MASTER

DISCLAIMER

This book 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, makes 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.

DEVELOPMENT OF AN
INTER-LAYER SOLUTE TRANSPORT ALGORITHM
FOR SOLTR COMPUTER PROGRAM

PART 1: THE ALGORITHM

Prepared by
Ian Miller and Kathleen Roman

Prepared for Lawrence Livermore Laboratory
✓ by Golder Associates, Inc. *which City Kirkland, WA*
Under Contract 8770703 *per Davis E. Davis*
LLH

Distribution:

35 copies - Lawrence Livermore Laboratory
10 copies - Golder Associates

77308N
D/73

DECEMBER 1979

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED *JP*

Golder Associates

PREFACE

In order to perform studies of the influence of regional groundwater flow systems on the long-term performance of potential high-level nuclear waste repositories, it was determined that an adequate computer model would have to consider the full three-dimensional flow system. Golder Associates' SOLTR code, while three-dimensional, has an overly simple algorithm for simulating the passage of radionuclides from one aquifer to another above or below it.

Part 1 of this report describes the algorithm developed to provide SOLTR with an improved capability for simulating interaquifer transport. Part 2 of the report will describe the implementation and testing of the algorithm in the SOLTR code. Part 2 will be prepared as part of FY80 work.

After review of a number of alternatives, the algorithm selected is essentially a numerical convolution process. At each time step, the rate at which solute is entering and leaving each of the leakage "pipes" is computed using the convolution algorithm. These rates are then maintained as boundary conditions during the next time step.

The convolution algorithm was tested on a number of cases and proved to be an accurate, computationally efficient procedure.

TABLE OF CONTENTS

	<u>Page</u>
PREFACE	
1.0 INTRODUCTION	1
2.0 ALGORITHMS CONSIDERED	
2.1 FINITE ELEMENT/FINITE DIFFERENCE ALGORITHM	2
2.1.1 Fully Coupled Equations	2
2.1.2 Uncoupled Equations	3
2.1.3 Partially Coupled Equations	3
2.2 ANALYTIC (CLOSED FORM) SOLUTION	3
3.0 THE ALGORITHM SELECTED	5
3.1 THE GREEN'S FUNCTION	6
3.2 THE INPUT HISTORY FUNCTION	9
3.2.1 Changes in Pipe Flow Velocity	11
3.2.2 Radioactive Decay	11
4.0 TESTING THE ALGORITHM	13
4.1 THE TEST COMPUTER PROGRAM (OUTFL)	13
4.2 RESULTS	13
5.0 DISCUSSION	16
REFERENCES	
APPENDIX I: Use of OUTFL Computer Program	
APPENDIX II: Listing of OUTFL Computer Program	

LIST OF FIGURES

FIGURE

- 1   Example of Three-Dimensional Finite Element Model
Used by AFPM and SOLTR
- 2 J_x as a Function of α/L and S
- 3   Input History Example
- 4   Interpolation Algorithm Demonstrating Storage of
Input History
- 5   Example of Computation of Stored Input History
- 6   Comparison of Results of Problems with Varying
Number of Time Slices, n
- 7   Comparison of Results of Problems with Varying
SOLTR Time Steps
- 8   Comparison of Results of Problems with Varying
Dispersivity

INTRODUCTION

In order to assess the regional transport of any radionuclides which escape from a repository, it is necessary to have a model of the regional groundwater flow system which simulates the way in which dissolved radionuclides are transported from the repository to their eventual discharge point. As part of the NRC program to develop an independent capability to assess repository safety, Golder Associates is enhancing the programs in its Groundwater Computer Package in order to improve their capability to model regional flow and solute transport systems. This report describes the development of an improved capability to simulate passage of nuclides between aquifer layers.

Figure 1 demonstrates the nature of the three-dimensional finite element models used in the AFPM code (for flow studies) and in the SOLTR code (for solute transport studies). Both of these codes make simplifying assumptions regarding flow through the barriers (aquitards) separating aquifers. The SOLTR program assumes that solute transit through aquitards is vertical and instantaneous. The assumption of vertical flow is acceptable in most cases, but the assumption of instantaneous transit is not always justifiable. In cases where the aquitard has high effective porosity, low permeability, low hydraulic gradient, or a high retardation factor, the solute transit time to cross the aquitard can be significant.

Based on the above considerations, it was decided to modify SOLTR to incorporate an improved model of interaquifer transport. Selecting a good algorithm for the calculation of interaquifer transport was essential; a poorly chosen algorithm would result in SOLTR developing the same problems of massive core storage and excessive run times which make conventional three-dimensional models unattractive.

EXAMPLE OF THREE DIMENSIONAL FINITE ELEMENT MODEL USED BY AFPM AND SOLTR

Figure 1

2.0

ALGORITHMS CONSIDERED

SOLTR assumes that interlayer transport occurs in what are essentially vertical pipes connecting nodes of different layers. While the interlayer leaks are defined on an element-to-element basis, the leakage is actually calculated at the nodes. Thus, the improved interlayer solute transport algorithm developed for SOLTR would be directly analogous to a solution of a problem with vertical seepage "pipes" connecting nodes in the different layers.

