One-Dimensional Transport with Inflow and Storage (OTIS): A Solute Transport Model for Small Streams

A Final Report

in cooperation with the

United States Geological Survey

Water Resources Division

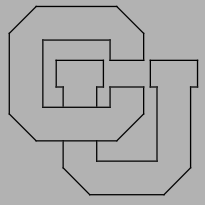
Robert L. Runkel
Robert E. Broshears

September 30, 1991

CENTER FOR ADVANCED DECISION SUPPORT FOR WATER AND ENVIRONMENTAL SYSTEMS

DEPARTMENT OF CIVIL ENGINEERING

UNIVERSITY OF COLORADO



Acknowledgments

This work was carried out as part of a co-operative agreement between CU-CADSWES and the United States Geological Survey (USGS) in Denver, Colorado. Funding for the work was provided by the USGS Toxic Substances Hydrology Program. The authors wish to thank Kenneth Bencala (USGS-Menlo Park), Briant Kimball (USGS-Salt Lake City), Diane McKnight (USGS-Denver) and Steven Chapra (CU-CADSWES) for their review of this document and their input during software development. A big 'thank you' also goes out to Sandra Braitberg and Judith Bond for their help in formatting this document and the 'Equations from Hades'.

Authors:

Robert L. Runkel
CU-CADSWES
2945 Center Green Court
Suite B
Boulder, Colorado 80301

Robert E. Broshears
U.S. Geological Survey
Water Resources Division
Mail Stop 415
Denver Federal Center
Lakewood, Colorado 80225

Abstract

One-Dimensional Transport with Inflow and Storage (OTIS): A Solute Transport Model for Small Streams

This report details the development and use of a solute transport model incorporating **O**ne-Dimensional **T**ransport with **I**nflow and **S**torage (OTIS). The model simulates the one-dimensional transport of multiple conservative substances in advective surface water systems. Non-conservative substances are also simulated through the specification of a first-order decay rate. In addition to providing dynamic and steady-state solution options, the model allows for the analysis of solute behavior under both steady and unsteady flow regimes.

The conceptual model presented herein is founded on the principle of transient storage. Transient storage or 'dead zone' models divide the stream into two distinct zones. The first zone, the stream channel, represents the main channel that is typically considered in conventional water quality models. The mechanisms of advection, dispersion, lateral inflow, transient storage and first-order decay are explicitly considered in this zone. A second zone, the storage zone, represents the recirculating pools and underflow channels in which portions of the transported solute become isolated from the main channel. Only the processes of storage and first-order decay are considered in this immobile storage zone.

The solute transport code is based on an earlier program by Bencala and Walters (1983). Several modifications and additions have been made to the earlier code. First, the governing differential equations have been decoupled. This decoupling, with the introduction of efficient numerical algorithms, has led to a dramatic reduction in computer runtime. Additional improvements include enhanced input/output options, multi-solute and steady-state simulation capabilities, and consideration of first-order decay. Finally, the code may be used in conjunction with watershed-scale routing models to analyze solute transport under unsteady flow conditions.

Detailed sections of the report provide descriptions of model theory, input and output requirements, program structure and sample applications. The report concludes with a description of model limitations and corresponding implications for future work. Several appendices supplement these main portions of the text.

One-Dimensional Transport with Inflow and Storage (OTIS):
A Solute Transport Model for Small Streams

Table of Contents **Page**

Table of Contents (continued) **Page**

One-Dimensional Transport with Inflow and Storage (OTIS): A Solute Transport Model for Small Streams

1.0 INTRODUCTION

This report details the work undertaken from October 1, 1990 to September 30, 1991 at the Center for Advanced Decision Support for Water and Environmental Systems (CADSWES) under contract with the U.S. Geological Survey. The purpose of the text that follows is to describe the development and use of a solute transport model incorporating **O**ne-Dimensional **T**ransport with **I**nflow and **S**torage (OTIS).

This report describes a stand-alone solute transport code developed in Fortran-77. An interface-driven version of the code is also available (OTIS-MHMS), but it is not discussed herein as it is currently under development.

The remaining sections of the report are summarized below:

- Section 2.0 - Contributions
- Section 3.0 - Theory
- Section 4.0 - User's Guide
- Section 5.0 - Programmer's Guide
- Section 6.0 - Model Application
- Section 7.0 - Concluding Remarks
- Appendices

Section 2.0 begins the report with a description of the rationale behind model development, and the contributions of this work. Section 3.0 gives a detailed summary of the theoretical constructs underlying the solute transport code. This section includes descriptions of the conceptual watershed, the governing differential equations, and the numerical methods used within the code. Section 4.0, a User's Guide, presents the input and output requirements of the Fortran computer program. Model parameters, print options and simulation control variables are detailed in this section. A Programmer's Guide is given as Section 5.0. This section describes the subroutine structure, the installation procedure, and several programming features. Section 6.0 presents sample input and output files and several applications of the model. A final section describes the model's limitations and the corresponding implications for future work. The report concludes with numerous appendices that supplement the main portions of the text.

2.0 CONTRIBUTIONS

Transient storage simulations have been conducted for a number of years using a computer program by Bencala and Walters (1983). However, several deficiencies limit the use of this program. First, routine simulations of conservative substances are computationally expensive. Second, the code has a rigid input/output structure. Third, simulations are restricted to a single conservative solute under steady flow conditions. Finally, the unstructured nature of the computer program makes modification and maintenance a difficult task.

In response to these shortcomings, the original code has been rewritten as a set of Fortran subroutines. Some of the features of the new code are given below.

- **Efficiency**. Using the original code, simulations for the benchmark case (see Section 5.5) often require several hours of run-time on the Prime computer. As a result, USGS personnel routinely conduct simulations on an overnight basis. Using the new code, simulations for the benchmark case run in ten minutes on a Data General Aviion workstation (Series 300). This decrease in run-time is attributed to the decoupling of the governing equations (Section 3.4.5) and efficient use of the Thomas Algorithm (Section 5.4.1).
- **User Interface.** Although not described in this report, a version of the new code is available with a graphical user interface. This interface includes a spreadsheet that facilitates data entry, and several options for viewing simulation results. For the original code, input and output are conducted using conventional 'flat files'. This format requires the user to enter data in specific columns using a text editor. Visual display of output is developed using a separate graphics package.
- **Flow Regimes.** All simulations with the original code assume a steady flow regime, i.e. flowrates and cross-sectional areas are constant in time. The new code allows for the consideration of steady or unsteady flow conditions. For the case of unsteady flows, time-varying flows and areas are read from an external flow file. This file allows the solute model to be linked with a routing model such as DR_3M (Alley and Smith, 1982).
- **Steady-State Simulations.** The new code provides an option to compute steady-state solute concentrations. This option is unavailable in the original code.
- **Non-Conservative Solutes.** The new code allows for the simulation of selected non-conservative solutes through the specification of first-order decay rates. First-order decay is not considered in the original code.
- **Multi-Solute Simulations.** Only one solute may be modeled using the original code. The new code allows for the simulation of multiple solutes.
- **Structured Programming Techniques.** The original program consists of four subroutines written in Fortran-IV. The new code is comprised of over three dozen routines written in Fortran-77. The modular nature of the new code facilitates program maintenance and modification.
- **Input/Output Flexibility.** The original code only produces simulation results at the end of each stream reach. The new program supplies results at any number of arbitrary locations along the stream channel.

3.0 THEORY

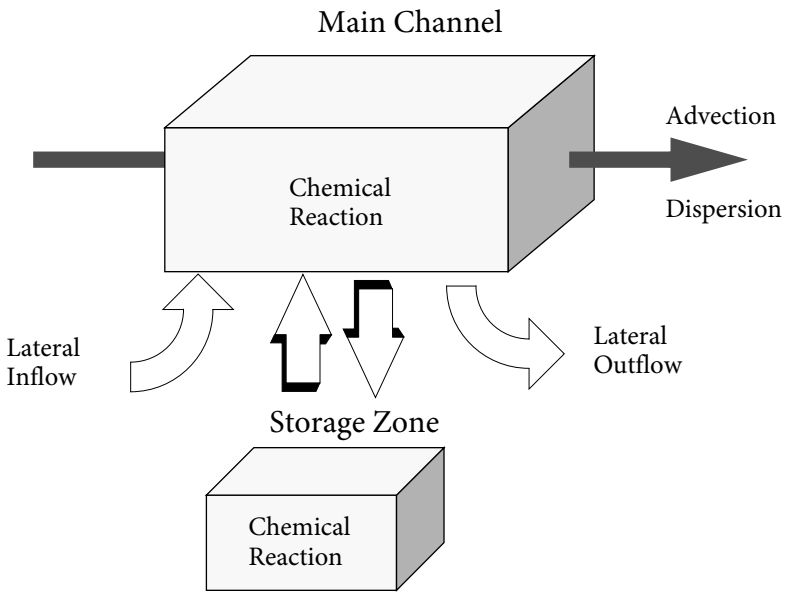
3.1 Background - Transient Storage Models

The model described herein is based on a transient storage model presented by Bencala and Walters (1983). Transient storage (or 'dead-zone') models have been formulated by a number of authors (Thackston and Krenkel, 1967; Thackston and Schnelle, 1970; Valentine and Wood, 1977; Nordin and Troutman, 1980; Jackman et al., 1984). These models add the process of transient storage to the traditional advection-dispersion equation.

Transient storage occurs in streams with zones of water that are immobile relative to the water in the main channel. Immobile zones may include pools, eddies, and water isolated behind rocks, vegetation, or other irregular bottom relief, as well as interstitial water within gravel beds and banks. As a solute pulse moves along such a stream, solute mass enters storage zones, and the concentration of the solute in the active channel is attenuated. After the main body of the pulse passes, transient storage zones become a source of solutes to the stream. This transfer of solute mass back to the active channel causes a gradual tailing of the concentration profile.

To account for transient storage, two distinct zones are defined (Figure 1). The first zone represents the main stream channel. Processes influencing solute concentrations in this zone include advection, dispersion, lateral inflow, transient storage, and chemical reaction. A second zone, the storage zone, represents all of the immobile areas described above. Advection, dispersion, and lateral inflow do not occur in the storage zone. Solute movement between the two zones is characterized by first-order mass transfer. A mathematical representation of the transient storage model is presented in the following section.

