VHBORE: A Code to Compute Borehole Fluid Conductivity Profiles With Pressure Changes in the Borehole

F.V. Hale and C.F. Tsang
Earth Sciences Division
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

June 1994

This work was carried out under U.S. Department of Energy Contract No. DE-AC03-76SF00098 for the Director, Office of Civilian Radioactive Waste Management, Office of External Relations, administered by the Nevada Operations Office in cooperation with the Swiss National Cooperative for Disposal of Radioactive Waste (Nagra).
DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
1. INTRODUCTION

This report describes the code VHBORE which can be used to model fluid electric conductivity profiles in a borehole intersecting fractured rock under conditions of changing pressure in the well bore. Pressure changes may be due to water level variations caused by pumping or fluid density effects as formation fluid is drawn into the borehole. Previous reports describe the method of estimating the hydrologic behavior of fractured rock using a time series of electric conductivity logs (Tsang et al., 1989), and an earlier code, BORE, to generate electric conductivity logs under constant pressure and flow rate conditions (Hale and Tsang, 1988).

The earlier model, BORE, assumed a constant flow rate, $q_i$, for each inflow into the well bore. In the present code the user supplies the location, constant pressure, $h_i$, transmissivity, $T_i$, and storativity, $S_i$, for each fracture, as well as the initial water level in the well, $h_w(0)$. In addition, the input data contains changes in the water level at later times, $\Delta h_w(t)$, typically caused by turning a pump on or off. The variable density calculation also requires input of the density of each of the inflow fluids, $\rho_i$, and the initial uniform density of the well bore fluid, $\rho_w(0)$. These parameters are used to compute the flow rate for each inflow point at each time step.

The numerical method of Jacob and Lohman (1952) is used to compute the flow rate into or out of the fractures based on the changes in pressure in the wellbore. A dimensionless function relates flow rate as a function of time in response to an imposed pressure change. The principle of superposition is used to determine the net flow rate from a time series of pressure changes. Additional reading on the relationship between drawdown and flow rate can be found in Earlougher (1977), particularly his Section 4.6, "Constant-Pressure Flow Testing."

The primary difference between variable and constant density applications of the code is that with variable densities the pressure profile is computed at each time step, and when
density variations in the well bore result in a significant pressure change (i.e. equal to a
column of water of height equal to one half of the cell height, $pg\Delta x/2$), this is also added
to the flow rate calculations. The wellbore is assumed to be vertical for purposes of
pressure calculations; however no gravity-driven flow effects are considered in this code.

The electrolyte concentration and the density are not related in the present code.

Therefore the code can be considered to model the transport of two independent
substances using the same flow field. Only the electrolyte is affected by diffusion in the
current code, and only the density affects the pressure profile. The decision to separate
the density calculation from the electrolyte concentration was made to permit modeling of
conditions where the density effects are due to something other than the electrolyte
solutions, such as drilling mud. Full radial mixing is assumed in the cells, and fingering
effects are not considered.

2. NUMERICAL AND ANALYTICAL SOLUTIONS

Governing Equation for Borehole Flow with Sources

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

$$\frac{\partial}{\partial x} \left( K \frac{\partial C}{\partial x} \right) - \frac{\partial}{\partial x} (C V) + S = \frac{\partial C}{\partial t}$$

(1)

where

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

This partial differential equation is solved numerically using the finite difference method
(FDM). The following initial and boundary conditions are also specified:
\[ C(x, 0) = C_0(x) \]  
\[ C(x > x_{\text{max}}, t) = 0 \]  
\[ K = 0 \quad \text{for} \quad x > x_{\text{max}} \quad \text{or} \quad x < x_{\text{min}} \]  

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.

This is the same governing equation used by the constant flow code, BORE. The differences in VHBORE are seen in the source, velocity and, if velocity dependent, dispersion terms. The source term, \( S_i \), is no longer constant, but include the effects of two-way flow between the well bore and the formation as the pressure changes in the borehole. The velocity term, \( V_i \), varies in response to the pressure history of the well, and thus, if the dispersion coefficient, \( K_i \), is velocity dependent, it will also vary continuously with time.

**Discretization in Space**

In the borehole, 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; 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 $x_{\text{max}} - (i - 1/2) \Delta x$, 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). Note that because all cells are assumed to have the same geometry, flow rates are directly proportional to linear velocities.

Each inflow is given a specific location in the input file, and the inflows are then assigned to specific cells. If multiple inflows are assigned to the same cell, their flow rates and mass transfer rates are summed to produce a single source term for the cell. The single source for the cell is assumed to be located at the midpoint of the cell.

The BORE code modeled a fracture as an infinite source of fluid with zero concentration until a specified time, after which time the infinite source provides fluid with a constant concentration. In the codes described here, a set of 50 “buckets” of variable volume and concentration have been set between the borehole and this infinite fluid source of fixed concentration. These buckets represent no particular geometry, but rather just a volume of water with a particular concentration.

As fluid flows from the borehole to the fracture, the buckets are filled, one at each time step. In the code, this is accomplished by storing a volume and concentration in corresponding positions in the matrices $V_{\text{FRAC}}$ and $CF_2$, which are indexed by fracture and bucket. The volume and concentration of the fluid transferred to the fracture depend on the time step, flow rate, and local borehole fluid concentration. If fluid has been flowing into the fracture for some time, and all of the buckets are full (i.e., all elements of the matrices $V_{\text{FRAC}}$ and $CF_2$ for the fracture are in use), the oldest ten volumes are combined (i.e., dumped into one bucket, providing nine more empty buckets) by summing the volumes and computing a new concentration based on the total mass.

As fluid flows from the fracture zone to the borehole, the buckets are emptied in the reverse order. If all of the buckets are empty, fluid is drawn from the infinite source of
constant concentration. If a number of buckets are necessary to meet the volume requirements for a single time step, the concentrations are averaged.

The user specifies an initial volume and concentration in the first bucket. There is no longer an initial period of zero concentration, rather fluid transfer between the borehole and the fracture begins immediately. It is possible, however, to specify an initial volume of infiltrated water with a zero concentration.

The concentration of the fluid moving between the borehole and the fracture zone (in either direction) is determined by subroutine CTFRAC which is executed at each time step. Subroutine TSTEP computes the actual mass transfer from the borehole to the fracture zone.

Discretization in Time

Because the flow rates are not constant, a constant time step is not practical. Rather than having the user specify a time step, the subroutines FLOWS, DFLOWS, FLOWSA and TSTEP work together to compute a variable time step based on maximum velocity and mass transfer rates. The time step is initially set by FLOWS or DFLOWS based on flow rates (velocities) in the previous time step, if possible. Then FLOWSA modifies the time step based on the expected flow rates during the present time step. If due to dispersion effects this time step results in mass transfer inconsistencies, the time step is further reduced by subroutine TSTEP.

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, \( K_{i+1/2} \): 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, $K_0$, is used for all the cell interfaces and Equation (1) simplifies to

$$K_0 \frac{\partial^2 C}{\partial x^2} - \frac{\partial}{\partial x}(CV) + S = \frac{\partial C}{\partial t}$$

The velocity scaled methods use a somewhat arbitrary reference dispersion coefficient $K_0$ defined as the dispersion coefficient at a depth where the flow velocity is equal to the mean velocity or the mean velocity squared,

$$\bar{v}^n = \frac{\min(V_i^n) + \max(V_i^n)}{2}$$

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

$$K_i = K_0 \left( \frac{V_i^n}{\bar{v}^n} \right)$$

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 $v_i$ (since $q_i = v_iA$, where $A$ is the uniform cross-sectional area).

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

$$K_{i\pm1/2} = 1 \left( \frac{1}{K_i} + \frac{1}{k_{i \pm 1}} \right)^{-1}$$

For 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.
where \( A \) is the uniform cross-sectional area.

If the dispersion type is flow rate dependent (\text{ITYPDK} is 2 or 3), the dispersion coefficients must be computed during each time step. This is done in subroutine \text{FLOWSA} after the current flow rates and velocities have been computed by subroutines \text{FLOWS} or \text{DFLOWS}.