2.1 FINITE ELEMENT/FINITE DIFFERENCE ALGORITHM

One way to model the pipe transport problem is to distribute a number of node points along the pipe and generate finite element or finite difference equations for the pipe's convective/diffusive solute transport model. This pipe subsystem could be incorporated in SOLTR using either fully coupled, uncoupled, or partially coupled equations.

2.1.1 Fully Coupled Equations

In this case, the pipe equations would be simply added to the global system equations. There are two drawbacks which render this approach unacceptable: the overall size of the global equation system would become extremely large and the stability/accuracy criteria for the time integration of the pipe equations would, in general, be quite different from those of the aquifers. Coupling the two sets of equations would either require a very inefficient approach of allowing the most critical component to control the time step or would require a significant development effort to produce an explicit/implicit time integration algorithm similar to that of Narasimhan and Neuman (1976).

2.1.2 Uncoupled Equations

In this case, the pipe output would be assumed to vary relatively slowly compared to the time step used for solving the aquifer equations. At the beginning of each time step, the equations for each pipe would be solved for the rate of entry of solute at the upstream end of the pipe and the rate of discharge at the downstream end. These rates would then be held as fixed boundary values for the next time step(s) of the aquifer transport solution.

This approach has some advantages over the fully coupled approach. The equations for each pipe are simple (tridiagonal) and can be solved using an optimal time step.

2.1.3 Partially Coupled Equations

In this method, the instantaneous transit pipes in the original version of SOLTR would be kept in the global matrix. Uncoupled equations would be solved for the difference between the instantaneous and finite difference solutions for the pipe. The differences between the two solutions would be applied as correction terms in the global matrix.

2.2 ANALYTIC (CLOSED FORM) SOLUTION

An alternative uncoupled approach is to use an analytic solution for the pipe. This involves storing a time history of the input to the pipe and computing the output at any time by convolving the input history with the solution for solute output following a unit slug input (the "Green's function" or "impulse-reponse function" for the pipe), as shown in Equation [1].

$$[1] \quad F_o(t) = \int_{-\infty}^t I(\tau)J(t-\tau)d\tau.$$

Equivalently,

$$[1a] \quad F_o(t) = \int_0^{\infty} I^*(a)J(a)da \quad (a \text{ is the "age"})$$

where:

$F_o(t)$ is the rate of discharge of solute from the pipe at time t . (MT^{-1})

t is the time at which the discharge is computed. (T)

$I(\tau)$ is the rate of influx of solute into the pipe at time τ . (MT^{-1})

$I^*(a)$ is the rate of influx at time $(t-a)$. (MT^{-1})

$J(x)$ is the rate of discharge at time x due to a slug input at time zero, divided by the input mass. (T^{-1})

τ is the time at which the input occurs and spans all time prior to t . (T)

There are two drawbacks to using this method: (1) storing the time histories of the input to each pipe could require a large amount of storage and would be awkward when the SOLTR time steps were varying and (2) the numerical integration of the convolution integral for each pipe at each time step would be time-consuming and difficult to perform accurately for low-dispersion problems.

3.0

THE ALGORITHM SELECTED

The final algorithm developed is a numerical approximation of the analytic solution described in Section 2.2. It is designed to optimize the accuracy of the result while minimizing the additional computational burden. The basic structure of the algorithm is:

- The "pipe" equations are uncoupled from the aquifer transport equations, as described in Section 2.1.2.
- The time history of each pipe's input is stored only for the period $(t-2T, t)$, where t is the present time and T is the mean transit time for the pipe. It is assumed that input prior to $(t-2T)$ will all have emerged from the pipe by time t . The input history I^* is stored as a function of age (parameter a in Equation [1a]).
- The convolution integral, Equation [1a], is thus only evaluated over the period 0 to $2T$:

$$[2] \quad F(t) = \int_0^{2T} I^*(a)J(a)da$$

- The integral in [2] is further approximated as the sum, over a user-defined number of time intervals, of the products of total influx for the interval times the average value of the Green's function:

$$[3] \quad F_o(t) \approx \sum_{i=1}^n I_i J_i$$

where:

$$I_i = \int_{(i-1)\Delta t}^{i\Delta t} I(a)da$$

$$J_i = \frac{1}{\Delta t} \int_{(i-1)\Delta t}^{i\Delta t} J(a)da$$

$$\Delta t = \frac{2T}{n}$$

n = number of time slices

- At the end of each SOLTR time step the input time history for each pipe is updated.
- At the beginning of each SOLTR time step the rate of outflux from each pipe is computed using Equation [3]. (The J_1 terms are computed in advance at the beginning of the run.) The outflux rates are then held as constant boundary values for SOLTR's next step.