FIGURE 1
Main Channel and Transient Storage Zones



3.2 Governing Differential Equations

3.2.1 Time-variable Equations

As with most water quality models, the OTIS solute transport code is based on differential equations that are derived using a fundamental concept, *conservation of mass*. To derive the governing equations, a mass balance is required for each arbitrary stream element or '*segment*'. For the case at hand, a mass balance is developed for both the main stream and the transient storage zone. A full account of the mass balance procedure is included as Appendix A.

The following table depicts the processes that are considered in the mass balance equations. These processes are assumed to influence solute behavior in the stream channel and/or the storage zone, as indicated below.

TABLE 1
Processes Considered

Process	Stream Channel	Storage Zone
Advection	✓	
Dispersion	✓	
Lateral Inflow	✓	
Transient Storage	✓	✓
First-Order Decay	✓	✓

Consideration of the above processes in the context of the mass balance gives rise to the coupled set of differential equations derived in Appendix A. Conservation of mass for the stream and storage-zone segments is given by Equations (1) and (2), respectively:

$$\frac{\partial C}{\partial t} = -\frac{Q}{A}\frac{\partial C}{\partial x} + \frac{1}{A}\frac{\partial}{\partial x}(AD\frac{\partial C}{\partial x}) + \frac{q_{LIN}}{A}(C_L - C) + \alpha(C_S - C) - \lambda C \quad (1)$$

$$\frac{dC_S}{dt} = \alpha\frac{A}{A_S}(C - C_S) - \lambda_S C_S \quad (2)$$

where

A - stream channel cross-sectional area [meters^2]

A_S - storage zone cross-sectional area [meters^2]

C - in-stream solute concentration [mass/meters^3]

C_L - solute concentration in lateral inflow [mass/meters^3]

C_S - storage zone solute concentration [mass/meters^3]

D - dispersion coefficient [meters^2/second]

Q - volumetric flowrate [meters^3/second]

q_{LIN} - lateral inflow rate [meters^3/second-meter]

t - time [seconds]

x - distance [meters]

α - storage zone exchange coefficient [/second]

λ - in-stream first-order decay coefficient [/second]

λ_S - storage zone first-order decay coefficient [/second]

To solve Equations (1) and (2) for the general case where the parameters vary in time and space, numerical solution techniques must be employed. Numerical techniques for the solution of the time-variable equations are the topic of Section 3.4.

3.2.2 Steady-State Equations

The governing differential equations and the corresponding numerical solution techniques can be greatly simplified if a steady-state solution is desired. Steady-state conditions arise when all loading functions (i.e. boundary conditions) and model parameters are held constant or 'steady' for an indefinite period of time. Under these conditions, the system eventually reaches an equilibrium state in which net mass accumulation equals zero.

Setting the accumulation terms ($\partial C/\partial t$ and dC_S/dt) equal to zero, Equations (1) and (2) become:

$$0 = -\frac{Q}{A}\frac{\partial C}{\partial x} + \frac{1}{A}\frac{\partial}{\partial x}\left(AD\frac{\partial C}{\partial x}\right) + \frac{q_{LIN}}{A}(C_L - C) + \alpha(C_S - C) - \lambda C \tag{3}$$

$$0 = \alpha\frac{A}{A_S}(C - C_S) - \lambda_S C_S \tag{4}$$

Equation (4), the storage zone equation, is now solved directly for C_S, yielding:

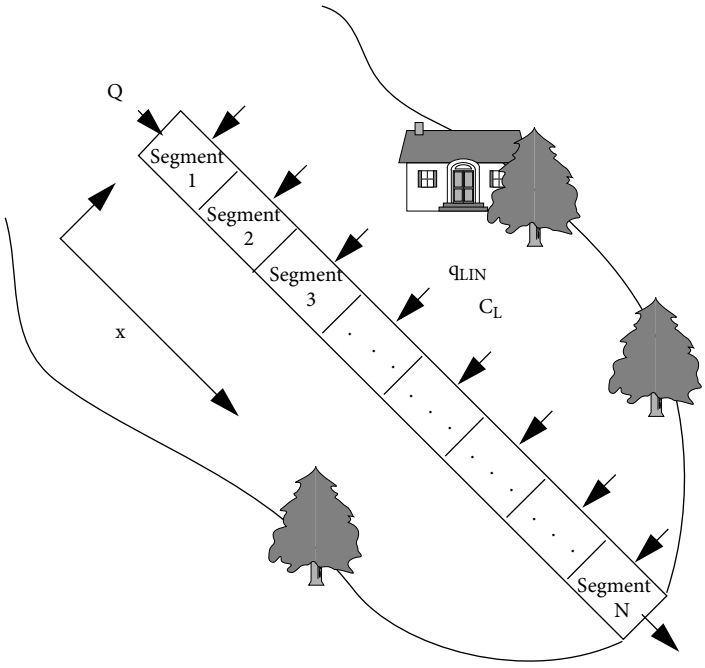
$$C_S = \left(\frac{\alpha A}{\alpha A + \lambda_S A_S}\right)C \tag{5}$$

The stream channel equation (Equation (3)), meanwhile, is solved numerically. This solution is discussed in Section 3.5.

3.3 The Conceptual Stream Channel

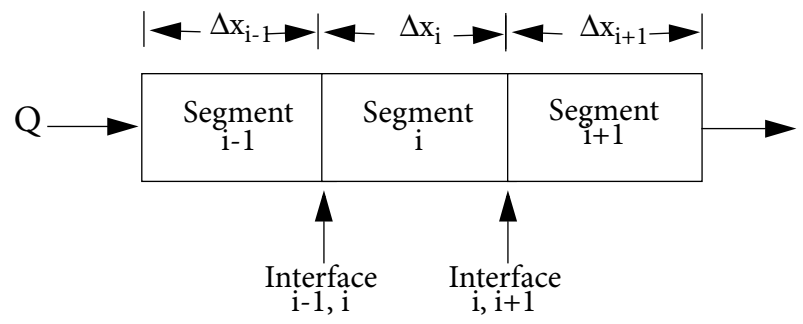
To effectively implement a numerical solution scheme, we first define the physical system. Figure 2 depicts an idealized system in which the stream is subdivided into a number of discrete units or *segments*. Each of these segments represents a control volume within which mass is conserved. Equations (1) and (2) therefore apply to each segment in the stream network.

FIGURE 2
Conceptual Watershed



Before proceeding to the next section, it is useful to introduce some additional nomenclature. Figure 3 below depicts three arbitrary segments from the stream network shown in Figure 2. Under this segmentation scheme, the subscripts *i, i-1*, and *i+1* denote concentrations and parameters at the center of each segment, while the subscripts *(i-1,i)* and *(i,i+1)* define values at the segment interfaces. The length of each segment, Δx, is also introduced in the figure. These definitions are used in Section 3.4 and Appendix B.

FIGURE 3
Segmentation Scheme



3.4 Numerical Solution - Time Variable Equations

3.4.1 Finite Differences

Due to the presence of temporal and spatial derivatives, Equation (1) is a partial differential equation (PDE). A common method of solving PDEs is to approximate the spatial derivatives, $\partial/\partial x$, using finite differences. Standard methods (e.g. Euler's Method, Runge Kutta, Crank-Nicolson) may then be implemented.

The finite difference approximations used for the spatial derivatives are given in Appendix B. Using these approximations, Equation (1) becomes:

$$\frac{dC}{dt} = L[C, Q, A, D, \Delta x] + \frac{q_{LIN}}{A_i}(C_L - C_i) + \alpha(C_S - C_i) - \lambda C_i \tag{6}$$

where

$$L[C, Q, A, D, \Delta x] = -\left(\frac{Q}{A}\right)_i\left(\frac{C_{i+1} - C_{i-1}}{2\Delta x}\right) + \frac{1}{A_i}\left[\frac{(AD)_{i,i+1}(C_{i+1} - C_i) - (AD)_{i-1,i}(C_i - C_{i-1})}{\Delta x^2}\right] \tag{7}$$

Note that Δx and the *i* subscripts are as defined in Section 3.3 and Figure 3. Also note that Equation (7) is for the special case of equal segment sizes ($\Delta x_{i-1} = \Delta x_i = \Delta x_{i+1}$). Appendix B may be consulted for the general case of variable segment lengths.

3.4.2 The Crank-Nicolson Method

As mentioned in the foregoing paragraphs, differential equations such as (2) and (6) may be solved using a variety of techniques. These techniques may be loosely grouped into '*explicit*' and '*implicit*' methods. In short, explicit methods project a solution using values of the dependent variable (e.g. solute concentration) from the current time level, whereas implicit methods utilize values from both current and future time levels. For additional information on implicit and explicit solution techniques, the interested reader is referred to Chapra and Canale (1988).

For reasons of accuracy, efficiency and stability, the implicit Crank-Nicolson method is used within the solute model. Several advantages of the method led to this choice. First, the Crank-Nicolson method is second-order accurate in both time and space. Second, the one-dimensional nature of the model results in the formation of a tridiagonal coefficient matrix that can be solved using an efficient LU Decomposition method, the *Thomas Algorithm*. Finally, the Crank-Nicolson method is unconditionally stable; the solution will not oscillate as the time step is increased.

The following two sections outline the application of the Crank-Nicolson approach to Equations (2) and (6). A full development of the matrix coefficients is presented in Appendix C.