**Calculation of Flow Rates**

At a given time the flow rate from a fracture zone to the borehole (or from the borehole to the fracture zone) is a function of the fracture zone hydraulic parameters and the pressure history in the borehole. At each time step the flow rate is computed by a superposition of the effects of individual pressure changes. This calculation is performed in subroutine \text{FLOWS} for constant density calculations and by subroutine \text{DFLOWS} for variable density calculations.

The effect of a single pressure change, \( j \), for a single inflow, \( i \), is computed using a dimensionless function, \( q_\text{D} \), which is defined as follows:

\[
q_{i,j}(t) = 2\pi T_i \Delta h_j q_D(t_D(t))
\]

where

\[
t_D(t) = \left( \frac{T_i(t - \tau_j)}{S_i \pi^2} \right)
\]

where \( T_i \) is the transmissivity of the inflow, \( S_i \) is the storativity of the inflow, \( r \) is the well bore radius, \( \Delta h_j \) is the drawdown for pressure change \( j \), and \( \tau_j \) is the time of pressure change \( j \).

In these codes, the terms \( 2\pi T_i \) and \( T_i/S_i \pi^2 \) are calculated in subroutine \text{RDFRAC} and stored as \text{QCOEF(I)} and \text{TCOEF(I)}, respectively. The dimensionless function \( q_D \) is

\[
\gamma_{i+1/2} = \frac{A K_{i+1/2}}{\Delta x}
\]
computed using linear interpolation on a table for arguments values between $10^{-4}$ and $10^{12}$. The table is from Jacob and Lohman (1952). For dimensionless times ($t_D$) greater than $10^{12}$, Earlougher (1977) suggests using $q_D = 2/(\ln(t_D) + 0.80907)$. For values below $10^{-4}$, a somewhat arbitrary constant value of 56.9 is used (this is the value for dimensionless time $10^{-4}$ given by Jacob and Lohman). Figure 1 shows the graph of the function $q_D$.

The calculation of flow rates is more complex for variable density cases. As the fluid of different densities enters the well bore, the pressure profile in the well bore changes. The pressure is no longer just a function of the drawdown values specified in the input file, but also of the changing density profile in the well. At any given time, the pressure (in meters of water) can be calculated as:

$$p_i(t) = \frac{1}{\rho_w} \int_{x_w}^{x_i} \rho(x,t) \, dx$$

where $\rho_w$ is the density of water, $g$ is the acceleration due to gravity, and $x_w$ is the depth of the water surface.

The use of the Jacob-Lohman solution requires that at every time step following a pressure change the effect of the pressure change be computed. In order to provide for some reasonable number of pressure changes to be recorded during the run, the wellbore pressure at each inflow is discretized in time. If a pressure change at an inflow exceeds one half of a cell length, it is considered significant, recorded, and affects all following flow rate calculations. All of this is done in subroutine DFLOWS. The array PHIST stores the pressure history (pressure and time) for each inflow. Stated more concisely, the current criterion for recording a significant pressure change due to density is:

$$|p_i(t_1) - p_i(t_2)| > \frac{\Delta x}{2}$$ (11)
This condition make the assumption that the water level, \( x_w \), does not change between times \( t_1 \) and \( t_2 \).

After the flows between the fracture zones and the borehole have been determined, the flow rates in the remaining cells are computed by subroutine FLOWSA assuming a closed lower boundary and open upper boundary in the borehole, using the discretized version of Equation (1).

Note that if the input value for the initial drawdown and inflow depths and pressure heads results in pressure differences between the input values for the fractures, it is assumed that this pressure change took place at the model starting time (TSTART).

Calculation During Each Time Step

The constant mass injection of BORE has been replaced in the variable pressure codes by possible two-way mass transfer between the borehole and the fractures. At each time step the flow rate (and direction) and fluid concentration for transfer between the borehole and the fracture zones is computed. This has been described above.

The mass transfer within the borehole during time step \( k \) is due to flow to or from the feed points (the source of electrolyte), advection, and dispersion. The finite-difference version of Equation (1) may be written in terms of mass transfer (kg/sec) as follows:

\[
\frac{(\Delta C_{i,k})}{\Delta t} (A\Delta x) = C^f_{i,k} (q^f) + C_{i-1,k-1}(q_{i-1/2}) - C_{i,k-1}(q_{i+1/2}) + (C_{i-1,k-1} - C_{i,k-1})(\gamma_{i-1/2}) - (C_{i,k-1} - C_{i+1,k-1})(\gamma_{i+1/2})
\]

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 \( \gamma_{i+1/2} \) 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:

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

(13)

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 time step is reduced.

Conservation of mass is verified during each time step and at the end of the problem. Mass may flow into the system from the infinite reservoir in each fracture zone, and mass may flow out of the system at the top of the borehole section. All other boundaries are closed.

Temperature Dependence of Conductivity

All calculations are made assuming a uniform temperature of 20°C throughout the borehole. Generally temperature increases with depth below the land surface, 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):

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

(14)

where \( T_x \) is the temperature (°C) at depth \( x \). The value of \( S \) is estimated at 0.022.
Conductivity as a Function of Concentration

Assuming that all of the ions in the borehole fluid can be converted to NaCl equivalents, the conductivity and concentration data in Shedlovsky and Shedlovsky (1971) can be fit fairly well using a quadratic approximation:

\[ \sigma = 2,075 \, C - 45 \, C^2 \]  

(15)

where \( C \) is the concentration in kg/m\(^3\) and \( \sigma \) is the conductivity in \( \mu S/cm \) at 25°C. 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.

Although the experimental values are for 25°C, they may be used at 20°C if multiplied by 0.89 (based on Equation (14) for the temperature dependence of conductivity). Thus the above relationship would be

\[ \sigma = 1,850 \, C - 40 \, C^2 \]  

(16)

3. DESCRIPTION OF FORTRAN CODE VHBORE

The main program is little more than a single time step loop and a series of subroutine calls. Before the time step loop begins, a single call to the subroutine INIT is made to initialize all data areas and read the problem descriptions. Within the time step loop, the flow rates are first computed using either constant or variable density approaches (using either subroutine FLOWS or subroutine DFLOWS). Once the flow rates are computed, a time step size is determined, and the mass transfer for the time step is accomplished using subroutine ONESTEP. If the time step was determined to be too large (too much mass was transferred out of a cell), the time step is reduced and the flow rates and mass transfer are reformed again. If the time step was successful, the subroutine GOODSTEP is called to check for output requests. After the ending time has been reached, a single call to the subroutine ENDPMB is made to produce the final messages regarding mass transfer.
Common Blocks

The program has four named common blocks, CORR, FRACS, SEGS and STEP which are defined in a separate source code file, VPCOMMS.F, and included as needed in the main source file. The common block source file uses a set of parameters to specify array sizes, making it easier to change array sizes as needed to suit different problems and computing environments.

Common block CORR contains the coefficients for converting electrolyte concentration in kg/m$^3$ to conductivity in $\mu$S/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, transmissivities, storativities, infiltration volumes, fluid densities, etc.

Common block SEGS contains the arrays describing each segment, including the concentration and density at the beginning and end of the time step, the fracture inflow average concentration and density, 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 and volume out of the system, cumulative mass and volume into the system, time at the beginning and end of the time step, number of fractures, conductivity output unit number, toggle index for the concentration array, etc.

Subroutines and Functions

The first level of subroutines includes INITs, FLOWS, DFLOWS, ONESTEP, GOODSTEP and ENDPROB. These subroutines are called by the main program.
INITS performs global variable initialization, opens the input and output files, reads the problem definition input file and performs the initial mass balance using subroutine SMASS.

The two main portions of the time step loop compute the variable flow rates and perform the mass transfer. For a constant density calculation, the variable flow rates are determined by subroutine FLOWS; while for a variable density case, the subroutine DLFLOWS is used. The primary difference between these two routines is that FLOWS using only depth to determine pressure differences (assuming a constant density fluid in the well bore), whereas DLFLOWS performs an actual integration of the density of all well bore fluid in order to determine the pressure at some depth. Both subroutines use the function QD to determine individual inflow rates using the Jacob-Lohman solution, and, after the individual inflow rates have been determined, both subroutines also call the subroutine FLOWSA to develop the full flow rate profile, estimate the time step size, and determine any velocity dependent aspects of the calculation.