3.1 THE GREEN'S FUNCTION

For a pipe of unit cross-sectional area, the outflux at time x due to a slug input at time zero, divided by the mass of the slug, is (Kaufman 1978):

$$[4] \quad J(x) = \frac{L}{\sqrt{4\pi L \alpha x^3}} e^{-\left(\frac{(L - \frac{L}{T}x)^2}{\frac{4\alpha L x}{T}}\right)}$$

where:

x = the time since the slug was input. (T)

L = the length of the pipe. (L)

$T = \frac{\epsilon R_d L}{V}$ is the median transit time. (T)

ϵ = the material porosity.

R_d = the retardation factor.

V = the Darcy velocity of the fluid. (LT⁻¹)

α = the dispersivity. (L)

Define a normalized time, S , as:

$$[5] \quad S = \frac{x}{T}$$

which combines with [4] to produce a new function, J_x :

$$[6] \quad J_x(S) = j(S) \cdot x = \frac{1}{\sqrt{4\pi S \frac{\alpha}{L}}} e^{\left(\frac{-(1-S)^2}{4 \frac{\alpha}{L} S} \right)}$$

The expression for J_x given in Equation [6] is a function of two dimensionless parameters: (1) normalized time, S , and (2) α/L , which is the ratio of the dispersivity to the pipe length. Figure 2 shows J_x as a function of S and α/L .

Figure 2

J_x AS FUNCTION OF α/L AND S

Referring to Equation [3], in order to carry out the numerical convolution, the average value of J for each of n substeps in the time range 0 to $2T$ (S ranging from 0 to 2) is needed. The number of substeps, n , is a variable selected by the

user. Typically, n lies between 3 and 20. This average value of J is approximated by computing the average value of J_x of the substep and dividing by the average value of x of the substep.

Thus, for substep i , which spans from $x = (i-1)(2T/n)$ to $x = i(2T/n)$:

$$[7] \quad J_i x_i = \frac{1}{2/n} \int_{\frac{2(i-1)}{n}}^{\frac{2i}{n}} J_x(S) dS$$

where x_i is the midpoint time of the i th slice:

$$[8] \quad x_i = \left(\frac{2i-1}{2} \right) \left(\frac{2T}{n} \right) = \left(\frac{2i-1}{2} \right) \Delta t = \left(\frac{2i-1}{2} \right) \left(\frac{2\epsilon R_d L}{nV} \right)$$

What is actually stored in the computer is the array $J_i \Delta t$

$$[9] \quad J_i \Delta t = \frac{n}{(2i-1)} \int_{\frac{2(i-1)}{n}}^{\frac{2i}{n}} J_x(S) dS$$

The array, $J_i \Delta t$, is a dimensionless ratio and is a function only of n , the number of time slices selected, and α/L , the dispersivity-to-length ratio. Once the user selects n , the program generates (by numeric integration of Equation [9]) tables of $J_i \Delta t$ for a user-selected range of α/L values. Each pipe in the problem is then assigned to the $J_i \Delta t$ table generated for the value of α/L closest to that of the pipe. This technique is particularly economical of computer memory as typically, no matter how large the problem, only about 10 tables of $J_i \Delta t$ are required, each just n words long.

During the solution process, the output from a pipe is very easily obtained from Equation [3]:

$$\begin{aligned}
 [3a] \quad F_o(t) &= \sum_{i=1}^n I_i J_i = \frac{1}{\Delta t} \sum_{i=1}^n I_i (J_i \Delta t) \\
 &= \frac{Vn}{2\epsilon R_d L} \sum_{i=1}^n I_i (J_i \Delta t)
 \end{aligned}$$

3.2 THE INPUT HISTORY

As defined in Equation [3], the input history to each pipe is stored as an array of the amount of solute entering over specified time slices in the past. The total history saved spans a time equal to twice the median convective transit time for the pipe; $2T = 2\epsilon R_d L/V$. This period is subdivided into n slices of duration $\Delta t = 2T/n$, as sketched in Figure 3.

Figure 3 INPUT HISTORY EXAMPLE

It is not feasible to store in computer memory the exact input history to each pipe. Instead, the computer stores n

"boxes" for each pipe, containing the amount of solute which entered during the corresponding time slice. At the conclusion of each SOLTR time step, the average rate of influx during the step is calculated and the total mass input is put into the first box. The solute previously existing in the boxes is "pushed" further into the array (e.g., into the next box).

When the SOLTR time step is not an even multiple of the pipe's time slice, the updating of the time history of input is done in two steps. First, the solute in the boxes is simply "pushed up" by as many full time slices as exist in a SOLTR time step. The remaining fraction of a time slice is handled by transferring only a portion of the mass in each box forward into the next box. Figure 4 demonstrates the interpolation algorithm used to determine how much mass to move.