3.4.3 The Stream Channel

In the Crank-Nicolson algorithm, the right-hand-side of Equation (6) is evaluated at both the current time (time *j*) and the future time (time *j+1*). In addition, the time derivative, *dC/dt*, is estimated using a centered difference approximation:

$$\frac{dC}{dt} = \frac{C_i^{j+1} - C_i^j}{\Delta t} \tag{8}$$

where

Δt - the integration time step [seconds]

j - denotes the value of a parameter or variable at the current time

j+1 - denotes the value of a parameter or variable at the advanced time

Equation (6) thus becomes:

$$\frac{C_i^{j+1} - C_i^j}{\Delta t} = \frac{G[C, C_L, C_S, \ldots]^{j+1} + G[C, C_L, C_S, \ldots]^j}{2} \tag{9}$$

where

$$G[C, C_L, C_S, Q, A, D, q_{LIN}, \Delta x, \alpha, \lambda] = L[C, Q, A, D, \Delta x] + \frac{q_{LIN}}{A_i}(C_L - C_i) + \alpha(C_S - C_i) - \lambda C_i \tag{10}$$

Because Equation (9) is dependent on the solute concentrations in the neighboring segments at the advanced time level (C_{i-1}, C_{i+1} at time *j+1*), it is not possible to explicitly solve for C_i^{j+1}. (Hence we have an 'implicit' method.) We can, however, rearrange Equation (9) so that all of the known quantities appear on the right-hand side and all of the unknown quantities appear on the left. One exception to this rearrangement in that an unknown quantity, the storage zone concentration at the advanced time level (C_S^{j+1}), remains on the right-hand side. This exception is discussed in the Section 3.4.5. For now, rearrangement yields:

$$\left[1 + \frac{\Delta t}{2}\left(\frac{q_{LIN}}{A_i} + \alpha + \lambda\right)\right] C_i^{j+1} - \frac{\Delta t}{2} L[C, Q, A, D, \Delta x]^{j+1} = C_i^j + \frac{\Delta t}{2}\left(G[C, C_L, C_S, etc]^j + \frac{q_{LIN}}{A_i} C_L^{j+1} + \alpha C_S^{j+1}\right) \tag{11}$$

This, in turn, may be simplified by grouping terms:

$$\boldsymbol{E}_i C_{i-1}^{j+1} + \boldsymbol{F}_i C_i^{j+1} + \boldsymbol{G}_i C_{i+1}^{j+1} = \boldsymbol{R}_i \tag{12}$$

where

$$\boldsymbol{E}_i = -\frac{\Delta t}{2A_i\Delta x}\left(\frac{Q_i}{2} + \frac{(AD)_{i-1,\,i}}{\Delta x}\right) \tag{13}$$

$$\boldsymbol{F}_i = 1 + \frac{\Delta t}{2}\left(\frac{(AD)_{i-1,\,i} + (AD)_{i,\,i+1}}{A_i\Delta x^2} + \frac{q_{LIN}}{A_i} + \alpha + \lambda\right) \tag{14}$$

$$\boldsymbol{G}_i = \frac{\Delta t}{2A_i\Delta x}\left(\frac{Q_i}{2} - \frac{(AD)_{i,\,i+1}}{\Delta x}\right) \tag{15}$$

$$\boldsymbol{R}_i = C_i^j + \frac{\Delta t}{2}\left(G[C, C_L, C_S, \ldots]^j + \frac{q_{LIN}}{A_i}C_L^{j+1} + \alpha C_S^{j+1}\right) \tag{16}$$

Equations (13) through (16) are for the special case of equal segment sizes. Appendix C may be consulted for the general case of variable segment lengths.

After developing Equation (12) for all of the segments in the stream network, we arrive at a set of linear algebraic equations. These equations must be solved simultaneously to obtain the in-stream solute concentration, C^{j+1}, in each of the stream segments. A hypothetical system of equations representing a five-segment network is shown below.

$$\begin{bmatrix} \boldsymbol{F_1} & \boldsymbol{G_1} & & & \\ \boldsymbol{E_2} & \boldsymbol{F_2} & \boldsymbol{G_2} & & \\ & \boldsymbol{E_3} & \boldsymbol{F_3} & \boldsymbol{G_3} & \\ & & \boldsymbol{E_4} & \boldsymbol{F_4} & \boldsymbol{G_4} \\ & & & \boldsymbol{E_5} & \boldsymbol{F_5} \end{bmatrix} \begin{bmatrix} C_1^{j+1} \\ C_2^{j+1} \\ C_3^{j+1} \\ C_4^{j+1} \\ C_5^{j+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \boldsymbol{R}_3 \\ \boldsymbol{R}_4 \\ \boldsymbol{R}_5 \end{bmatrix} \tag{17}$$

Systems of equations such as the one shown as Equation (17) may be efficiently solved using the *Thomas Algorithm*. A complete description of the Thomas Algorithm is given in Section 5.4.

3.4.4 The Storage Zone

We now apply the concepts presented in the preceding section to the storage zone equation (Equation (2)). This results in the following Crank-Nicolson equation:

$$\frac{C_S^{j+1} - C_S^j}{\Delta t} = \frac{\left(\alpha\frac{A}{A_S}(C - C_S) - \lambda_S C_S\right)^{j+1} + \left(\alpha\frac{A}{A_S}(C - C_S) - \lambda_S C_S\right)^{j}}{2} \tag{18}$$

In contrast to the stream equation, Equation (18) may be solved explicitly for the variable of interest, C_S^{j+1}. This yields:

$$C_S^{j+1} = \frac{(2 - \gamma - \Delta t \lambda_S) C_S^j + \gamma (C^j + C^{j+1})}{2 + \gamma + \Delta t \lambda_S} \tag{19}$$

where

$$\gamma = \frac{\alpha \Delta t A}{A_S} \tag{20}$$

This explicit formulation is of use in the following section.

3.4.5 Decoupling of the Stream and Storage Zone Equations

At first glance, Equations (12) and (19) appear to be a set of coupled equations due to the presence of an unknown quantity, C_S^{j+1}, on the right-hand side of Equation (12). (Recall from Section 3.4.3 that the right-hand side is to contain only known quantities.) This coupling suggests an iterative solution technique whereby Equations (12) and (19) are solved in sequence until some desired level of convergence is obtained. This iterative process is inefficient in that (12) and (19) must be solved more than once for each time step.

Fortunately, these equations can be decoupled by noting the explicit form of Equation (19). Examining (19) we see that the storage concentration is a function of two known quantities, C^j and C_S^j, and one unknown quantity, C^{j+1}. Substituting Equation (19) into Equation (16), we arrive at a new expression for **R**:

$$\boldsymbol{R}_i = C_i^j + \frac{\Delta t}{2}\left(G[C, C_L, C_S, \ldots]^j + \frac{q_{LIN}}{A_i} C_L^{j+1} + \alpha\left(\frac{(2 - \gamma - \Delta t \lambda_S) C_S^j + \gamma (C^j + C^{j+1})}{2 + \gamma + \Delta t \lambda_S} \right) \right) \tag{21}$$

Although **R** still contains an unknown quantity, Equation (21) is a much more convenient expression. The unknown quantity is now C^{j+1}, a variable that already appears on the left-hand side of Equation (12). We can therefore move the term involving C^{j+1} to the left side of (12), creating new expressions for **F** and **R**:

$$\boldsymbol{F}_i = 1 + \frac{\Delta t}{2}\left(\frac{(AD)_{i-1,\,i} + (AD)_{i,\,i+1}}{A_i \Delta x^2} + \frac{q_{LIN}}{A_i} + \alpha\left(1 - \frac{\gamma}{2 + \gamma + \Delta t \lambda_S} \right) + \lambda \right) \tag{22}$$

$$\boldsymbol{R}_i = C_i^j + \frac{\Delta t}{2}\left(G[C, C_L, C_S, \ldots]^j + \frac{q_{LIN}}{A_i} C_L^{j+1} + \alpha\left(\frac{(2 - \gamma - \Delta t \lambda_S) C_S^j + \gamma C^j}{2 + \gamma + \Delta t \lambda_S} \right) \right) \tag{23}$$

Because **R** now involves only known quantities, Equation (12) can be solved independently for the in-stream solute concentration, C^{j+1}. Having solved (12), the storage zone equation (Equation (19)) becomes a function of three **known** quantities, C_S^j, C^j, and C^{j+1}. We have thus decoupled the governing Crank-Nicolson expressions.

3.5 Numerical Solution - Steady-State Equations

As with the time-variable equations, we approximate the spatial derivatives in Equation (3) via finite differences. These approximations are given in Appendix B. The resultant algebraic equation for the stream channel is given by:

$$0 = L[C, Q, A, D, \Delta x] + \frac{q_{LIN}}{A_i}(C_L - C_i) + \alpha(C_S - C_i) - \lambda C_i \tag{24}$$

The stream channel equation's dependence on the storage concentration can be eliminated by substituting Equation (5) into the above. This yields:

$$0 = L[C, Q, A, D, \Delta x] + \frac{q_{LIN}}{A_i}(C_L - C_i) - \frac{\alpha\lambda_S A_S}{\alpha A + \lambda_S A_S} C_i - \lambda C_i \tag{25}$$

Since Equation (25) is dependent on the solute concentrations in the neighboring segments (C_{i-1}, C_{i+1}), it is not possible to explicitly solve for C_i. We can, however, rearrange Equation (25) so that the unknown quantities, C_{i-1}, C_i and C_{i+1}, appear on the left-hand side. This rearrangement yields:

$$\boldsymbol{E}_i C_{i-1} + \boldsymbol{F}_i C_i + \boldsymbol{G}_i C_{i+1} = \boldsymbol{R}_i \tag{26}$$

where **E**, **F**, **G** and **R** are developed in Appendix D. The in-stream solute concentrations are determined by solving a system of equations analogous to that shown as Equation (17).

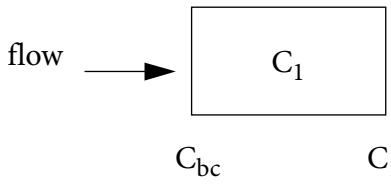
3.6 Boundary Conditions

Two boundary conditions must be specified to solve second-order differential equations such as Equation (1). For the case of a one-dimensional stream channel, these boundary conditions are applied at the upstream and downstream boundaries. A thorough discussion of these boundary conditions is given in Appendix B.

3.6.1 Upstream Boundary Condition

To implement the finite difference scheme described in Appendix B, an upstream boundary condition must be defined. This condition is shown in Figure 4, where the concentration at the upstream boundary is fixed as C_{bc}. Note that C_{bc} is a time-varying forcing function that is set by the user. User-specification of the upstream boundary condition is discussed in Section 4.5.2.

FIGURE 4
Segment 1 - Upstream B.C.