The mass transfer for a single time step is carried out by subroutine ONESTEP. This subroutine advances the simulation clock, save current state variables in case the time step must be reversed, and calls subroutines CTFRAC and TSTEP. Subroutine CTFRAC determines the average concentration and density of the source terms for the time step based on the inflow rates, the time step, and any storage of fluid in the inflow zones. Subroutine TSTEP performs the actual mass transfer calculation and checks for conservation of mass. If the subroutine TSTEP indicates that an attempt was made to transfer too much mass out of a cell during the time step (due to the combined effects of advection and diffusion), a flag is set and the time step is reduced.

As mentioned above, the subroutine GOODSTEP is called if the time step was successful in order to determine if any output has been requested following the current time step. Actual output of conductivity profiles is produced by subroutine CPRT.
At the end of the computation, the subroutine ENDPROB is called to produce a number of informational messages about the calculation.

4. INPUT AND OUTPUT GUIDE

Input and Output Files

The model uses one input and four 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. The output files consist of (1) messages produced by the model, (2) conductivity-depth pairs for each borehole cell at the requested output times, (3) flow rate-depth pairs for each cell at the requested output times, and (4) density-depth pairs for each cell at the requested output times. The following table summarizes the input and output files and indicates their FORTRAN unit numbers.

<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>
<tr>
<td>8</td>
<td>OUTPUT</td>
<td>Flow rate profile</td>
</tr>
<tr>
<td>9</td>
<td>OUTPUT</td>
<td>Density profile</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. 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>XT OP</td>
<td>m</td>
<td>Top of study area, surface is zero and positions increase downward, adjusted if necessary to fit XBOT and DELX</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></td>
<td>WATLEV</td>
<td>m</td>
<td>Initial water level in well</td>
</tr>
<tr>
<td></td>
<td>TOPD</td>
<td>none</td>
<td>Initial density of well bore fluid normalized by density of water (spec. grav.). TOPD&lt;0 for constant density calculation</td>
</tr>
<tr>
<td>2</td>
<td>IFLIM</td>
<td>none</td>
<td>Number of inflows</td>
</tr>
<tr>
<td>2-1</td>
<td>XIN(I)</td>
<td>m</td>
<td>Position of inflow</td>
</tr>
<tr>
<td>I=1,IFLIM</td>
<td>CIN(I,1)</td>
<td>kg/m³</td>
<td>Constant concentration for formation water</td>
</tr>
<tr>
<td></td>
<td>TFRAC(I)</td>
<td>m²/sec</td>
<td>Inflow transmissivity</td>
</tr>
<tr>
<td></td>
<td>SFRAC(I)</td>
<td></td>
<td>Inflow storativity</td>
</tr>
<tr>
<td></td>
<td>HFRAC(I)</td>
<td>m</td>
<td>Inflow pressure (over p_g)</td>
</tr>
<tr>
<td></td>
<td>VFRAC(I, l)</td>
<td>m³</td>
<td>Inflow initial volume of infiltrated deionized water</td>
</tr>
<tr>
<td></td>
<td>CPZ(I,1,1)</td>
<td>kg/m³</td>
<td>Infiltrated water concentration</td>
</tr>
<tr>
<td></td>
<td>CIN(I,2)</td>
<td>none</td>
<td>Density of inflow fluid normalized by density of water (spec. grav.) (only for variable density calculation)</td>
</tr>
</tbody>
</table>
Table 4-2. Continued.

<table>
<thead>
<tr>
<th></th>
<th>IDHLIM</th>
<th>DELHT(I)</th>
<th>none</th>
<th>Number of water level changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>DELHH(I)</td>
<td>m</td>
<td></td>
<td>Time at which water level in borehole changes</td>
</tr>
<tr>
<td>4</td>
<td>TSTART</td>
<td>hr</td>
<td></td>
<td>Change in water level (positive down)</td>
</tr>
<tr>
<td>4.I</td>
<td>TEND</td>
<td>Hr</td>
<td></td>
<td>Problem start time</td>
</tr>
<tr>
<td></td>
<td>TSMAX</td>
<td>min</td>
<td></td>
<td>Problem end time</td>
</tr>
<tr>
<td></td>
<td>ILPT</td>
<td></td>
<td></td>
<td>Maximum time step</td>
</tr>
<tr>
<td></td>
<td>PT(I)</td>
<td>hr</td>
<td></td>
<td>Number of print times</td>
</tr>
<tr>
<td>5</td>
<td>ITYPDK</td>
<td>none</td>
<td></td>
<td>Time to print profile</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DK</th>
<th>m²/sec</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>ICON</td>
<td>none</td>
</tr>
<tr>
<td>6.I</td>
<td>X(I)</td>
<td>m</td>
</tr>
<tr>
<td>6.I</td>
<td>C0(I)</td>
<td>kg/m³</td>
</tr>
<tr>
<td>6.I</td>
<td>D0(I)</td>
<td>none</td>
</tr>
</tbody>
</table>

| OFFSET | μS/cm | Constant term for converting concentration in kg/m³ to conductivity in μS/cm |
| COEFA  | (μS·m²)/100 kg | Linear coefficient for converting concentration to conductivity |
| COEFB  | (μS·m⁵)/1000 kg² | Quadratic coefficient for converting concentration to conductivity |

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. The output subroutine CPRT also converts flow rates to liters per minute just prior to output.
5. EXAMPLES

Verification Example: Single inflow, single pressure change

The first example allows verification by comparing with Example 4.4 in Earlougher (1977). A single pressure change is applied to a single inflow. Earlougher's example refers to an inflow zone of length 57.9 meters; this will be represented in two different ways, both as a single inflow and as a set of 58 inflows with a cell length of one meter. The example applies a pressure drop of 703.6 meters (1000 psi). The hydrological characteristics include a permeability of 6.5 millidarcy and a viscosity of 1.35 centipoise, giving a transmissivity of \(2.697 \times 10^{-6} \text{m}^2/\text{sec}\). The porosity-compressibility product is \(2.05 \times 10^{-6}\) psi\(^{-1}\), giving a storativity of \(168.7 \times 10^{-6}\). The single drawdown occurs at time zero hours.

The input for this example problem with the inflow represented as a single point is shown below.

<table>
<thead>
<tr>
<th>LINE</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>000, 2000, 1.6096, 0, -1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2.1</td>
<td>1900, 1.00 2.697e-6 168.7e-6 1899.5 0.0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3.1</td>
<td>0 703.57</td>
</tr>
<tr>
<td>4</td>
<td>0,100,15,9</td>
</tr>
<tr>
<td>4.1</td>
<td>0.2</td>
</tr>
<tr>
<td>4.2</td>
<td>0.5</td>
</tr>
<tr>
<td>4.3</td>
<td>1</td>
</tr>
<tr>
<td>4.4</td>
<td>2</td>
</tr>
<tr>
<td>4.5</td>
<td>5</td>
</tr>
<tr>
<td>4.6</td>
<td>10</td>
</tr>
<tr>
<td>4.7</td>
<td>20</td>
</tr>
<tr>
<td>4.8</td>
<td>50</td>
</tr>
<tr>
<td>4.9</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>1.05e-3</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>73., 1870., -40.</td>
</tr>
</tbody>
</table>

The input for the case of 58 inflows is identical to that above except for lines 2 and 2.x. The transmissivity and storativity values have been scaled by 58:
The negative value at the end of the first line of the input indicates a constant density calculation. Both approaches to the inflow description result in the same number of simulation time steps (5,415), ranging from 7.8 to 73 seconds.

Figures 2 through 6 illustrate different aspects of the calculation. Figure 2 shows the flow rate as a function of time. The units are Earlougher's oil-field system of barrels per day (one liter per minute is 9.0573 barrels per day). This figure matches Figure 4.13 in Earlougher (1977). The flow rate is the same regardless of which representation of the inflow zone is used.