Figure 4 INTERPOLATION ALGORITHM DEMONSTRATING STORAGE OF INPUT HISTORY

Figure 5 demonstrates for a hypothetical case (where the SOLTR time step equalled $3\frac{1}{2}$ box time slices) the manner in which the input time history would be computed.

EXAMPLE OF COMPUTATION OF STORED INPUT HISTORY Figure 5

NOTE: 1 SOLTR time step equals 3-1/2 box widths.

a.) Initial condition before input.

b.) Condition at the end of first SOLTR time step.

c.) Condition at end of second SOLTR time step during which the input rate is 60% of that of first time step.

d.) Condition at end of third SOLTR time step during which the input rate is 50% of that of first time step.

As can be seen in Figure 5, when the SOLTR time step is not an integral multiple of the box time slice, there is unavoidable "artificial diffusion" created in the boxes. The amount of artificial diffusion is minimal when a reasonable number of boxes is used.

For a very quickly reacting pipe, the SOLTR time step will be greater than $2T$ and the boxes will be completely filled with the latest input value. This gives a very accurate result. The artificial diffusion inherent in the boxes is significant only when the pipe transit time is long, n is small, and the dispersivity is small compared to the pipe length.

3.2.1 Changes in Pipe Flow Velocity

For problems involving unsteady flow conditions (due to climatic changes, etc.), the velocity of flow in the pipes will change. However, provided the direction of flow in the pipe does not reverse, no change at all is required in the input history function. There is an instantaneous change in the output rate (Equation [3a]) caused by the change in T .

If the flow direction reverses the algorithm will not work correctly, although only the solute which is in transit at the time of the reversal will be affected.

3.2.2 Radioactive Decay

The input history array containing the total mass of solute which entered the pipe during the preceding n time slices is "decayed" at each SOLTR time step. Thus, the array contains the

present values of the masses. The decay is handled at the end of each SOLTR time step by:

$$[10] \quad \text{Mass}_{\text{new}} = \text{Mass}_{\text{old}} \times e^{-\lambda \Delta t}$$

where λ is the radioactive decay rate.

4.0

TESTING THE ALGORITHM

4.1 THE TEST COMPUTER PROGRAM (OUTFL)

In order to test the algorithm, a small computer program called OUTFL was developed. This program drives the subroutines which are intended to control the algorithm when it is added to the SOLTR program. The driver program and subroutines, which are listed in Appendix II, have the following functions.

<u>Name</u>	<u>Type</u>	<u>Function</u>
OUTFL	Main Program	Driving program for subroutines, performs convolution of $J_i \Delta t$ array with input history array
MBOX	Subroutine	Updates input history by "pushing" solute through the boxes
PLRT	Subroutine	Plots pipe outflux versus time
PR811	Subroutine	Goldr library routine for plotting; used by subroutine PLRT
BBTAB	Subroutine	Computes $J_i \Delta t$ array for specified α/L values
ANALS	Subroutine	Calculates analytic solution of pipe outflux for slug input

OUTFL uses the algorithm as described in Section 3 to compute the time history of outflux for a pipe due to either a slug input at time zero or to a user-defined history of the solute concentration in the incoming water. Appendix I contains notes on how to use OUTFL.

4.2 RESULTS

A series of test cases was run using OUTFL to determine the influence of varying α/L , n , and the SOLTR time step. The

analytic solution was used for comparison to determine the accuracy of the numeric solution.

The base problem consisted of determining outflux from a pipe of unit area due to a slug input. Parameters of the model were:

pipe length, $L = 100.0$
 mass of slug, $M = 100.0$
 constant velocity, $V = 10.0$
 porosity, $\epsilon = 1.0$
 retardation, $R_d = 1.0$
 dispersivity, $\alpha = 2.0$
 radioactive decay rate, $\lambda = 0.0$
 number of time slices in pipe, $n = 25$
 "SOLTR" time step = 1.25

The results of the base problem are shown in Figure 6. Also shown are the results of the case where $n = 7$. While the numeric solution of the base problem shows fair agreement with the analytic solution, when n is decreased to 7 the numeric solution produces noticeably poorer results. Note that the time axis in Figure 6 and following figures is divided by the median transit time, $\bar{T} = \epsilon R_d L / V = 10$.

The results of the base problem are repeated in Figure 7 for comparison with those of cases with varying SOLTR time steps. When the SOLTR time step equals the base width,

$$\Delta t = \frac{2\epsilon R_d L}{nV} = 0.8$$

the results are very good because no artificial dispersion is created. However, when the SOLTR time step does not equal Δt , artificial dispersion is created, as is shown by the figures of results when the SOLTR time step equals 0.125 or 1.25.

COMPARISON OF RESULTS OF PROBLEMS WITH VARYING NUMBER OF TIME SLICES , n

Figure 6

NOTE: Time step for the pipe = 0.8

--- Analytic Solution
 — Numeric Solution

COMPARISON OF RESULTS OF PROBLEMS WITH
 VARYING SOLTR TIME STEPS

Figure 7

The results of three cases with varying dispersivity are shown in Figure 8. When the dispersivity is high (thus the spreading of the pulse is large), agreement with the analytic solution is good. As dispersivity decreases, the accuracy of the numeric solution also decreases. This error can be controlled to any desired level of accuracy by increasing the number of time slices n (e.g., Figure 6).