3.6.2 Downstream Boundary Condition

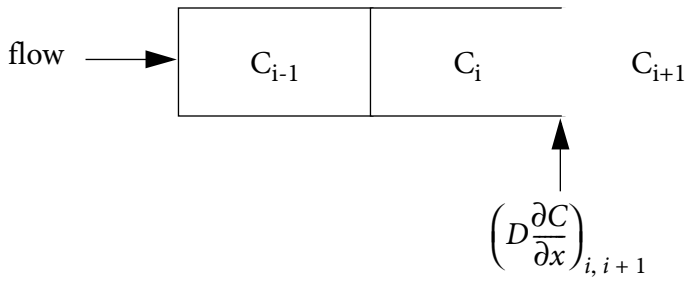
In contrast to the upstream condition, the downstream boundary condition is not a fixed concentration, but rather a fixed dispersive flux. To implement the downstream boundary condition, a dispersive flux is defined at the interface between segments *i* and *i+1*. Here the subscript *i* refers to the last segment in the stream network, while *i+1* refers to a fictitious segment adjacent to the last segment. The dispersive flux is defined as:

$$\left(D\frac{\partial C}{\partial x}\right)\Bigg|_{i,\,i+1} = DSBOUND \tag{27}$$

where DSBOUND is a user-supplied value for the dispersive flux. Note that if DSBOUND is set equal to zero, Equation (27) is a *zero gradient boundary condition* (Arden and Astill, 1970). This implies that the concentration in segment *i* is equal to that in segment *i+1*. Due to the potential error introduced by this equality, the modeled stream network should extend well beyond the region of interest (for example, see Section 6.1).

Application of the downstream boundary condition is shown in Figure 5 and discussed in Appendix B. User-specification of DSBOUND is discussed in Section 4.5.2.

FIGURE 5
Downstream Boundary Condition



4.0 USER'S GUIDE

Now that we have outlined a conceptual basis for the solute transport code, it is time to direct our attention to some operational issues. This section provides the details needed to execute the Fortran-77 computer program.

4.1 Applicability

Originally developed for use in small mountain streams, the OTIS solute transport model may be used for any stream or river in which one-dimensional transport can be assumed. As described by Fischer et al. (1979), one-dimensional analysis is valid for systems in which solute mass is uniformly distributed over the stream's cross-sectional area. Although absolute uniformity rarely occurs in nature, it is a reasonable assumption for streams of small to moderate width and depth.

The model is capable of simulating the transport of conservative substances in one-dimensional streams and rivers. Certain non-conservative substances may also be simulated through the specification of a first-order decay or production rate. The physical processes of advection, dispersion, lateral inflow and transient storage are explicitly considered in the development of the underlying mass balance equations.

4.2 Program Features

Several simulation options are available in the solute transport code. These options, described below, allow the model to be used in a flexible manner.

4.2.1 Simulation Modes: Dynamic and Steady-State

The solute transport code may be used to solve either the dynamic equations of Section 3.2.1 or the steady-state equations of Section 3.2.2. The following paragraphs describe these two simulation modes. A full description of the relevant input parameters is presented in Section 4.5.

The dynamic simulation option is selected when the user is interested in determining the time-variable nature of the transported solute. This option requires the user to specify a time-variable loading function (via the upstream boundary conditions), an integration time step (input variable TSTEP) and a time interval for the printing of results (input variable PSTEP).

As described in Section 3.2.2, the steady-state equations are applicable when the long-term response of the system to constant loading conditions is of interest. This steady-state option is invoked by setting the integration time step, TSTEP, to zero.

4.2.2 Conservative and/or Non-Conservative Substances

Another flexible feature of the code is the ability to model conservative or nonconservative substances. Conservative substances are subject to the physical processes of advection, dispersion, lateral inflow and transient storage. One additional process, first-order decay (or production), is considered when non-conservative substances are modeled.

To model a conservative substance, the first-order decay coefficients for the main channel and the storage zones are set to zero. These coefficients, denoted by the input variables LAMBDA and LAMSTOR, are described in Section 4.5.

Although non-conservative substances may undergo complex chemical, physical and biological reactions, consideration of these processes is beyond the scope of work described herein. Non-conservative substances are therefore modeled using first-order decay (or production) coefficients. These coefficients represent a one-way loss (gain) of mass from (to) the system.

To simulate a non-conservative substance, the main channel and storage zone input variables, LAMBDA and LAMSTOR, are set accordingly. Positive coefficient values are specified if the substance is subject to first-order decay, while negative values are used to specify rates of first-order production. Note that first-order decay coefficients may be calculated using half-life values that are commonly found in the literature:

$$\Lambda = \frac{0.693}{t_{1/2}} \tag{28}$$

where

Λ - first-order decay rate [/second] for the main channel or storage zone

$t_{1/2}$ - half life [seconds]

4.2.3 Flow Regimes

In addition to the foregoing options, the model has the ability to simulate transport under a variety of flow conditions. This feature is especially significant, because an accurate simulation relies on a realistic description of the hydrologic flow regime. To this end, the model considers spatially (uniform/non-uniform) and temporally (steady/unsteady) varying flows. For a review of the various flow regimes, the reader is referred to Henderson (1966).

User specification of the governing hydrologic regime is summarized in the table below. The first two options shown in the table are relatively straightforward as they involve a steady flow regime. Under this regime, model parameters such as lateral inflow and cross-sectional area remain constant with respect to time. This 'steadiness' results in a fairly simple set of user input files.

In contrast to steady flow, the third option in Table 2, unsteady flow, involves model parameters that vary in time. This option is typically selected when the solute model is used in conjunction with a watershed-scale hydraulic routing model. Such a routing model would provide time-varying lateral inflows, channel flowrates and cross-sectional areas for use as input to the solute model. The input format used to couple the two models is given in Section 4.5.5.

TABLE 2
Flow Regimes

Flow Regime	User Specifications
Steady, Uniform Flow	Set the lateral flow terms, QLATIN and QLATOUT, to zero. Also set the change in flow indicator, QSTEP, to zero.
Steady, Non-Uniform Flow	Set the lateral inflow terms, QLATIN and QLATOUT, to the desired values. Set the change in flow indicator, QSTEP, to zero.
Unsteady, Non-Uniform Flow	Prepare the *flow file* using flow, lateral inflow and area data that are generated by a suitable routing model (e.g. DR_3M). Set the change in flow indicator, QSTEP, equal to the output time step of the routing model.

4.3 Conceptual Stream Channel, Revisited

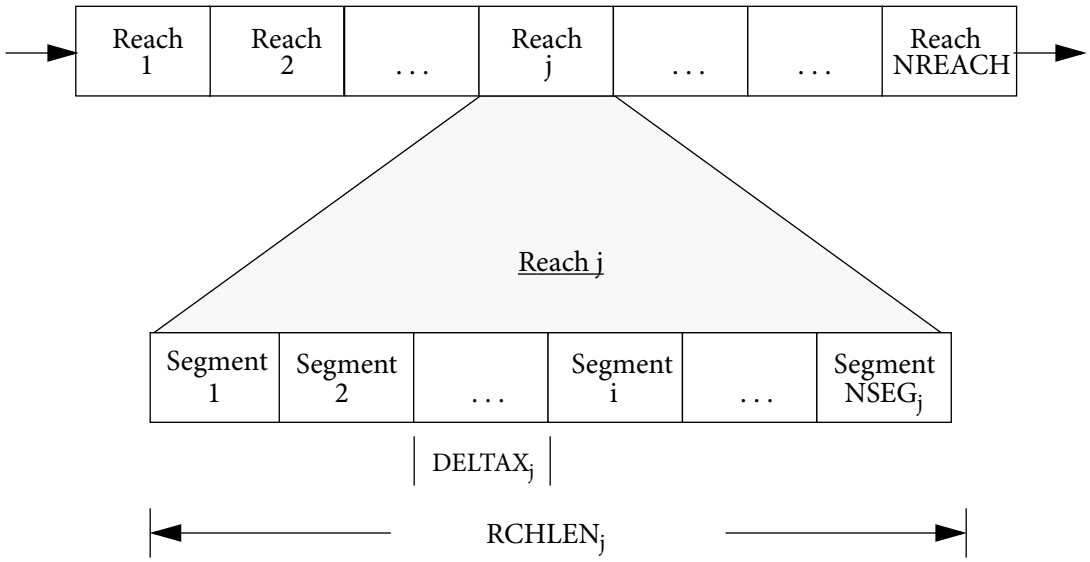
Before giving a detailed description of the model's input requirements, it is useful to define some of the program variables in terms of the conceptual watershed. In the text that follows, all user-supplied input parameters are printed using uppercase characters.

Figure 6 depicts the stream network as a series of *reaches*. For our purposes, a *reach* is defined as a continuous distance along which model parameters remain constant. A reach, for example, will have a spatially constant dispersion coefficient, decay rate and lateral inflow rate. The number of reaches defined for a given system reflects both its inherent variability and the availability of data. A spatially uniform stream may be modeled using a single reach, while a stream with a well-characterized variation in channel properties may be simulated using several reaches. The number of reaches in the modeled system is specified by the NREACH input parameter.

Each reach is subdivided into a number of computational elements or segments. Each segment represents a control-volume over which the governing mass-balance equations apply. For a given reach, there are NSEG segments of length DELTAX. Note that DELTAX is determined from the reach length, RCHLEN, and the number of segments:

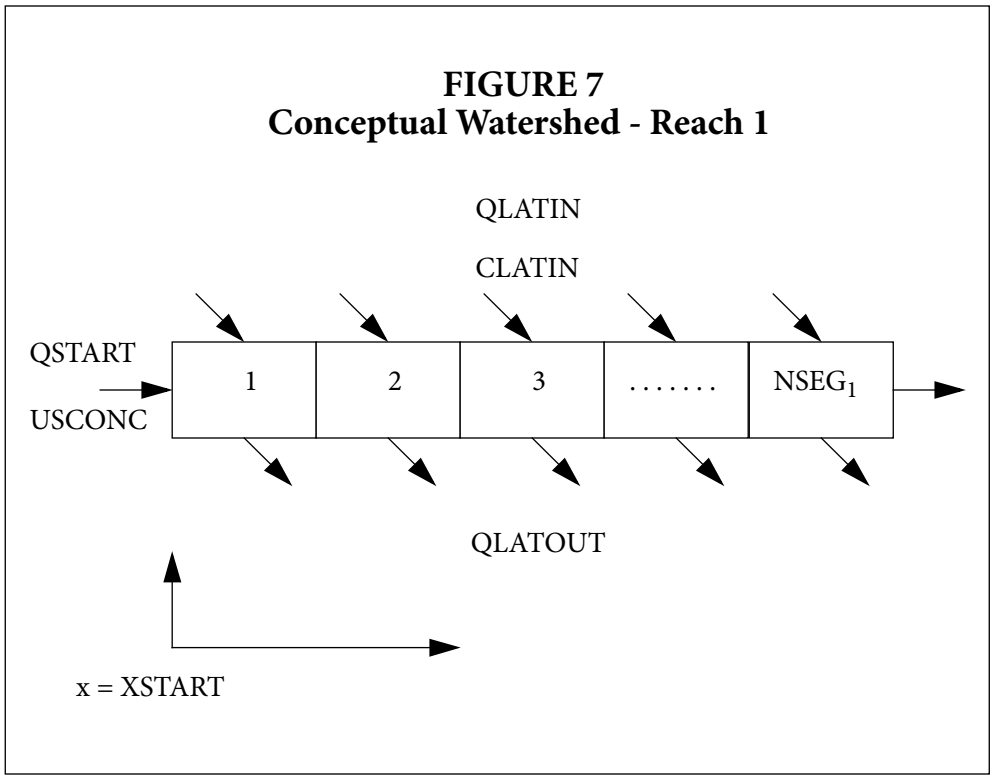
$$DELTAX = \frac{RCHLEN}{NSEG} \tag{29}$$

FIGURE 6
Conceptual Watershed - Reaches and Segments



Additional program variables are shown in Figure 7. Here we see the first reach in the stream network and its relationship to some of the required input variables. Because this reach begins at the upstream boundary of the system, we define an incoming flowrate, QSTART, and its associated solute concentration, USCONC. USCONC is the upstream boundary condition, C_{bc}, discussed in Section 3.6.1. The program variable denoting the starting stream distance, XSTART, also applies at the upstream end of reach one. Note that these three variables apply only to the first reach of the stream network.