The flow rate and conductivity profiles for the single point representation are shown in Figures 3 and 4, and Figure 5 and 6 show the same for the 58-cell representation. The different approaches to modeling the inflow section have fairly limited effects. The flow rate shows a ramp for the line source, as opposed to a step for the point source, and there is a slower buildup of saturated concentration levels with the line source.

Variable Density Examples

In order to demonstrate the effect of well bore fluid density on the flow rate, two variations on the above single inflow, single pressure drop case are presented here. In the first, the well bore fluid is twice as dense as the inflow fluid; and, in the second case, the inflow fluid is twice as dense as the well bore fluid. The changes in the input file are shown below:

<table>
<thead>
<tr>
<th>LINE</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>58</td>
</tr>
<tr>
<td>2.1</td>
<td>1900. 1.00 46.495e-9 2.9092e-6 1899.5 0.0.</td>
</tr>
<tr>
<td>2.2</td>
<td>1901. 1.00 46.495e-9 2.9092e-6 1900.5 0.0.</td>
</tr>
<tr>
<td>2.3 to 57</td>
<td>(omitted)</td>
</tr>
<tr>
<td>2.58</td>
<td>1957. 1.00 46.495e-9 2.9092e-6 1956.5 0.0.</td>
</tr>
</tbody>
</table>
Figures 7 and 8 show the flow rate as a function of time for these two cases. In Figure 7, with the wellbore fluid having a higher density, the pressure difference between the inflow and the wellbore at the inflow depth is initially reduced, so the flow rate is also reduced. As the less dense fluid flows into the wellbore and the denser fluid is pumped out, the pressure difference at the inflow depth increases, until, after about 30 hours the denser fluid initially in the wellbore has been flushed out, and the problem converges with the constant density solution.

In Figure 8, with the inflow fluid having a higher density, the flow rate is initially similar to the constant density case. But as the denser fluid flows into the well, driving up the pressure at the inflow depth, the flow rate decreases. Eventually, after a few hundred hours, the flow stops since even with the drawdown of over 700 meters, the denser fluid in the wellbore has resulted in equilibrium between the well and the inflow at the inflow depth.

Figure 9 shows the driving pressure differences in meters of water as a function of time for these cases.

**Multiple Inflow, Multiple Pressure Change**

One more complex example is presented which involves three inflows and two drawdown changes. The problem is a variation of the constant density verification
example at the start of this section. The inflow and drawdown change input is as follows: shown below:

<table>
<thead>
<tr>
<th>LINE</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>1500.00 2.697e-6 168.7e-6 1399.5 0.0 0.0</td>
</tr>
<tr>
<td>2.2</td>
<td>1700.00 2.697e-6 168.7e-6 1799.5 0.0 0.0</td>
</tr>
<tr>
<td>2.3</td>
<td>1900.00 2.697e-6 168.7e-6 1899.5 0.0 0.0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3.1</td>
<td>10 703.57</td>
</tr>
<tr>
<td>3.1</td>
<td>50 703.57</td>
</tr>
</tbody>
</table>

Notice that when the water level is at the surface, there is a circulation within the well from 1700 meters to 1500 meters due to inflow pressure head differences. The water level changes approximate turning a pump on at 10 hours and off at 50 hours.

Figure 10 shows the flow rate across the inflow at 1500 meters as a function of time. Note that initially the flow is negative, i.e. water is flowing from the wellbore into the formation. Figure 11 shows the concentration of the water flowing across the inflow.

Figures 12 and 13 show the flow rate and conductivity profiles for the entire wellbore at times before, during and after pumping.

6. ACKNOWLEDGEMENTS

Discussions and cooperation with NAGRA personnel, especially S. Vomvoris and P. Bluemling are much appreciated. We would also like to thank Christine Doughty of LBL for reviewing and commenting on the manuscript and software. This work was carried out under U.S. Department of Energy Contract No. DE-AC03-76SF00098 for the Director, Office of Civilian Radioactive Waste Management, Office of External Relations, administered by the Nevada Operations Office in cooperation with the Swiss National Cooperative for Disposal of Radioactive Waste (Nagra).
7. REFERENCES


Jacob, C.E. and S.W. Lohman (1952), “Nonsteady flow to a well of constant drawdown in an extensive aquifer,” Transactions, American Geophysical Union, v. 33, no. 4, pp. 559-569.

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


Figure 1. Dimensionless flow rate as a function of dimensionless time, showing the three solution techniques used in different subdomains.
Figure 2. Flow rate as a function of time for $T=2.697\times10^{-6}$ m$^3$/m/s, $S=168.7\times10^{-6}$, drawdown of 703.57 meters. Matches Figure 4.13 in Earlougher (1977) for example of constant-pressure testing in an infinite-acting reservoir (Earlougher's Example 4.4).
Figure 3. Flow rate profile for 1-cell inflow.

Figure 4. Conductivity profile for 1-cell inflow.
Figure 5. Flow rate profile for 58-cell inflow.

Figure 6. Conductivity profile for 58-cell inflow.
Figure 7. Flow rate as a function of time with initial well bore fluid density doubled. Initially, the flow rate is lower than the constant density case; as the higher density wellbore fluid is flushed out, the flow rate converges with the constant density curve.

Figure 8. Flow rate as a function of time with inflow density doubled. Initially, the flow rate is the same as the constant density case, but as the higher density fluid flows into the well bore, the driving pressure difference at the inflow depth decreases until the flow stops after a few hundred hours.
Figure 9. Driving pressure difference (in meters of water) as a function of time. For the constant density calculation, this is simply the drawdown change of 703.6 meters. With higher density initial wellbore fluid, the pressure difference is reduced initially but converges with the constant density solution. With higher inflow density, the difference is initially the same as the constant density case, but decreases to zero as higher density fluid moves into the well bore.
Figure 10. Flow rate as a function of time across the inflow at 1500 m. Notice that the flow is initially negative, from the borehole into the formation. The pump is turned on at 10 hours, and it is turned off at 50 hours.

Figure 11. Fluid electric conductivity as a function of time across the inflow at 1500 m. The most striking feature is the delayed rise in conductivity after the pump starts at 10 hours. This is due to pre-production infiltration of low conductivity fluid from borehole to inflow formation. At late times after pumping, as more of the high conductivity fluid moves out of the wellbore, the concentration gradually drops as the front moves toward 1500 m.
Figure 12. Flow rate profile for multiple inflow, multiple drawdown. Before pumping (e.g., 0.2 hr) there is a circulation from 1700m to 1500m. During the pumping period (e.g., 20 hr), the typical inflow pattern is seen. After pumping (e.g., 100 hr), there is a slight inflow at 1700m, a slight outflow at 1900m, and an outflow at 1500m.