COMPARISON OF RESULTS OF PROBLEMS WITH VARYING DISPERSIVITY Figure 8

5.0

DISCUSSION

The algorithm developed in Section 3 of this report appears to be well-suited to inclusion in the SOLTR program. It produces acceptably accurate results using a minimum of computer time and memory. Nevertheless, a SOLTR analysis using the new algorithm will take considerably more computer time than when using the instantaneous transport assumption.

The question which remains to be addressed is the extent to which a pseudo-three-dimensional method, using "pipes" for interlayer flow and transport, can adequately simulate groundwater flow and solute transport in large sedimentary basins.

Incorporating the algorithm into SOLTR is projected as part of FY80 work by Golder Associates. Part 2 of this report will describe the results.

REFERENCES

Kaufman, A. M., 1978, Waste Pipe Calculus: Lawrence Livermore Laboratory, UCID 17953.

Neuman, S. P., and Narasimhan, T. N., 1976, Mixed Explicit-Implicit Iterative Scheme for Diffusion Type Problems, Part I, Theory: Int. J. Num. Methods Eng.

APPENDIX I
USE OF OUTFL COMPUTER PROGRAM

INPUT TO OUTFL

The program OUTFL was written to allow the user to specify the type of problem (slug input or input history) and the type of output desired. It also allows multiple problem input, as the user can specify several pipe dispersivity values and SOLTR time step values for analysis.

Two types of problems are allowed: slug input and input history. When slug input is desired, the user specifies the mass of the slug, the velocity of the fluid in the pipe (held constant for slug input problems), and the SOLTR time step value. The mass is then input into the pipe at time = 0. It is pushed through the input array boxes every SOLTR time step.

When the input history option is desired, the user specifies the concentration of the fluid entering the pipe at the beginning of the SOLTR time step, the length of the following SOLTR time-step (variable with this option), and the velocity of the fluid during the following time step (also variable). The program will automatically set at time = 0 the initial concentration to zero and the initial velocity to the first input velocity value.

The user can specify which output options are desired. The user has the option of producing printer plots of the pipe outflux with time and of producing the analytic solution for comparison with the numeric solution (this option is only available for problems with slug input and decay = 0.0).

If the same problem is to be run with changing dispersivity or SOLTR time step values, these values can all be input and the problem run again without having to restart. However, for every new dispersivity or time step value to be analyzed, the input mass or concentration history (card 6) must be input again.

Card 1: Retardation factor, porosity, pipe length, radionuclide decay rate, I, J, K, Idebug (4F10.3, 4I5)

If

I = 1   Printer plots of the pipe output will be performed (only if user has subroutine PR811 otherwise set I = 0). Default: I = 0.

J = 1   Analytic solution for slug input will be performed (only if K = 0). Default: J = 0.

K = 0   Input will be an initial slug of mass at $t = 0$. Default: K = 0.

K = 1   A user-defined input concentration history will be provided.

Idebug = 1   Box values of the $J_i \Delta t$ array will be printed (for debugging purposes). Default: Idebug = 0.

Card 2: NALPH, NBOX, NADAT, NTDAT, NHIS, NINT (6I5)

NALPH = Number of dispersivity values for which $J_i \Delta t$ will be determined and stored.

NBOX = Number of substeps the time range 0 to $2T$ is divided into. There will be NBOX $J_i t$ values stored for each dispersivity value.

NADAT = Number of dispersivity values for which the basic problem will be analyzed. Usually NADAT = 1.

NTDAT = Number of SOLTR time step values for which the basic problem will be analyzed; usually NTDAT = 1. It is ignored if K = 1.

NHIS = For slug input problems, this value is the number of outflux values desired in the printout. The total time range of the outflux is $(NHIS-1) \times SOLTR$ time step.

= For input history problems, NHIS is the number of input history data cards (Card 6).

NINT = The number of divisions per substep in the numerical integration of J_x . The larger the value of NINT, the more accurate is the array $J_i \Delta t$. Large NINT values increase the cost of running OUTFL. A good initial value to use is $NINT = 10$.

Card 3: Dispersivity values for which the $J_i \Delta t$ array will be determined and stored (NALPH x F5.0)

Card 4: Dispersivity values for which the basic problem will be analyzed (NADAT x F5.0)

Card 5: "SOLTR" time step values for which the basic problem will be analyzed (NTDAT x F5.0). Omit if $K = 1$.