The remaining program variables shown in the figure, QLATIN, CLATIN and QLATOUT, are specified for each reach in the modeled system (note that for unsteady flows, these variables may not necessarily correspond to specific reaches - see Section 4.5.5). The lateral inflow rate, QLATIN, represents the flow entering the channel via overland flow, interflow and groundwater discharge. This additional flow carries a solute concentration, CLATIN. The final variable, QLATOUT, is a lateral outflow term representing a loss of water from the stream channel. Both QLATIN and QLATOUT are specified on a per unit length basis (i.e. in meters^3/second-meter).



4.4 Input/Output Structure

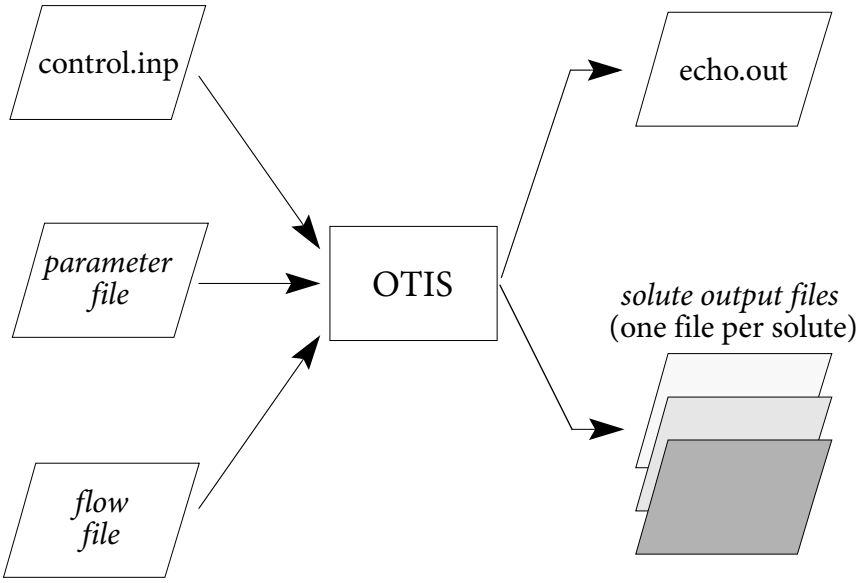
4.4.1 Input Files

The input/output structure of the solute transport code is depicted in Figure 8. Three input files are used to specify model parameters, print options and simulation control variables. A brief description of each of these files is given here. For a full description of the input files and program variables, see Section 4.5.

The first input file, '*control.inp*', specifies the number of simulations, the filenames of the remaining input files and the names of the output files. Unlike the other input filenames, '*control.inp*' is hardcoded; its name cannot be set by the user. It is important to note that the specification of filenames under the UNIX operating system is case specific, i.e. there is a difference between upper and lower case characters. The filename, 'CONTROL.INP', for example, is not equivalent to 'control.inp'.

A second input file, the *parameter file*, sets simulation options, boundary conditions and model parameters that remain constant throughout the run. The final input file, the *flow file*, contains model parameters that could potentially vary during the course of the simulation (e.g. volumetric flowrate, stream cross-sectional area). The filenames of the parameter and flow files are specified by the user in the control file, *control.inp*.

FIGURE 8
Input/Output Diagram



4.4.2 Output Files

Also shown in Figure 8 are the output files created by the solute transport code. Upon completion of a model run, the file '*echo.out*' contains a listing of ("echo of") all of the user-specified simulation options and model parameters. This file also contains any error messages generated during program execution. A sample listing of *echo.out* is given in Appendix E.

In addition to the echo file, the model creates one output file for each solute. These *solute output files* contain a time-series of solute concentrations at various locations along the stream. The filenames of the output files are specified in *control.inp*.

4.5 Input Format

Three input files must be assembled prior to program execution. In the sections that follow, each input file is described in terms of a set of '*record types*'. Within each of these record types, one or more '*fields*' is used to specify various input parameters. In general, record types refer to lines in the input file, while fields correspond to specific columns within each record.

After describing the record types, a sample input file is given. These sample files show the basic structure of each input data set. Of particular interest is the way in which some record types are used repeatedly within a given input file. The sample files given herein are for a dynamic simulation of two conservative substances in a hypothetical three-reach system.

4.5.1 The Control File

The control file, *control.inp*, is used to specify the number of model runs and the filenames for the various input and output files. As shown in Table 3, the control file contains four distinct record types.

TABLE 3
The Control File

Record Type	Program Variable	Format	Col.	Description
1	NRUNS	I	1-5	Number of Runs
2*	FILE	C	1-20	Filename for the *Parameter File*
3*	QFILE	C	1-20	Filename for the *Flow File*
4*	FILES	C	1-20	Filenames for the *Solute Output Files*

I - Integer
C - Character

*Notes: 1) Repeat Record Type 4 for each solute modeled

2) Repeat the block of Record Types 2-4 for each model run, i.e. types 2-4 are first specified for run one, followed by types 2-4 for run two, etc.

Record Type 1 - The Number of Runs

For a given execution of the program, the underlying simulation model may be executed more than once. For example, the user may wish to run the model using two different *parameter files*. This would be accomplished by placing the integer value '2' in one of the first five columns of record type 1.

The number of model runs, NRUNS, is set using record 1. Note that for NRUNS > 1, multiple input and output files must be specified by repeating the block of record types 2-4 for each model run.

Record Types 2-4 - Input and Output Filenames

Records 2-4 specify filenames for the input and output files. The filename for the *parameter file* is set using record type 2, while record type 3 specifies the name of the *flow file*. Finally, record type 4 specifies the names of the *solute output files*.

Although Table 3 lists four *record types*, the control file often contains more than four *records*, as some of the record types may be repeated. Record type 4, for example, is repeated for each solute (an output file is created for each solute). Furthermore, record types 2-4 are repeated for each model run (when NRUNS > 1).

Sample Control File

A sample control file is shown below. In this example, two model runs are requested (Record Type 1). For the first run, the parameter and flow files are named, '*params1.inp*' and '*flow.inp*', respectively (Record Types 2 and 3); the two output files are dubbed '*solute1a.out*' and '*solute2a.out*' (Record Type 4, repeated). For the second run, there is a different parameter file (*params2.inp*) but the same flow file; output is directed to the files *solute1b.out* and *solute2b.out*.

FIGURE 9
Sample Control File

```
2
params1.inp
flow.inp
solute1a.out
solute2a.out
params2.inp
flow.inp
solute1b.out
solute2b.out
```

4.5.2 The Parameter File

The *parameter file* specifies print options, boundary conditions and the model parameters that remain constant throughout the run. The format of the parameter file is given in Tables 4-7. The parameter file is created using the 17 record types discussed below.

Record Type 1 - Simulation Title

The first record in the parameter file is a simulation title of up to 80-characters. This title is printed as part of the echo output file.

Record Types 2 & 3 - Print Parameters

The print option and the print step are set using record types 2 and 3. Record type 2 indicates the hydrologic region of interest. If a value or '1' is entered, results are printed for the main channel only. Results for both the main channel and the storage zone are printed if a '2' is entered.

Following the specification of the print option, record type 3 sets the time interval at which results are printed, the print step. If, for example, results are needed every 15 minutes, a value of 0.25 hours is entered for the print step. The actual print step and the requested print step may differ if the requested step is not an even multiple of the integration time step. In this case, the program internally sets the print step equal to the nearest multiple of the integration time step.

Record Types 4-6 - Time Parameters

The next step in constructing the parameter file is to enter the appropriate time parameters. The first such parameter, the integration time step (TSTEP), is set in record type 4. Great care should be taken when setting the integration time step for the Crank-Nicolson solution procedure (See Section 3.4). Multiple model runs may be required to determine a time step that yields an accurate solution. As discussed in Section 4.2.1, the model's steady-state solution scheme is employed if TSTEP is set to zero. The time-variable equations are solved when TSTEP is greater than zero.

The remaining time parameters, TSTART and TFINAL, are set via record types 5 and 6. The input variable TSTART denotes the simulation start time, while TFINAL specifies the end of the simulation.

Record Type 7 - Distance at the Upstream Boundary

The input parameter XSTART indicates the distance at the upstream boundary of the stream network. This distance is relative to an arbitrary one-dimensional coordinate system. During program execution, XSTART is utilized to calculate the distances at various locations downstream. As the upstream boundary is at the beginning of the modeled area, XSTART is commonly set to 0.0 meters.

Record Type 8 - Downstream Boundary Condition

A downstream boundary condition, DSBOUND, is set using record type 8. In many modeling applications, the flux represented by the boundary condition is set to zero. The user should, however, ensure that the location of the boundary is sufficiently downstream from the nearest location of interest. A more complete description of the downstream boundary condition is given in Section 3.6 and Appendix B.

Record Type 9 - The Number of Reaches

As discussed in Section 4.3, the stream channel is divided into a number of reaches, NREACH. This parameter is set by record type 9. For each reach, spatially constant parameters are specified using record type 10.