Figure 13. Conductivity profile for multiple inflow, multiple drawdown. Flow before pumping (up to 10 hr) is from 1700m to 1500m. During pumping (10 to 50 hr), inflows are seen at 1900m. 1700m and 1500m. After pumping (after 50 hr), flow is again toward 1500m, both from above and below, pulling the high conductivity fronts back toward 1500m.
fracture/borehole transient flow model

June 1994

Lawrence Berkeley Laboratory, Earth Sciences Division

Implicit double precision (a-h)
Implicit double precision (o-z)

include 'vpcomms.f'

call iisits()

BEGINNING OF TIME STEP LOOP

compute flow rates and time step

100 if (ivard.eq.0) then
   call flows
else
   call dflows
endif

call onestep

if the time step is to big, try again

if (ifailt.eq.0) goto 200
   t = tp
   write(iout,*)'Reducing time step...'
   goto 100

TIME STEP SIZE IS GOOD

100 call goodstep

END OF TIME STEP LOOP

calculate flows through each segment

this subroutine is called at the beginning of each time step

subroutine flows

implicit double precision (a-h)
Implicit double precision (o-z)

include 'vpcomms.f'

maximum time step is minimum time to flush one-tenth of a segment. Note that this time may still be too large

if dispersion is significant
if (ifailt.eq.0) then
   if (qtmax.ne.0.) then
      delt = 0.5 * vol/qtmax
   else
      delt = 0.01 * tsmax
   endif
endif

initialize segment flows

do i=1,iflim
   qin(i) = 0.
   qin(i) = qin(i) + (qdtid * qcoef(i) * delhh(idraw))
   enddo

compute flow into segments from fractures

it is possible to have more than one fracture in a single segment

do i=1,iflim
   qin(i) = hfrac(i)-xs(ifseg(i))-watlev
   if (qin(i).ne.0. .and. t.ge.tstart) then
      qdtid = avgint(t,tstart,tcoef(i),delt)
      qin(i) = qin(i) + qdtid * qcoef(i) * qin(i)
   enddo
   qfrac(ifseg(i)) = qfrac(ifseg(i)) + qin(i)
   enddo

try out a time step

subroutine onestep()

implicit double precision (a-h)
Implicit double precision (o-z)

include 'vpcomms.f'

ifailt = 0
   tp = t
   t = t + delt

save current mass and volume totals

gtmins = gtmina
gtdns = gtdina
gtmos = gtmout
gtdos = gtdout
gtfbms = gtfbm
gtbfms = gtbfm

```fortran
ghbore.£

Wed Oct 5 16:04:32 1994
tovins = gtvina
tovos = gtvout
tofbvs = gtfbv
tobfvs = gtbfv
topms = topm
topds = topd

*compute fracture inflows and concentrations to the borehole*
call cfrac

*compute mass transfer for the time step*
call tstep
return
end

*complete a time step*
subroutine goodstep

implicit double precision (a-h)
implicit double precision (o-z)
include 'vpcomms.f'

do j=1,ilpt
  if (t.ge.pt(j)) then
    call cprt
    pt(j)=2.*tend
  enddo
if (delt.le.O.) then
  writeout, *) 'ERROR—time step <= 0!'
  stop
else
  tsavg = tsavg + delt
  ntstep = ntstep + 1
  if (delt.lt.tslo) tslo = delt
  if (delt.gt.tshi) tshi = delt
endif

*write extra output for report plots*
cfl = cfrac(ifsegd), 1)
write(80,'(3fl2.5)')((t-delt/2.)/3600),
(cfl(0:2)+cflM coefa + cfl*coefb))
return
end

*initialize constants, etc.*
subroutine inits

implicit double precision (a-h)
implicit double precision (o-z)
include 'vpcomms.f'

cfl 

do i=1,imax
cfrac(i,l)=0.
delt 

*input and initialization*
do i=1,imax
  cfrac(i,l)=0.
  enddo
  cfrac(1,l)=0.
else
  if (topd.0.) then
    liin = 5
    writeout, *'(Variable density model')
  else
    liin = 6
    topd = 1.0
    writeout, *'(Constant density model')
endif

*convert diameter in cm to m*
diam = diam/100.

*check geometry*
do (xbot.lt.xtop) then
```

This text snippet appears to be a Fortran program, possibly a subroutine that deals with borehole parameters and flow computations. It's a part of a larger code that likely models the flow of fluids through a borehole, possibly for a hydrological or geological study. The code includes functions for calculating inflows and concentrations, managing the time step, and handling input parameters like borehole diameter, density models, and other geological data.
write(iout,940)
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)

write(iout,960)ilim,delx
writedout, 970)deltop,xtop
else
endif

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
    writedout, 970)deltop,xtop
  else
    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 i=l,ilim
    xsd(i) = xbot + (0.5-i)*delx
  enddo
  c
  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
  c*** read fracture data
  c
  read(iin,*xhlim)
  if (iflim.gt.ifmax) then
    write(iout,910)ifmax
  stop
  endif
  idhlim = idhlim
  if (i+hlim.gt.idhmax) then
    write(iout,1910)idhmax
  stop
  endif
  j=ihlim
  do i=l,j
    readdin, *)inx(i) ,cin(i,l), tfrac(i),sfrac(i),hfrac(i),
    vfrac(i),cfz(i,l,1),cfz(i,l,2)
    endif
    if (vfrac(i,1).gt.0.) then
      i+znum(i) = 2
    else
      i+znum(i) = 0
    endif
  c
  c*** locate fracture on segments
  c
  iskipit = 0
  if (xin(i).gt.xbot) .or. (xin(i).lt.xtop) then
    write(iout,910)xin(i)
    iflim = iflim-1
    iskipit = 1
  endif
  if (iskipit.eq.0) then
    ifseg(i) = iconvxlxtop,xbot,delx,xin(i))
    rad = diam/2
    tcoef(i) = tfrac(i) / (sfrac(i) *rad*rad)
    qcoef(i) =2. * pi * tfrac(i)
  endif
  enddo
  c
  c*** read drawdowns and times
  c*** input stream -- free format
  c*** number of changes
  c*** one record for each change--
  c*** time (hrs), drawdown (m)
  c
  c*** read drawdown data
  c
  read(iin,*hdhlim)
  if (idhlim.gt.idhmax) then
    write(iout,1910)idhmax
  stop
  endif
  idhlim = idhlim
  do i=1, idhlim
    readdin, *)delht(i) ,delhh(i)
    delht(i) = delht(i) * 3600.
  enddo
  c
  c*** read time parameters
  c*** input stream--
  c*** start time (hrs), end time (hrs), number of prints
  c*** then, print times (hrs)
  c*** note: all times are converted to seconds
  c
  read(iin,*tstart,tend,tsmax,ilpt)
  tstart = tstart * 3600.
  tend = tend * 3600.
  tsmax = tsmax * 60.
  if (tend.lt.tstart) then
    write(iout,2910)
  stop
  endif
  if (ilpt.gt.impt) then
    write(iout,915)impt
  endif
  c
**vhbore.f**

---

**Wed Oct 5 16:04:32 1994**

```fortran
! read in print-time parameters
read(in,*)(pt(i),i=1,ilpt)
do ilpt=1,ilpt
   pt(i)=pt(i)*3600.
enddo

! read dispersion parameter
! input stream--(one record)
! type code, parameter
! at the end of this routine, dks has units m**3/sec
creadin,*itypdk,dk
   if (itypdk.eq.1) then
      dksi = dk * area/delx
   else
      do 1=1,ilim
         dks(i) = dksi
      enddo
   endif

! read initial concentrations and densities
! input records:
! icOn number of initial values specified
! x, c0 pairs of x values and initial concentrations
creadin,*icOn
   if (icOn.gt.0) then
      do i=1,icOn
         if (ivard.eq.0) then
            read(iin,*)x,c0
            do 2=1,1
               cxt(i,l,1)=c0
               cxt(i,2,2)=c0
            enddo
         else
            read(iin,*)x,c0,d0
         endif
         if ((x.gt.xbot) .or. (x.lt.xtop)) then
            writedout,900)x
         else
            iseg=iconvx(x,xbot,delx,x)
            if (cxt(iseg,1,1).eq.0.) then
               cxt(iseg,1,1)=c0
               cxt(iseg,2,2)=d0
            else
               writedout,915)x
            endif
         endif
      enddo
   endif
enddo
creadin,*offset,coefs,coefs
            ! compute the starting mass and volume in the system
            ! write(sout,'(9(9f15.5,1p,f15.2))')
close input file after reading data
close (unit=iin)
compute the starting mass and volume in the system
write(iout,*)'Input processed...'
call mass
sdf = sfb
sd = sdc
s = sdb
topv = top
mthi = 0.
talo = 1.e30
trev = 0.
tf2 = 0.
t = tetart
delt = 0.003*tsmax
return
```