Card 6: Slug input option (NADAT x NTDAT cards)
Mass of slug, velocity of fluid in pipe (2F10.0)

Input history option (NHIS x NADAT x NTDAT cards)
SOLTR time-step, concentration of fluid, velocity of fluid (3F10.0)

APPENDIX II
LISTING OF OUTFL COMPUTER PROGRAM

```

PROGRAM OUTFL (INPUT,OUTPUT,TAPE6=OUTPUT)
REAL LNGT,LAMBDA,MASS
LOGICAL PLOT,ANSOL,SLUG
DIMENSION BTAB(50),PARR(50),ALPH(50),BTABLE(50,50),DEP(500),
* TIME(500),ADAT(20),TDAT(20)
DATA DEP/250*0.0/
SLUG=PLOT=ANSOL=.FALSE.
READ 110,RD,POR,LNGT,LAMBDA,I,J,K,IDEBUG

C
C THESE VALUES WILL EVENTUALLY ARRIVE FROM SOLTR.
C INPUT RETARDATION FACTOR,POROSITY, PIPE LENGTH, DECAY RATE,
C INDICATORS FOR PLOT,ANSOL, SLUG, AND IDEBUG
C
IF(I.NE.0)PLOT=.TRUE.
IF(J.NE.0)ANSOL=.TRUE.
IF(K.EQ.0)SLUG=.TRUE.

C
READ 120,NALPH,NBOX,NADAT,NTDAT,NHIS,NINT
NALPH=NUMBER OF ALPHA VALUES, NBOX=NUMBER OF BOXES DESIRED
NADAT=NUMBER OF ALPHA VALUES TO BE ANALYSED, NTDAT=NUMBER OF SOLTR
C TIME STEPS FOR ANALYSIS
C NHIS=NUMBER OF HISTORIES TO BE COMPUTED, NINT=NUMBER OF
C BTABLE INTEGRATION STEPS PER NBOX
C
WRITE 160
C DEFAULT VALUES
RATIN=0.
CONC=0.
JK=0
DT=1.
VEL=-9.999E-23
IF(NTDAT.LE.0)NTDAT=1
IF(NADAT.LE.0)NADAT=1
IF(NBOX.LE.0)NBOX=15
IF(NINT.LE.0) NINT=60/NBOX
READ 130,(ALPH(I),I=1,NALPH)

C
C ADAT IS THE ALPHA VALUES TO BE ANALYSED
C TDAT IS THE SOLTR TIME STEP VALUES TO BE ANALYSED
C
READ 130,(ADAT(I),I=1,NADAT)
IF (SLUG)READ 130,(TDAT(I),I=1,NTDAT)

C
C CALL BTAB(NBOX,NINT,NALPH,ALPH,LNGT,BTABLE,IDEBUG,DELT)
DELT=DELT*POR*RD*LNGT
DO 500 MM=1,NADAT
AL=ADAT(MM)
IF(.NOT.SLUG)TDAT(1)=1.
DO 500 MM=1,NTDAT
DT=TDAT(MM)
C DT NOW DEFINED FOR SLUG INPUT
C
C
A=AL/LNGT
IF(.NOT.SLUG) GO TO 20
READ 140,MASS,VEL
VEL2=VEL
CONC=MASS/(VEL*DT)
RATIN=CONC*VEL
20 WRITE 100,AL,RD,POR,LNGT,LAMBDA
WRITE 150,NHIS,NBOX,NALPH,NADAT,NINT
WRITE 80
NN=NALPH-1
IF(NN.EQ.0) ALPH(2)=ALPH(1)
NN=MAX0(NN,1)
DO 30 I=1,NN
ADIF=(ALPH(I)+ALPH(I+1))/(2.*LNGT)
30 IF(A.LE.ADIF) GO TO 40
I=NALPH
40 CONTINUE
C
DO 50 J=1,NBOX
PARR(J)=0.0
BTAB(J)=BTABLE(J,I)
50 TIM = 0.
FVEL=RATIN
THASS=MASS
FULD=0.
C