TABLE 4
The Parameter File - Record Types 1-9

Record Type	Program Variable	Format	Col.	Units	Description
1	TITLE	C	1-80	---	Simulation Title
2	PRTOPT	I	1-5	---	Print Option
3	PSTEP	D	1-13	hours	Print Step
4	TSTEP	D	1-13	hours	Integration Time Step
5	TSTART	D	1-13	hour	Simulation Starting Time
6	TFINAL	D	1-13	hour	Simulation Ending Time
7	XSTART	D	1-13	meters	Distance at the Upstream Boundary
8	DSBOUND	D	1-13	mass/m^2-sec	Downstream Boundary Condition
9	NREACH	I	1-5	---	Number of Stream Reaches

I - Integer
C - Character
D - Double Precision

Record Type 10 - Reach Specific Parameters

This record type sets the parameters for each reach (Table 5). Unlike record types 1-9, record type 10 contains more than one program variable. The first field in the record indicates the number of segments located within the reach, NSEG. A second field specifies the length of the reach, RCHLEN. It is important to note that these two parameters define the length of each segment, DELTAX (see Section 4.3).

A third field in this record type is the dispersion coefficient, DISP. The storage zone cross-sectional area, AREASTOR, and the storage zone exchange coefficient, ALPHA, complete record type 10. Because this record type is used for the reach specific parameters, it is repeated NREACH times, i.e. once for each reach in the stream network.

TABLE 5
The Parameter File - Record Type 10

Program Variable	Format	Col.	Units	Description
NSEG	I	1-5	---	Number of Segments in Reach j
RCHLEN	D	6-18	meters	Length of Reach j
DISP	D	19-31	$meters^2/sec$	Dispersion Coefficient for Reach j
AREASTOR	D	32-44	$meters^2$	Storage Zone Area for Reach j
ALPHA	D	45-57	/second	Storage Zone Exchange Coefficient

I - Integer
D - Double Precision

Note: Record Type 10 is repeated once for each reach in the stream network (it repeats 'NREACH' times)

Record Types 11-13 - Solute Specific Parameters

The number of solutes to model, NSOLUTE, is given by record type 11 (Table 6). For each solute, it is possible to specify a first-order decay rate in each reach for both the main channel and the storage zone. The main channel decay coefficient, LAMBDA, is set in record type 12, while type 13 is used for the storage zone decay coefficient, LAMSTOR.

The specification of record types 12 and 13 depends on the number of solutes and the number of reaches. The decay coefficients for reach one are first set using record types 12 and 13. Within each of these record types, the decay field is repeated horizontally for each solute, i.e. the coefficient for solute one appears in columns 1-13, the coefficient for solute two appears in columns 14-26, etcetera. After specifying LAMBDA and LAMSTOR for each solute in reach one, record types 12 and 13 are repeated vertically for reach two. This repetition continues until the decay coefficients are set for all of the reaches. In general, there are NREACH blocks of record types 12 and 13. Each record contains NSOLUTE fields that are 13 characters wide.

Record Types 14 & 15 - Print Locations

While record types 2 and 3 deal with the timing and type of printed output, record types 14 and 15 control the locations at which results are printed. The flexible approach described here allows the user to request solute concentrations at a number of locations along the stream. Record type 14 specifies the number of print locations, NPRINT, while record type 15 declares the distance, in meters, of a given print location (PRTLOC). Record type 15 is repeated NPRINT times. Note that XSTART (record type 7) and PRTLOC must use the same coordinate system.

TABLE 6
The Parameter File - Record Types 11-16

Record Type	Program Variable	Format	Col.	Units	Description
11	NSOLUTE	I	1-5	---	Number of Solutes
12*	LAMBDA	D	1-13*	/second	In-stream First-Order Decay Rate
13*	LAMSTOR	D	1-13*	/second	Storage Zone First-Order Decay Rate
14	NPRINT	I	1-5	---	Number of Print Locations
15*	PRTLOC	D	1-13	meters	Print Location
16	NBOUND	I	1-5	---	Number of Boundary Conditions

I - Integer
D - Double Precision

*Notes: 1) If there is more than one solute, Record Types 12 and 13 repeat horizontally, i.e. the decay rate for solute one is placed in columns 1-13, the rate for solute 2 is placed in columns 14-26, etc.

2) Record Types 12 and 13 repeat for each reach in the stream network. Note that Records 12 and 13 are first given for reach one, followed by Records 12 and 13 for reach two, etc.

3) Record Type 15 repeats for each print location (it repeats 'NPRINT' times).

Record Types 16 & 17 - Upstream Boundary Conditions

The final record types in the parameter file specify the upstream boundary conditions. As shown in Table 6, the number of boundary conditions, NBOUND, is set using record type 16. In general, NBOUND equals the number of values for the upstream boundary concentration. Note that NBOUND should be set to one for a steady-state simulation.

Each upstream boundary condition is characterized by a starting time and a set of solute concentrations. The time at which a boundary condition goes into effect, USTIME, is specified in field one of record type 17 (Table 7). Field two, USCONC, denotes the solute concentration at the upstream boundary of reach one. This field is repeated horizontally for each solute modeled, i.e. columns 14-26 contain the boundary concentration for solute one, columns 27-39 for solute two, etcetera. After specifying the USTIME and USCONC, record type 17 is repeated for each change in the boundary concentrations (it is used NBOUND times).

TABLE 7
Parameter File - Record Type 17
Upstream Boundary Conditions

Program Variable	Format	Col.	Units	Description
USTIME	D	1-13	hour	Time Boundary Condition Begins
USCONC	D	14-26*	mass/meter^3	Concentration @ Upstream Boundary

D - Double Precision

*Notes: 1) If there is more than one solute, USCONC repeats horizontally, i.e. the concentration for solute one is placed in columns 14-26, the concentration for solute two is placed in columns 27-39, etc.

2) Record 17 repeats once for each boundary condition (it is used 'NBOUND' times).

Sample Parameter File

A sample parameter file is shown in Figure 10. This file specifies parameters for a single simulation, with results printed every 0.10 hours. The simulation runs using a time step of 0.01 hours. It begins at 0.0 hours and continues until 50.0 hours. The upstream boundary of the network is located at 0.0 meters and there is a no-flux downstream boundary condition. There are three reaches with variable lengths, numbers of segments, dispersion coefficients, storage zone areas and exchange coefficients. Two solutes are conservative in both the main-channel and the storage zone of each reach. Print locations are defined at 50, 150 and 200 meters. There are also three upstream boundary conditions. The boundary concentrations for both solutes are zero when the simulation begins, but increase to 2.4 and 1.2 units after 2 hours. The concentrations remain at these levels until the 35 hour mark, at which time they return to zero.

The left-hand margin of the figure includes a column denoting the record type for each line of the input file. Note that this column is not part of the file itself. From this annotation, we see how some of the record types repeat within the file. Record type 10, for example, is used once for each reach in the stream network. In a similar manner, the pair of record types 12 and 13 are repeated for each reach. Finally, record type 15 is repeated for each print location, while record type 17 is repeated for each boundary condition.

FIGURE 10
Sample Parameter File

Record Type					
1	sample input file - 3 reaches, 2 solutes				
2	1				
3	0.10				
4	0.01				
5	0.00				
6	50.0				
7	0.0				
8	0.0				
9	3				
10	25	50.00	0.02	0.05	3.0E-5
10	50	100.0	0.05	0.06	3.0E-5
10	100	200.0	0.05	0.06	5.0E-5
11	2				
12	0.0	0.0			
13	0.0	0.0			
12	0.0	0.0			
13	0.0	0.0			
12	0.0	0.0			
13	0.0	0.0			
14	3				
15	50.0				
15	150.0				
15	200.0				
16	3				
17	0.0	0.0	0.0		
17	2.0	2.4	1.2		
17	35.0	0.0	0.0		

4.5.3 The Flow File

As previously discussed, the *flow file* defines the model parameters that can potentially vary in time. These parameters include the volumetric flowrate (Q and QSTART), lateral flow rates (QLATIN and QLATOUT), stream cross-sectional area (AREA), and solute concentrations associated with the lateral inflows (CLATIN). Collectively, these parameters are known as the *flow variables*.

The format of the flow file depends on the nature of the parameters contained therein. If all of the flow variables are constant with respect to time, the flow is '*steady*', and the format given in Section 4.5.4 is used. If any of the flow variables change in time, the flow regime is considered '*unsteady*' and the format is as described in Section 4.5.5.

Note that for steady flow conditions, the flow variables are specified on a reach by reach basis. When the flow regime is unsteady, the flow variables are specified using flow locations. This distinction is shown in the following sections.

4.5.4 The Flow File - Steady Flow

When the flow variables are constant with respect to time, the '*steady*' flow option is invoked. As shown in the following sections, the specification of a steady flow file is fairly straightforward.

Record Type 1 - Change-in-Flow Indicator

The first record type in the flow file is the change-in-flow indicator, QSTEP. This record type is used to define the flow regime. As shown in Table 8, QSTEP is the time interval between changes in the flow variables. Here we are concerned with a steady flow regime in which the flow variables are constant. In this case, QSTEP is simply set to zero.

TABLE 8
Steady Flow File - Record Type 1

Program Variable	Format	Col.	Units	Description
QSTEP	D	1-13	hour	Change-in-Flow Indicator (set to zero)

D - Double Precision

Record Types 2 & 3 - Flow Variables

The remaining record types for a steady flow file are as described in Tables 9 and 10. Record type 2 sets the volumetric flowrate at the upstream boundary, QSTART. The remaining flow variables are set using record type 3. As shown in Table 10, the first three fields of this record specify the lateral flow rates (QLATIN and QLATOUT) and the cross-sectional area (AREA) for a particular reach. The fourth field, used to indicate the solute concentration of the lateral inflows (CLATIN), repeats horizontally for each solute modeled. Since the flow variables vary from reach to reach, record type 3 must be repeated for each reach in the network.

TABLE 9
Steady Flow File - Record Type 2

Program Variable	Format	Col.	Units	Description
QSTART	D	1-13	$meters^3/sec$	Flowrate at the Upstream Boundary

D - Double Precision

TABLE 10
Steady Flow File - Record Type 3

Program Variable	Format	Col.	Units	Description
QLATIN	D	1-13	m^3/sec-m	Lateral Inflow Rate for Reach j
QLATOUT	D	14-26	m^3/sec-m	Lateral Outflow Rate for Reach j
AREA	D	27-39	$meters^2$	Stream Channel Area for Reach j
CLATIN*	D	40-52*	$mass/meters^3$	Concentration of Lateral Inflow

D - Double Precision

*Notes: 1) If there is more than one solute, CLATIN repeats horizontally, i.e. the lateral inflow concentration for solute one is placed in columns 40-52, the concentration for solute two is placed in columns 53-65, etc.