---

**NOTE***

- initial concentration position', invalid--ignored'
- position is ',f15.5)
- Maximum number of fractures is ',i4,' -- aborting')
- Maximum number of print times is ',i4,' -- aborting')
- Top of region is below bottom of region--aborting')
- Region has been divided into ',i4,' segments with', length ',f10.4,' (m)'
- Maximum number of segments is ',14,' -- aborting')
- Maximum number of drawdowns is ',i4,' -- aborting')
- End time before start time—aborting')
- *** NOTE *** multiple initial concentrations', second value ignored—position is ',f15.5)
**vhbore.f** Wed Oct 5 16:04:32 1994

```fortran
implicit double precision (a-z)

idxup = (xbot-x) / delx+1.
if (x.eq.xtop) idxup = idxup-1
iconvx = idxup
return
end

c*** compute the mass and volume in the system

subroutine smass()

smb = 0.
sdb = 0.
c*** sum mass in borehole

do i=1,ilim
   smb = smb + cxt(i,itog,1)
   sdb = sdb + cxtd(i,itog,2)
enddo
smb = smb
sdb = sdb

c*** compute volume in borehole

svb = ilim * vol

c*** sum mass and volume in fractures

smf = 0.
sdf = 0.
svf = 0.
do i=1,iflim
   if (ifznumd) .gt.0) then
      do j=1,ilim
         vij = vfrac(i,j)
         smf(vij) = smf + csi(j,1,vij)
         sdf(vij) = sdf + csi(j,2,vij)
         svf(vij) = svf + vij
      enddo
   endif
enddo

return
end

c*** subroutine cprt

subroutine cprt

c*** print out concentration, flow rate and density 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*** note: units of output conductivity is S/m

c*** subroutine cprt

implicit double precision (a-h)

sigma(i) = ofset + ci*(coefa + ci*coefb)
qlmd(j) = 60000. *qnxt(i)
enddo

writedout,920)th
write(iprt, '(2fi5.5)')(xs(i),sigma(i),i=ilim,1,-1)
writedout,920)th
write(idout,'(2fi5.5)')(xs(i),cxtd(iitog,2),i=ilim,1,-1)
writedout,920)th
return
920 format(/' Printing conductivity at time ',el5.5,' (hrs)')
end

c*** end of problem

 subroutine endprob()

implicit double precision (a-h)

include 'vpcomms.f'

check for final print

needp = 0
do j=1,ilpt
   if (pt(j)(2.*tend)) needp=1
enddo
if (needp.ne.0) call cprt

c*** end of problem, check mass conservation

call smass

c*** total system mass

esmass = smf + smb + topsm
sdens = sdf + sdb + (topsd0*topv) - gdensout
edens = (tcd0 + tcd0 + (topd0*topv) + gdensin) - gdensout
per = .0.
dur = 0.
if (esmass.ne.0.) per = (esmass-esmass) / esmass * 100.
if (sdens.ne.0.) dur = (sdens-sdens) / sdens * 100.

```
```c
exf = (tmO + gtbfv) - gtbm
exv = (tvO + gtbfv) - gtvb
pverr = 0.
if (exf.ne.0.) pverr = ((svf-exf) / exf) * 100.
if (exv.ne.0.) pverr = ((svb-exv) / exv) * 100.

*** total system volume (convert from m**3 to l)
if (topv.gt.topvO) then
  gtvout = gtvout + (topv-topvO)  
else
  if (topvO.lt.topv) then
    gtvout = gtvout + (topvO-topv) 
endif
svf = (svf-exfv) / exfv) * 100.
svb = (svb-exb) / exb) * 100.
tvO = tvO + topvO + tdfO + gtvina - gtvout

gtvina = gtvina + (topv - topvO)
gtfb = (gtfb - gtvout) + topvO - topv

evol = (tv0 + tvf0 + topv0 + gtvina) - gtvout

gtvout = gtvout + (topvO-topv)

evol = svf + svb + topv

exfd = (tdfO + gtbfd) - gtfbd
exfm = (tmfO + gtbfm) - gtfbm
if (exvol.ne.0.) pverr = ((evol-exvol) / exvol) * 100.

*** fracture volume
exf = (tvf0 + gtbfv) - gtfv0
pverr = 0.
if (exf.ne.0.) pverr = ((svf-exf) / exf) * 100.

write(*,*) 'Expected ending mass (kg): ', edens
write(*,*) 'Actual ending mass (kg): ', edens
write(*,*) (Boreholes (kg): tau, sb,)
write(*,*) (Fractures (kg): tau, sb,)
write(*,*) (Top section (kg): tau, sb,)
write(*,*) 'Percent error: ', derr
write(*,*) 'VOLUMES BALANCE:'
write(*,*) 'Starting vol in system (l): ', (topvO+tvfO+topv)
write(*,*) 'Boreholes : tvO, '
write(*,*) 'Fractures : tvfO, '
write(*,*) 'Top section : topv, '
write(*,*) 'Actual ending vol (l): ', svf
write(*,*) 'Mass to frac from bore (kg): ', gtbm
write(*,*) 'Mass to frac from frac (kg): ', gtbm
write(*,*) 'Expected ending vol (l): ', svf
write(*,*) 'Mass to frac from bore (kg): ', gtbm
write(*,*) 'Mass out of system (1): ', gtvout
write(*,*) 'Number of time step used: ', nntstep
write(*,*) 'Minimum time step used (min): ', (tsto/60.)
```
**whbore.f**

Wed Oct 5 16:04:32 1994 7

```fortran
write(iout,*)'Average time step used (min):',((tsavg/60.) /ntstep)
write(iout,*)'Maximum time step used (min): ',(tshl/60.)
write(iout,*)'Maximum percentage cell mass transfer: ',pcmax
write(iout,*)
write(iout,*)'*** PROBLEM FINISHED ***
return
**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***
imPLICIT DOUBLE PRECISION (A-H)
imPLICIT DOUBLE PRECISION (O-Z)
**c***
include 'vpcomms.f'
**c***
tml — initial mass in segment
tm2 — final mass in segment
dmal — rate change in mass due to advection from left (+ = in)
dmar — rate change in mass due to advection to right (- = out)
dmdl — rate change in mass due to diffusion with left (+ = in)
dmdr — rate change in mass due to diffusion with right (- = out)
dmf — rate change in mass due to fracture inflow (+ = in)
dme — rate change in mass due to external outflow (- = out)
**c***
pcmt = 0.
fact = 1.e30
dme = 0.
dde = 0.
**c*** loop over each segment
**c***
itogx = 3-itog
do 1=1, ilim
ip = i+1
cti = cxt(i, itog, 1)
dti = cxt(i, itog, 2)
starting mass in the segment
**c***
tax = cxti * vol
tax = cxti * vol
**c*** mass transfer with fracture zone
**c***
if (cfrac(i,1).ne.0.0) then
  dmf = cfrac(i,1) * qfrac(i)
else
  dmf = 0.
endif
if (cfrac(i,2).ne.0.0) then
  dd = cfrac(i,2) * qfrac(i)
else
  dd = 0.
endif
**c*** advection/diffusion with next lower cell
**c***
**c*** advection/diffusion with next upper cell
**c***
**c*** note: full upstream weighting is used. without
**c*** upstream weighting, these equations would
**c*** use (cti+ct(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
dder = -cxti * qnxt(i)
dde = -dd * qnxt(i)
else
dder = -cxt(ipl, itog, 1) * cxt(ipl, itog, 2) * qnxt(i)
dde = -cxt(ipl, itog, 1) * cxt(ipl, itog, 2) * qnxt(i)
endif
**c*** if the current cell has an outflow,
**c*** advection mass in flows out to outflow (as much as
**c*** possible)
**c***
if (qnxt(i-1).gt.0.) then
  if (abs(qfrac(i)).ge.abs(qnxt(i-1))) then
    dmf = -dmar
    dd = -ddar
    if (dmdr.gt.0.) then
      dmf = dmf - dmdr
    endif
  else
    dmf = -abs(qfrac(i))/abs(qnxt(i-1)) * dmar
    dd = -abs(qfrac(i))/abs(qnxt(i-1)) * ddar
    if (dmdr.gt.0.) then
      dmf = dmf - abs(qfrac(i))/abs(qnxt(i-1)) * dmdr
    endif
  endif
endif
**c*** if (qnxt(i).lt.0.) then
  if (abs(qfrac(i)).ge.abs(qnxt(i))) then
    dmf = -dmar
    dd = -ddar
    if (dmdr.gt.0.) then
      dmf = dmf - abs(qfrac(i))/abs(qnxt(i)) * dmdr
    endif
  else
    dmf = -abs(qfrac(i))/abs(qnxt(i)) * dmar
    dd = -abs(qfrac(i))/abs(qnxt(i)) * ddar
    if (dmdl.gt.0.) then
      dmf = dmf - abs(qfrac(i))/abs(qnxt(i)) * dmdl
    endif
endif
```