```



```

DO 60 I=1,NHIS
IF(SLUG) 60 TO 61
READ 140,RT,CONC1,VEL2
IF(JK.NE.0)CONC=CONC1
IF(VEL.EQ.-9.99E-23) VEL=VEL2
55 RATIN=CONC*VEL2
MASS=RATIN*DT
FUEL=RATIN
61 PARR(NBOX+1)=DELT/VEL
C GET THE AVERAGE INFLUX
FOLD=(EXP(-LAMBDA*DT/2.))*(FUEL+FOLD)/2.
DECAY=EXP(-LAMBDA*DT)
C
C STORE MASS IN
CALL MBOX(FOLD,PARR,DT,NBOX)
PFLUX=0.0
DO 65 J=1,NBOX
PFLUX=PFLUX+PARR(J)*BTAB(J)/PARR(NBOX+1)
45 PARR(J)=PARR(J)*DECAY
JJ=NBOX+2
DEP(I+NHIS)=PFLUX
TIME(I+NHIS)=TIM
C
C COMPUTE EXACT FLUX
TIME(I)=TIM
FAC=VEL*TIM/(POR*RD)
J=I
IF(ANSOL.AND.SLUG.AND. LAMBDA.EQ.0.) CALL ANALS(AL,LNGT,TIM,J,
*FAC,THASS,DEP)
WRITE 90,TIME(I),DT,CONC,MASS,RATIN,VEL,VEL2,PFLUX,DEP(I)
C
IF(JK.NE.0:OR,SLUG) GO TO 66
CONC=CONC1
JK=1
TIM=TIM+DT
GO TO 55
C
C
56 FOLD=FUEL
MASS=0.
CONC=0.
RATIN=0.
TIM=TIM+DT
VEL=VEL2
60 IF(SLUG)FUEL=0.
IF(PLOT) CALL PLRT(NHIS,DEP,TIME)
500 CONTINUE
C
100 FORMAT(/1X,'DISPERSIVITY = ',1PE5.2,2X,'RETARDATION = ',1PE8.2, /
*2X,'POROSITY = ',1PE8.2,2X,'PIPE LENGTH = ',1PE8.2,2X,'DECAY = ',
*1PE8.2,/)
80 FORMAT(5X,'TIME',10X,'TIME',10X,'INPUT',9X,'INPUT',8X,'INFLUX',
*7X,'VELOCITY',6X,'VELOCITY',6X,'COMPUTED',6X,'ANALYTIC'/19X,
*'STEP',10X,'CONC',10X,'MASS',9X,'RATE',8X,'PAST STEP',5X,
*NEXT STEP',6X,'OUTFLUX',7X,'OUTFLUX',/)
90 FORMAT(1X,9(1PE12.4,2X))
110 FORMAT(4F10.3,4I5)
120 FORMAT(6I5)
130 FORMAT(16F5)
140 FORMAT(3F10)
150 FORMAT(1X,'NO. OF TIME HISTORIES = ',I5,4X,'NO. OF BOXES'
*, ' = ',I5,4X,'NO. OF ALPHA VALUES, BOXES WILL BE COMPUTED FOR'
*, 'R = ',I5,1X,'NO. OF ALPHA VALUES TO BE ANALYSED = ',I5,4X,
*'NO. OF ITERATIONS PER BOX = ',I5,/)
160 FORMAT('1',50(' ')/4X,'SOLUTE TRANSPORT IN FLOW THROUGH'
*' PIPES'/X,50(' ')/4X)
STOP
END

```

```

SUBROUTINE MBOX(FVEL,PARR,DT,NBOX)
C   DT IS THE SOLTR TIME STEP
C   DELT IS THE TIME STEP OF RIAB
C
C   DIMENSION PARR(S0)
C   DELT = PARR(NBOX+1)
C   DRAT = DT/DELT
C   IBUMP=DRAT
C   IBUMP IS THE INTEGER PART OF THE BOXES
C   TO BE MOVED
C
C   (IBUMP = MIN0(IBUMP,NBOX)
C   IF (IBUMP.EQ.0) GO TO 25
C   IF (IBUMP.EQ.NBOX) GO TO 15
C
C   MOVE THE WHOLE BOXES
C   J=NBOX
C   K=NBOX-IBUMP
C   DO 10 M=1,K
C   PARR(J)=PARR(J-IBUMP)
C   J=J-1
C
C   DO 20 M=1,IBUMP
C   PARR(M) = FVEL*DELT
C   IF (IBUMP.EQ.NBOX) RETURN
C
C   NOW WE HAVE MOVED ALL THE COMPLETE MEMORY BOXES
C   MOVE THE FRACTIONAL PART
C
C   DDT=DRAT-IBUMP
C   IF (DDT.EQ.0.) RETURN
C   W1=.5+DDT/2.
C   W2=1.-W1
C   FLUXA=FVEL*DELT*DDT
C   J=IBUMP+1
C   D=PARR(NBOX-1)
C   IF (J.GT.(NBOX-1)) GO TO 40
C   NN=NBOX-1
C   DO 30 K=J,NN
C   FLUXD=(W1*PARR(K)+W2*PARR(K+1))*DDT
C   IF (FLUXD.GT.PARR(K)) FLUXB=PARR(K)
C   PARR(K)=PARR(K)+FLUXA-FLUXB
C   FLUXA=FLUXB
C
C   FLUXB=(W1*PARR(NBOX)+W2*(2.0*PARR(NBOX)-D))*DDT
C   IF (FLUXB.GT.PARR(NBOX)) FLUXB=PARR(NBOX)
C   PARR(NBOX)=PARR(NBOX)+FLUXA-FLUXB
C   RETURN
C   END