2) Record type 3 repeats once for each reach in the stream network.

Sample Flow File - Steady Flow

A sample flow file for a steady flow regime is shown in Figure 11. In this example, the volumetric flow rate at the upstream boundary is fixed using record type 2. Lateral inflow occurs in reaches 2 and 3, but the inflow concentrations for both solutes equal zero. Cross-sectional areas increase from reach to reach. This flow file is for a hypothetical three-reach system.

FIGURE 11
Sample Steady Flow File

Record Type

```
1   0.00
2   6.10E-3
3   0.00      0.00   0.20   0.00   0.00
3   3.10E-6   0.00   0.25   0.00   0.00
3   1.65E-4   0.00   0.35   0.00   0.00
```

4.5.5 The Flow File - Unsteady Flow

With the allowance for time-variable flow parameters comes a dramatic increase in model complexity. As a result of this complexity, the format of the unsteady flow file differs markedly from that presented for the steady case. In addition, the unsteady flow file is much larger than its steady counterpart, as several record types must be repeated for each change in the flow variables.

The format used here is compatible with standard routing models such as DR_3M (Alley and Smith, 1982). Record types 2-8 allow for the specification of time-varying parameters at various locations along the stream channel. Due to its generic nature, computer-aided or manual reformatting of routing output files may be necessary to produce an appropriate flow file for use as input.

Record Type 1 - Change-in-Flow Indicator

For an '*unsteady*' flow regime, the flow variables are read from the flow file when changes in the flow values occur. QSTEP therefore defines the times at which the flow variables are updated. If the flow variables change every 15 minutes, for example, QSTEP is set to 0.25 hours and the variables are read at the appropriate times.

TABLE 11
Unsteady Flow File - Record Type 1

Program Variable	Format	Col.	Units	Description
QSTEP	D	1-13	hour	Time Interval Between Changes in Flow

D - Double Precision

Record Types 2 & 3 - Flow Locations

When detailed field measurements are available, it may be appropriate to specify time-varying flow variables at various locations along the stream. These values may be available as output from hydrologic or hydraulic routing models. The format of the flow file allows the user to define a number of *flow locations* at which the flow variables are specified. These flow locations often correspond to the reaches that are defined in the parameter file. (This correspondence is not required, however.) Using the data provided at the flow locations, the model linearly interpolates flow and area values (Q and AREA) for each segment in the stream network. Lateral inflow rates and concentrations (QLATIN, CLATIN) are also set using the flow locations.

Record type 2 indicates the number of flow locations, NFLOW. Record type 3 specifies the distance, in meters, of a given flow location. This record is used NFLOW times, i.e. once for each flow location.

Several requirements must be kept in mind when specifying flow locations. The requirements, listed below, are checked internally by the computer code:

- The flow locations must be entered in ascending (downstream) order, i.e. the second location must be downstream from the first, the third must be downstream from the second, etcetera.
- The first flow location must be placed at the upstream end of the stream network. This implies that $FLOWLOC_1$ equals XSTART, where XSTART is the starting location specified in the parameter file. Note that the flow specified at this first location is analogous to the QSTART parameter in the steady flow file.
- The last flow location must be placed at or below the downstream boundary.

TABLE 12
Unsteady Flow File - Record Types 2 & 3

Record Type	Program Variable	Format	Col.	Units	Description
2	NFLOW	I	1-5	---	Number of Flow Locations
3*	FLOWLOC	D	1-13	meters	Flow Location

I - Integer
D - Double Precision

*Note: Record Type 3 repeats for each flow location (it repeats 'NFLOW' times).

Record Type 4-7 - Lateral Flows, Flows and Areas

In contrast to the steady flow file, lateral flows and concentrations (QLATIN, CLATIN) do not necessarily correspond to specific reaches. In the unsteady flow file, these parameters are specified for each flow location. The value specified is used for all segments in-between the current flow location and the flow location immediately upstream (from location *j-1* to location *j*). Note that this scheme corresponds to that for the steady flow file if the flow locations are placed at the end of each reach. As stated earlier, the flow and area (Q and AREA) values at the flow locations are used to interpolate values for the segments within the network.

Lateral inflow, flow and area values are set for each flow location using record types 4-6, respectively. As shown in Table 13, the input fields in these record types (QLATIN, Q and AREA) repeat horizontally for each flow location. The values for flow location one appear in columns 1-13, while the those for flow location two appear in columns 14-26, etcetera.

Record type 7 is used to indicate the lateral inflow concentrations at each flow location. The CLATIN field repeats horizontally within record type 7 for each solute, i.e. the inflow concentration for solute one appears in columns 1-13, the concentration for solute two appears in columns 14-26, etcetera. This record type is used once for each flow location.

Note that record types 4-7 repeat for each change in the flow variables. For example, if the flow variables change every 15 minutes (QSTEP=.25 hours), these record types will repeat four times for every hour of simulation time. As a result of this repetition, the unsteady flow file may contain a large number of records.

TABLE 13
Unsteady Flow File - Record Types 4-8

Record Type	Program Variable	Format	Col.	Units	Description
4*	QLATIN	D	1-13*	$meters^2/sec$	Lateral Inflow Rate from Location j-1 to j
5*	Q	D	1-13*	$meters^3/sec$	Flowrate at Flow Location j
6*	AREA	D	1-13*	$meters^2$	Area at Flow Location j
7*	CLATIN	D	1-13*	$meters^2$	Inflow Concentration from Location j-1 to j

D - Double Precision

*Note: 1) QLATIN, Q and AREA repeat horizontally for each flow location specified, i.e. the values for flow location one are set using columns 1-13, for location two using columns 14-26, etcetera.

2) If there is more than one solute, CLATIN repeats horizontally, i.e. the lateral inflow concentration for solute one is placed in columns 1-13, the concentration for solute two is placed in columns 14-26, etc.

3) Record Type 7 is repeated for each flow location.

4) The block of Record Types 4-7 repeats for each change in the flow variables.

Sample Flow File - Unsteady Flow

A sample flow file for the case of unsteady flow appears as Figure 12. This sample file corresponds to a situation in which flow values are available every 15 minutes. As with the previous examples, this sample file is for a hypothetical three-reach system.

Record type 1 sets the time interval at which changes in flow occur (QSTEP = 0.25 hours). Four flow locations are then set via record types 2 and 3. Here we have chosen to place the flow locations at the end of each reach (see Figure 10 - Sample Parameter File). This is an arbitrary

choice of locations, since the flow locations need not correspond to the reach end points. Note that the flow locations appear in ascending or downstream order, with the first location at the upstream end of the stream network.

After defining the flow locations, record type 4 is used to set the lateral inflow rate. Channel flow and cross-sectional areas for each flow location are set by record types 5 and 6, respectively. Record type 7 is used repeatedly to specify the solute concentrations in the lateral inflows. Although not shown in the figure, record types 4-7 are repeated for every 0.25 hours (i.e. QSTEP) of simulation time.

FIGURE 12
Sample Unsteady Flow File

Record Type				
1	0.25			
2	4			
3	0.00			
3	50.0			
3	150.0			
3	350.0			
4	0.00e-0	0.00e-0	3.10e-6	1.65e-4
5	6.10e-3	6.10e-3	6.41e-3	3.94e-2
6	0.20	0.20	0.25	0.35
7	0.00	0.00		
7	0.00	0.00		
7	0.00	0.00		
7	0.00	0.00		
4				
5				
6				
7				
7	(Records 4-7 repeat for each change in flow)			
7				
7				
.				
.				
.				
.				

4.6 The Post-Processor

A simple post-processor is available for use with the OTIS solute transport code. This postprocessor reformats the *solute output files* so that they may be plotted using the Xgraph plotting utility (Harrison, 1989).

5.0 PROGRAMMER'S GUIDE

This section provides an overview of the solute transport computer code. Of special importance are the following topics:

- Simple User Modifications - Maximum Dimensions (Section 5.2.1)
- Computational Efficiency (Section 5.4)
- Program Installation (Section 5.6)

5.1 Model Development

The OTIS computer code is written in ANSI Fortran-77. This code is loosely based on an earlier program by Bencala and Walters (1990). Software development was conducted on a SUN4 workstation under the UNIX operating system (SunOS 4.0.3). The model has also been compiled and tested on a Data General Aviion workstation.

Although primarily intended for UNIX-based platforms, the code can be adapted for other operating systems such as PRIMOS, VMS or DOS. Minor program modifications may be necessary for these systems.

5.2 Include Files

As described in Section 5.7, the code consists of several small subroutines. To facilitate program modification, two '*Include*' files are used. By using include files, key program information may be shared between subroutines. This information may be modified by editing the include files rather than each of the individual routines.

5.2.1 Maximum Dimensions - *fmodules.inc*

One of the shortcomings of the Fortran computer language is the static allocation of memory. Under this allocation scheme, the size or '*dimension*' of each vector and array must be fixed prior to program execution. This requires some *a priori* knowledge of the maximum dimension for each model parameter. Selection of an appropriate size for each parameter is a difficult task, as excessively small values limit program applicability, while excessively large values waste program memory.

To minimize this shortcoming, the maximum dimensions for the entire model are defined in a single include file, '*fmodules.inc*'. Increases or decreases in the maximum dimensions are made by editing the include file and compiling the model as described in Section 5.6.

The default values for the maximum dimensions are shown in Table 14. In general, each of these dimensions corresponds to a user-supplied input variable. This correspondence is shown parenthetically in the third column of the table.

When running the model, the input variables may not exceed the maximum values. The number of print locations (see Section 4.5.2 - Record Type 14), for example, may not exceed the maximum value given by MAXPRINT. When an input value exceeds its given maximum, program execution is terminated and a error message is issued. At this point the user must increase the appropriate maximum (by editing *fmodules.inc*) and re-compile the program.