The code snippet above is a subroutine for calculating concentrations at each time step in a mass transport model. It includes calculations for initial and final masses, rate changes due to advection, diffusion, and fracture inflow, as well as handling external outflows and upstream weighting.
outflow considered only at top (no mass flow back in at top)

if (qnx(i).gt.0.) then
  dme = -cxt * qnx
  dde = -dxt * qnx
else
  if (qnx(i).lt.0.) then
    dde = -top * qnx
  endif
endif
endif

check for segment mass limitations

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

summo = 0.
if (dmf .lt.O.) summo = summo - dmf
if (dme .lt.O.) summo = summo - dme
if (dmar .lt.O.) summo = summo - dmar
if (dmar .lt.O.) summo = summo - dmar
if (dmdr .lt.O.) summo = summo - dmdr

attempted to transfer more than 10% mass out of the cell
-time step is too big
if (summo .gt.O.) then
  pcm = 100.*(summo*delt) / tml
  if (pcm .gt.pcm) pcm = pcm
  if (pcm .gt 100) then
    f = 100./pcm
    if (f .lt.fact) fact = f
    ifailt = 1
  endif
endif

compute new mass in segment

tm2 = tml + (dmf+dme+dmal+dmdl+dmar+dmdr)*delt
td2 = tdl + (ddf+dde+ddal +ddar )*delt
cxt(i,itogx,1) = tm2/vol
cxt(i,itogx,2) = td2/vol

*** end of loop over segments

*** if the mass transfer is good (time step not too big),
*** add mass to fracture zone volumes if necessary

do i=1,iflim
  if (qin(i).lt.O) then
    vii = abs(qind) *delt
    cii = qfmass(ifseg(i))/vii
    dii = qfdens(ifseg(i))/vii
    ifznum = ifznum(i)+1
    if (ifznum.gt.ifzmax) then
      enddo
  endif

  vim = (cii+cim)/vii
  vdm = (dii+dmetop)/vii
  vtfm = vtfm + vii
  vtm = vtm + vim
  vtd = vtd + vdm
  gbfm = gbfm + vii*cii
  gtfm = gtfm + vii*dii
  gbfd = gbfd + vii*cim
  gtfd = gtfd + vii*dii
  gbfv = gbfv + vii*cii
  gtfv = gtfv + vii*dii
enddo

*** transfer to fracture

vtfm(i,ifznum(i)) = vii
cfs(i,1,1) = cfs(i,1,1)*vtfm(i,1)
cfs(i,1,2) = cfs(i,1,2)*vtfm(i,2)
else
  if (j .le.10) then
    cfs (1,1,1) = cfs(1,1,1) + (cfs(i,1,1)*vtfm(i,1)+cfs(i,1,2)*vtfm(i,2))
    cfs (1,1,2) = cfs(1,1,2) + (cfs(i,1,1)*vtfm(i,1)+cfs(i,1,2)*vtfm(i,2))
    vtfm(i,1) = vtfm(i,1) + vtfm(i,1)
    else
    cfs (1,j-9,1) = cfs(i,j-9,1)
    cfs (1,j-9,2) = cfs(i,j-9,2)
    vtfm(i,j-9) = vtfm(i,j-9)
  endif
enddo

cfs(1,1,1) = cfs(1,1,1) / vtfm(1,1)
cfs(1,1,2) = cfs(1,1,2) / vtfm(1,2)
ifznum(i) = ifzmax+8
enddo
if (pcmt.gt.pcmax) pcmax = pcmt

c*** write out flow at top in liters/min
   thrs = t / 3600.
   ql = qnxt(ilim) * 60000.
   qol = gtvout * 1000.

   c*** toggle row pointer
   itog=itogx

   c*** emass is expected total mass in system (total in - total out)
   c*** gtm is actual total mass in system
   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
   call emass
   gtm = smb + smf + topm
   gtd = sdb + sdf + (topd*topv)
   if (gtmina.ne.0.) then
      emass=(tm0 + tmfo + topmO + gtmina)-gtmout
      errm=gtm-emass
      errp=(100.*errm)/(tmO + tmfO + topmO + gtmina)
      if (abs(errp).gt.0.1) then
         tmOx = tmO + tmfO + topmO
         write(iout,900)errp,gtm,emass,tmOx,gtmina,gtmout,
                  smb,smf,topm
         stop
      endif
   endif
   if (gtdina.ne.0.) then
      edens=(td0 + tdf0 + (topd0*topv0) + gtdina)-gtdout
      errd=gtd-edens
      drrp=(100.*errd)/(td0+tdf0+(topd0*topv0)+gtdina)
      if (abs(drrp).gt.0.1) then
         tdOx = tdO + tdfO + (topd0*topv0)
         write(iout,920)drrp,gtd,edens,tdOx,gtdina, gtdout,
                  sdb,sdf,(topd*topv)
         stop
      endif
   endif
else
   c*** time step was too big -- cancel mass transfer, reduce delta-t
   ot = delt
   dalt = 0.5 * (dalt + fact)
   gtmass = gtmass
   gtdima = gtdima
   gtdout = gtdout
   gtdb = gtdb
   gtdma = gtdma
   gtdv = gtdv
   gtdv = gtdv
   gtdv = gtdv
   gtdv = gtdv
   gtdv = gtdv
   gtdv = gtdv
   c*** restore fracture values if the time step is too big
   do i=1,ilim
      frac(i) = frac(i)
      if (i.eq.1) then
         qtot(i) = qtot(i)
      else
         isgn = qtot(i)*qfrac(i)
         if (isgn.ge.0.) then
            qtot(i) = abs(qtot(i-1)) + qfrac(i)
         else
            if (abs(qfrac(i)) .le.abs(qtot(i-1))) then
               qtot(i) = abs(qtot(i-1))
            else
               qtot(i) = abs(qfrac(i))
            endif
         endif
      endif
   enddo
end

900 format(*** NOTE *** mass conservation error ',fl0.5,' (%)'
        Observed mass in system (kg) ',£20.5/
        Expected mass in system (kg) ',£20.5/
        Initial mass in system (kg) ',£20.5/
        Added mass to system (kg) ',£20.5/
        Removed mass from system (kg) ',£20.5/
        Mass in borehole (kg) ',£20.5/
        Mass in fractures (kg) ',£20.5/
        Mass in top (kg) ',£20.5/
910 format(£215.6)
920 format(*** NOTE *** dens conservation error ',£10.5,' (%)'
        Observed mass in system (kg) ',£20.5/
        Expected mass in system (kg) ',£20.5/
        Initial mass in system (kg) ',£20.5/
        Added mass to system (kg) ',£20.5/
        Removed mass from system (kg) ',£20.5/
        Mass in borehole (kg) ',£20.5/
        Mass in fractures (kg) ',£20.5/
        Mass in top (kg) ',£20.5/
end
**vhbore.f**

```fortran
integer, parameter :: ilim = 10

c*** compute flow to next cell up the hole, qnxt(i+1) is flow
c*** out of (into) the top of the section

c
qnxt(i) = qnxt(i+1) + qfrac(i)
endif
enddo

c

c*** compute maximum time step and Reynold's number

c
qtmax = qtot(i)
do i=2,ilim
   if (qtot(i).gt.qtmax) qtmax=qtot(i)
enddo

vmax = qtmax/area
diam = 2.*sqrt(area/3.14159)
rnmax = vmax*diam / 0.3e-6
dtm = delt/60.


c

c*** compute the velocity dependent dispersion coefficients

c*** if dispersion type is 2 or 3

c
if (itypdk.gt.l) then
   dksi = dk*area/delx
endif

find first nonzero, minimum, maximum and mean velocities

gfnz = 0.
gmin = 1.020
gmax = 0.
if (itypdk.eq.2) then
   do i=l,ilim
      if (qtot(i).eq.0.) then
         qfnz=qtot(i)
      else
         if (qtot(i).gt.gmax) qmax=qtot(i)
         if (qtot(i).lt.gmin) qmin=qtot(i)
      endif
   enddo
else
   do i=l,ilim
      qtoti = qtot(i)*qtot(i)
      if (qtoti.eq.0.) then
         qfnz=qtoti
      else
         if (qtoti.gt.qmax) qmax=qtoti
         if (qtoti.lt.qmin) qmin=qtoti
      endif
   enddo
endif

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

do i=l,ilim
   if (qtot(i).eq.0.) then
      qtoti = qfnz
   else
      qtoti = qtot(i)
   endif
   if (itypdk.eq.3) qtoti = qtoti*qtoti
endif

c

c*** compute harmonic mean of scaled dispersion parameter

c
c
```

