the segment, write an error message and stop. This problem can be fixed by reducing the time step or increasing the segment length.

```fortran
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
```

compute new mass in segment

```fortran
tm2 = tml + (dms + dme + dmal + dmdl + dmar + dmdr)*delt
cxt(i,itolgx) = tm2/vol
```

check for conservation of mass

```fortran
gtmin is total inflow from fractures
gtout is sum of external outflows
```
c
  gtmout=gtmout-(dme*delt)
gtmina=gtmina+(dmf*delt)
gtm=gtm+tm2
100 continue

c*** end of loop over segments

c*** emass is expected total mass in system (total in - total out)
c*** gtm is actual total mass in system

c*** if these numbers differ by more than 0.1% of total input mass, 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*** toggle row pointer

c itog=itogx
return

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

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

- 48 -
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),qnxr(1000),
   qtot(1000),xs(1000),dks(1000)
common /step/ delta,ilim,imax,vol,gmtout,gtmina,
   t,iout,tp,iflim,iprt,ito,tm0
common /corr/ ofset,coefa,coefb

c*** output time in hours and concentrations with ascending
** c*** values (note negative increment in loop)

th=t/3600.
write(iprt,900)th
isig=1
if (isig.eq.1) then
   do 100 i=1,ilim
      ci=cxt(i,ito)
      sigma(i)=(ofset + ci*coefa + ci*ci*coefb)
   continue
   write(iprt,910) (xs(i),sigma(i),i=ilim,1,-1)
else
   write(iprt,910) (xs(i),cxt(i,ito),i=ilim,1,-1)
endif
write(iout,920)th
return

900 format( '' ,''):f15.3,'''
910 format(3(fl0.3,el5.5)
920 format(/' Printing conductivity at time ','el5.5,' (hrs)')
end

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*** compute continuous mass inflow
gtmin=0.
do 100 i=1,iflim
dt = tend - t0(i)
    if (dt.gt.0) gtmin = gtmin + (dt*qin(i)*cin(i))
100 continue

*** compute difference between continuous and discreet mass inflows

c   aerr = gtmina - gtmin
   rerr = (100.*aerr)/gtmin

*** if discretization error in total mass into system is greater than 0.01%, write an information message

c   if (abs(rerr).gt.0.01) write(iout,900)gtmin,gtmina,aerr,rerr
   return

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

*** subroutine chkt

*** check fracture flow times to make sure they fit into problem

subroutine chkt(t0,iflim,tstart,tend,iout)
dimension t0(iflim)

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

900 format(/' *** NOTE *** fracture ',i3,' t0 at ',f15.5,' (hrs)'
   ,/ is outside of problem time limits')
end

*** subroutine rdc0

*** read initial concentrations

subroutine rdc0(cxt,imax,ilim,xtop,xbot,delx,vol,tm0,iin,iout)
**code**

```c

dimension cxt(imax,2)

do 100 i=l,ilim
    cxt(i,l)=0.
100 continue

tm0=0.

read(iin,*)icOn
if (icOn .gt. 0) then
    do 500 i-1,icOn
        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,l).eq.0.) then
                cxt(iseg,l)=c0
                tm0=tm0 + vol*c0
            else
                write(iout,910)x
            endif
        endif
    500 continue
endif

return

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)

*** subroutine rddcorr
*** read correlation between concentration and conductivity:
*** ofset is the background conductivity in S/m
*** coefa is the linear term for converting concentrations
*** in kg/m**3 to conductivity in S/m (at 20 C)
*** coefb is the second degree term for converting
*** concentrations to conductivity (at 20 C)

subroutine rddcorr(ofset,coefa,coefb,iin)
    read(iin,*)ofset,coefa,coefb
    return
end
```

---

**description**

The provided code snippet is a section of a larger program that deals with numerical computations involving concentration and conductivity. The code begins with initializing a 2D array `cxt` for storing concentration data. It then reads initial concentration values from an input file `iin` and processes them based on certain conditions related to concentration levels within specified ranges. The program also features a subroutine `rdcorr` which reads and processes correlation coefficients for converting concentrations to conductivities. The code includes error handling and output formatting for reporting initial concentration positions and multiple initial concentration values for a single segment. The comments within the code provide explanations for each section, including variable roles and the conversion process between concentrations and conductivities. The overall purpose seems to be the analysis or simulation of a system involving concentration and conductivity calculations.