```

```

SUBROUTINE PLRT(NHIS,DEP,TIME)
C THIS SUBROUTINE PLOTS ANALYTIC FLUX VS. COMPUTED FLUX
  DIMENSION DEP(500),TIME(500),ISET(6,2),AXIS(3,4),TITLE(8),
  *LEGEND(5,5)
  DATA ISET /1,0,1,2,'A','A'.0,0,1,2,'C','C'/
  DATA AXIS/'LINEAR','TIME',' ','LINEAR','OUTFLUX ',' ','6%' */
  DATA TITLE/'PREDICTED ','VS OBSERVE','D FLUXES '/
  DATA LEGEND/'A ANALYTIC',4%' ','C COMPUTED',19%' */
  XSC=YSC=0.
  XASC=YASC=0.
  ISET(2,1)=NHIS
  ISET(2,2)=NHIS
  CALL PRB11(0.,.TRUE.,.TRUE.,,AXIS,XSC,YSC,XASC,YASC,FUNC,2,ISET,
  * TIME,DEP,TITLE,LEGEND,6,IERR)
  IF (IERR.NE.0) STOP 'PLOT'
  RETURN
END

```

```

C      SUBROUTINE BBTAB(NBOX,NINT,NALPH,ALPH,LNGT,BTABLE,DEBUG,DELT)
C
C
C      THIS SUBROUTINE RETURNS BTABLE FOR EACH "BOX" WHICH CONTAINS
C      AVERAGE OUTFLOW FOR UNIT SLUG INPUT DURING A TIME STEP DIVIDED
C      BY CUMULATIVE TIME IN TRANSIT DIVIDED BY DELT
C
C      ALPH CONTAINS DISPERSIVITY VALUES FOR WHICH BTABLE ARRAYS
C      WILL BE GENERATED
C      DIMENSION ALPH (NALPH),BTABLE(50,NALPH)
C      REAL LNGT
C
C      DO 10 IA=1,NALPH
C      AL=ALPH(IA)/LNGT
C      IF (AL) 4,5,6
C
C      ALPH IS POSITIVE
C      DELT=2.0/NBOX
C      DT=DELT/NINT
C      CTR=0.
C      OUTFIX=0.
C
C      COMPUTE THE VALUE OF EACH BOX IN BTABLE
C      NN=NBOX+1
C      DO 9 IDOX=1,NN
C      SUM=OUTFIX
C      IF(IDOX.EQ.(NBOX+1))DT=NINT*DT
C      IF(IDOX.EQ.(NBOX+1))SUM=SUM*NINT
C      INTEGRATE THE BOX IN NINT STEPS
C
C      DO 8 I=1,NINT
C      TIM=TIM+DT
C      OUTFIX=DT*EXP(-((1.-TIM)**2)/(4.*AL*TIM))/SQRT(12.5664*AL
C      *TIM)
C      SUM=SUM+OUTFIX
C      OUTFIX=OUTFIX/2.0
C      SUM=SUM-OUTFIX
C      BTABLE(IDOX,IA)=SUM*2.0/(DELT*(2*IDOX-1))
C      BTABLE(NBOX,IA)=BTABLE(NBOX,IA)+BTABLE(NBOX+1,IA)/NINT
C      GO TO 10
C
C      ALPH IS ZERO
C      DO 7 IDOX=1,NBOX
C      BTABLE(IDOX,IA)=0.
C      J=NBOX/2
C      IF((J*2).EQ.NBOX)GO TO 11
C      BTABLE(J+1,IA)=1.0
C      GO TO 10
C      11 BTABLE(J,IA)=0.5
C      BTABLE(J+1,IA)=0.5
C      GO TO 10
C
C      ERROR - ALPH IS NEGATIVE
C      4 WRITE(6,20) ALPH
C      10 CONTINUE
C      IF(DEBUG.EQ.0) RETURN
C      WRITE(6,30) ((BTABLE(I,J),I=1,NBOX),J=1,NALPH)
C      RETURN
C
C
C      20 FORMAT('***ERROR IN SUBROUTINE BBTAB - INVALID ALPHA
C      * VALUE. INPUT VALUE ISX ',1PE12.3)
C      30 FORMAT(/1X,'BTABLE= ',1PE12.3)
C      END

```

```

SUBROUTINE ANAL5(AL,LNGT,TIM,I,F,AVEL,DEP)
C
C THIS SUBROUTINE CALCULATES THE ANALYTIC VERSION OF PIPE
C FLOW WITH A SLUG INPUT
C
  DIMENSION DEP(50)
  REAL LNGT
  IF (TIM) 4,5,6
C
C LESS THAN ZERO
C
  4  WRITE 10
    DEP(I)=-99999.
    RETURN
C
C EQUALS ZERO
C
  5  DEP(I)=0.
    RETURN
C
C GREATER THAN ZERO
C
  6  IF (AL) 4,8,7
  7  DEP(I)=(LNGT*AVEL/SQRT(12.5664*AL*F*TIM**2))
    * EXP(-(LNGT-F)**2/(4.*AL*F))
    RETURN
C
  8  DEP(I)=999999.
    RETURN
C
  10 FORMAT(//1X, '*** ERROR IN COMPUTING THE ANALYTIC FLUX')
    END

```