**910 format(/' *** NOTE *** average flow rate is zero -- aborting') and
** subroutine ctfac
** c*** prepare fracture concentrations for time step
** c*** this subroutine is executed during each time step,
** c*** and it loops over each fracture
**
```

** subroutine ctfac
** implicit double precision (a-h)
** implicit double precision (o-z)
** include 'vpcomms.f'

subroutine ctfac
implicit double precision (a-h)
implicit double precision (o-z)
include 'vpcomms.f'

c*** reset values first

do i=l,iflim
   cfrac(i,1,1)=0.
cfrac(i,2,1)=0.
enddo

c*** save fracture values in case the time step is too big

do i=1,iflim
   n = ifznum(i)
   fsaved(i,1)=ifznum(i)
   if (n.gt.0) then
      do j=1,n
         fsave(j,1) = vfrac(i,j)
         fsave(j,2) = cfz(i,j,1)
         fsave(j,3) = cfz(i,j,2)
      enddo
   endif
enddo

c

c*** compute concentration

if (qin(i).eq.0.) then
   c = cin(i,1)
c = cin(i,2)
d = cin(i,3)
de = cin(i,4)
else
   if (ifznum(i).eq.0.) then
      c = cin(i,1)
c = cin(i,2)
d = cin(i,3)
de = c*0.3e-6
d = c*0.3e-6
d = c*0.3e-6
d = c*0.3e-6
else
   c = qin(i)*delt
endif
enddo
```

**v = qin(i)*delt**
include "vpcmas.f"

**computes the pressures in each segment**

```fortran
    wold = watlev
    watlev = watlev -
    do idraw=1,idhlim
      (if (v.gt.(vz+vtz))
        watlev = watlev + delhh(idraw)
      enddo

    if (v.ne.0.) then
      dv = (wold - watlev)*time
      else
        if (v.ge.(vz+vtz)) then
          watlev = watlev + delhh(idraw)
        enddo
    endif

    enddo
```

**maximum time step is minimum time to flush one-tenth of a segment. note that this time may still be too large if dispersion is significant**

```fortran
    if (vz.lt.v) then
      c = cm / v
      cm = cm + cz(i,1)*v
      else
        c = 0.
        cm = cm + cz(i,2)*v
      endif
      enddo
```

```fortran
    **inflow of mass flow during entire time step (kg/sec)**
```

```fortran
    if (qfrac(i).ne.0.0.) then
      cfrac(i,1) = cfrac(i,1) + c*qin(i)/qfrac(i)
      cfrac(i,2) = cfrac(i,2) + d*qin(i)/qfrac(i)
    endif
    enddo
```

**initialize segment flows**

```fortran
    do ilim,ilim
      qfrac(i) = 0.
    enddo
```

**compute flow into segments from fractures**

```fortran
    if (ilim.ge.0) then
      if (qtmax.ne.0.) then
        delt = 0.1 + *qtmax
        else
          delt = 0.91 + *qtmax
      endif
```

**compute flow rate as a function of pressure drop**

```fortran
    qin(i) = 0.
  ```
if (iplim(1) .eq. 0) then
    iplim(1) = 2
    phist(1,1,1) = hfrac(1)
    phist(1,1,2) = -l.e20
    phist(1,2,1) = xse(ifseg(1)) - watlev
    phist(1,2,2) = tstart
endif
if (dabs( pressdfsegd ) - phist(i, iplimd), 1) ) .gt. (delx/2.) ) then
    iplimd(1) = iplim(i) + 1
    if (iplimd(1) .gt. npdps) then
        writedout, * , 'Max iplim is', npdps, ' -- aborting'
        do iii = 1, npdps
            writedout, * , phist(i, iii, 1)
        enddo
        stop
    endif
    phistd(iplimd), 1) = press(ifseg(i))
    phistd(iplimd), 2) = t
endif

note: if pressure change occurs exactly on a
time step boundary, there is no result
do ip=2, iplimd1)
  c*** calculate effect of change between times
c  if (t .ge. phist(i, ip, 2)) then
    qdtid = avgint(t, phistd, ip, 2), tcoef(i), delt)
    qin(i) = qin(i) +
        ( qdtid * qcoefd) * (phistd(dp-l),l)-phist(i,ip,l))
  endif
doto

qfrac(ifseg(i))=qfrac(ifseg(i))+qin(i)
doto
call flowsa
return
end

c*** determine effective dimensionless flow rate for time step
c double precision function avgint(t, tp, tcoef, delt)
implicit double precision (a-h)
implicit double precision (o-z)
dimension ts(101), qs(101)
c*** first check endpoints
c write(6,998) (t/3600., (tp/3600., delt))
998 format(1 ' in avgint, t, tp, delt', (sec) = ',3f13.4)
ts(1) = t - tp
qs(1) = qd(ts(1)*tcoef)
to(2) = ts(1) + delt
qs(2) = qd(ts(2)*tcoef)
nvals = 2
slope = (qs(2) - qs(1)) / (ts(2) - ts(1))
write(6,999) slope, delt
999 format(' slope=',.6f4.5, ' delt=',.6f4.5)
if (slope .gt. 1.0e-2) then
    nvals = 101
te(nvals) = ts(2)
qe(nvals) = qs(2)
dt = delt/(nvals-1.)
c write(6,9) Integrating qd...
c write(6,9) qd(1) = qs(1)
c write(6,9) qd(101) = .,qs(nvals)
do i=2,(nvals-1)
t(i) = (t(i-p) + (i-1)*dt)
qi(i) = qd(te(i)*tcoef)
endo
endif
endif

sun = 0.
do 1, nvals
    if (i .eq. 1 or i .eq. nvals) then
        sum = sum + qe(i)/2.
    else
        sum = sum + qe(i)
    endif
endo
sum = sum / (nvals-1.)
ec return
end

c*** compute dimensionless flow rate as a function of dimensionless
time.
c*** input: dimensionless time, td
c*** output: dimensionless flow rate, qd
c*** error conditions:
c*** if td < 1.e-4, qd is set to first value in table
c*** if an interpolation error is detected,
c*** qd is set to one endpoint value, and
c*** an error message is printed

c*** solution method: table lookup and linear interpolation
c*** range: 1.e-4 < td <= 1.e12
c*** references: table 1, page 561, of
"Nonsteady flow to a well of constant drawdown in an
extensive aquifer," by C.E. Jacob and S.W. Lohman,
Transactions, American Geophysical Union, v.33, no.4,
August 1952, pp. 559-569.
c*** solution method: direct numerical approximation

c*** range: 1.e-4 < td

c*** references: equation 4.41 of
"Advances in Well Test Analysis," by Robert C.
Earlougher, Jr., American Institute of Mining,
Metallurgical, and Petroleum Engineers, Inc., 1977,
p. 40.
c
double precision function qd(td)
imPLICIT double precision (a-h)
imPLICIT double precision (o-z)
dimension qdtd(145), tds(145)
```

```