TABLE 14
Maximum Dimensions
Default Values from *fmodules.inc*

Dimension	Default Maximum	Maximum Number of ...
MAXREACH	20	Stream Reaches (NREACH)
MAXSEG	6000	Stream Segments (Σ $NSEG_j$, j=1 to NREACH)
MAXSOLUTE	5	Solutes Modeled (NSOLUTE)
MAXPRINT	30	Print Locations (NPRINT)
MAXBOUND	20	Upstream Boundary Conditions (NBOUND)
MAXFLOWLOC	20	Flow Locations (NFLOW)

5.2.2 Logical Devices - *lda.inc*

In the Fortran computer language, a unit number is assigned to each file used for input and/or output. These unit numbers, also known as logical device assignments (*ldas*), must be specified for each read and write operation. Program variables used to store the unit numbers are shared between the input and output subroutines using a Fortran Common block. This common block is defined in the include file *lda.inc*.

5.3 Error Checking

The program's input subroutines perform several tests to validate the input data. If errors are detected, an error message is issued and program execution is terminated. The error checking capabilities of the model are as follows:

- The number of reaches, NREACH, must not exceed the maximum, MAXREACH.
- The number of segments, IMAX, must not exceed the maximum, MAXSEG.
- The number of print locations, NPRINT, must not exceed the maximum, MAXPRINT.

- The number of solutes, NSOLUTE, must not exceed the maximum, MAXSOLUTE.
- The number of upstream boundary conditions, NBOUND, must not exceed the maximum, MAXBOUND.
- The number of flow locations, NFLOW, must not exceed the maximum, MAXFLOWLOC.
- The storage area, AREASTOR, must be greater than zero.
- A given print location, $PRTLOC_i$, must lie within the modeled network.
- The flow locations, FLOWLOC, must be entered in ascending (i.e. downstream) order.
- The first flow location, $FLOWLOC_1$, must be placed at the upstream boundary. The last flow location must be at or below the downstream boundary.
- The print option, PRTOPT, must equal '1' or '2'.

5.4 Efficiency

Two goals must be kept in mind when solving the solute transport equations. First, the numerical solution technique should provide an accurate approximation of the underlying differential equations. This goal is met by the Crank-Nicolson procedure, wherein the approximations are second-order accurate in both time and space (see Section 3.4). A second goal is to provide an appropriate degree of accuracy while minimizing computational expense. To this end, an optimized form of *Thomas Algorithm* is employed. This is the topic of the sections that follow.

5.4.1 Thomas Algorithm

As described in Section 3.4, application of the Crank-Nicolson procedure results in the formation of a linear system of algebraic equations:

$$[\boldsymbol{M}]\{\boldsymbol{C}\} = \{\boldsymbol{R}\} \tag{30}$$

where

[M] - tridiagonal *coefficient matrix* of dimension N by N

{C} - vector of length N representing the unknown solute concentrations

{R} - forcing function vector of length N, as given by Equation (23)

N - number of segments in the stream network

Due to the tridiagonal nature of matrix **[M]**, Equation (30) may be solved for **{C}** using an efficient technique, the Thomas Algorithm. The Thomas Algorithm belongs to a general class of techniques know as *LU Decomposition* methods. These methods decompose the coefficient matrix into two 'lower' and 'upper' diagonal matrices. This '*decomposition*' is followed by a '*substitution*' step that provides the matrix solution for **{C}**.

The primary advantage of the LU Decomposition approach is the ability to efficiently evaluate multiple right-hand side vectors (**{R}**). Because the first step, decomposition, involves only the coefficient matrix, **[M]**, it need not be repeated for each vector **{R}**. As a result, solutions for multiple right-hand side vectors may be obtained using a single decomposition step in conjunction with multiple substitution steps.

For the case at hand, there are multiple right-hand side vectors, as **{R}** is a function of the time-varying solute concentrations. Given a steady flow regime (QSTEP = 0), the coefficient matrix remains constant throughout the simulation. A decomposition step is therefore not required for each time step. This considerably reduces the number of operations required to complete a given simulation.

Application of the Thomas Algorithm is described in the following paragraphs. For a more detailed description of the method itself, see Chapra and Canale (1988).

Conservative Substances

For conservative substances, the coefficient matrix is a function of the model's physical parameters, i.e. **[M]** is not specific to a given solute. Due to this solute independence, the decomposition phase for the Thomas Algorithm is completed only once for each set of physical parameters. A substitution phase, meanwhile, is required for each solute at each time step, as **{R}** is a function of the solute concentrations at the current time level (see Equation (23)).

Note that if the physical parameters are time-invariant (QSTEP = 0), only one decomposition step is completed for the entire simulation. For unsteady flow regimes (QSTEP > 0), the decomposition step is performed for each change in the flow variables.

Non-Conservative Substances

Unlike the conservative case, the coefficient matrix for nonconservative solutes is a function of both physical and chemical parameters. Matrix decomposition is therefore required for each solute being modeled, as the values of the chemical parameters vary between solutes. As with conservative substances, the substitution step is required for each solute at each time step.

For steady flow regimes (QSTEP = 0), the number of decomposition steps equals the number of solutes. When the flow variables are time-variable (QSTEP > 0), the decomposition step must be repeated for each change in the flow variables.

Optimizing the Thomas Algorithm

As explained in the foregoing sections, the solution of Equation (30) is dependent on the types of solutes being modeled. Specifically, the coefficient matrix is solute-independent for conservative substances, while it is solute-specific for non-conservative solutes.

To take advantage of this difference, the solution scheme implemented in the model is determined by the nature of the solutes. If the simulation is entirely conservative ($\lambda = \lambda_s = 0$ for all solutes), the simulation is deemed '*conservative*' and the appropriate subroutines are called. If, on the other hand, any of the solutes are non-conservative, the simulation type is set to '*decay*' and another set of subroutines is used.

Since the conservative subroutines are somewhat more efficient than their decay counterparts, it may be advantageous to conduct separate simulations for conservative and non-conservative substances. Mixed simulations involving both solute types require the use of the decay routines for all of the solutes.

5.5 Benchmark Runs

This section presents the results from three benchmark runs. The purpose of this presentation is twofold:

- To show the relative performance of OTIS solute transport code versus the 'original' code of Bencala and Walters (1990).
- To show the computational advantages of utilizing a UNIX-based workstation, as opposed to the USGS Prime computer.

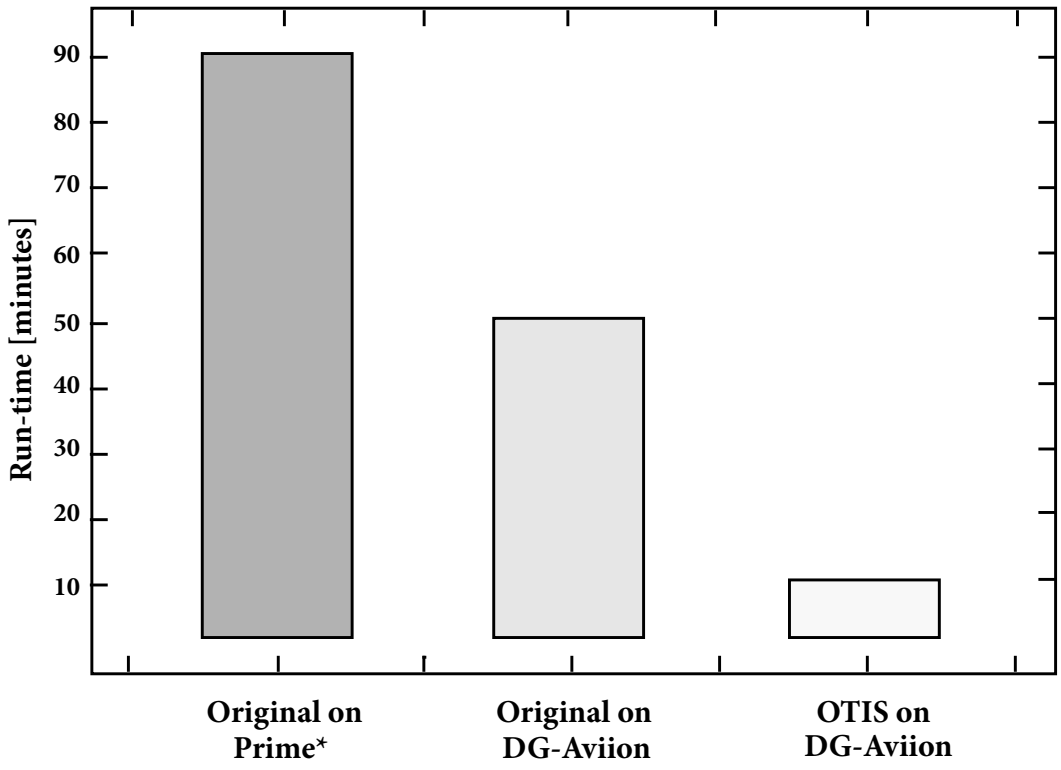
The first two benchmarks are for the original code of Bencala and Walters (1990) executed on the USGS Prime Computer and on a Data General Aviion workstation, respectively. A third and final benchmark is for the OTIS code on a Data General workstation. The simulation parameters used for the benchmark runs are shown in Table 15.

TABLE 15
Simulation Parameters for Benchmark Runs

Parameter	Value
Number of Solutes	1
Type of Solute	Conservative
Flow Regime	Steady
Integration Time Step [hours]	0.01
Simulation Period [hours]	68
Number of Segments	1900
Segment Length [meters]	1

Figure 13 depicts the results from the benchmark simulations. Two conclusions may be drawn from the figure. First, the time required for a given simulation is hardware dependent. Execution time for the original code is substantially lower on the Data General workstation, as compared to that on the Prime. Second, the OTIS code is computationally more efficient than the original code. This is due to decoupling of the governing equations (Section 3.4.5) and judicious use of the Thomas Algorithm (Section 5.4.1).

FIGURE 13
Benchmark Runs



*Note: Execution time on the Prime is approximate. Personal communication with USGS personnel indicates that the times vary from 1.5 to 5.0 hours, depending on the time-of-day.

5.6 Installation

This section describes the steps required to install and compile the OTIS solute transport code. Note that the procedure described herein is specific to the UNIX operating system.

Step 1 - Create the Directory Structure

Two directories must be created to store the computer code and the executable program. The first directory stores the files that contain the Fortran source code. This directory, known as the *source* directory, can be given any name. To create the source directory, issue the following command:

mkdir source

where 'source' is the user-supplied directory name.

The second directory stores the executable computer program and the associated input files. This subdirectory, by convention, is named '*run*'. The run directory must be created as a subdirectory under the source directory:

mkdir source/run

Step 2 - Source Code

The next step is to copy the code into the source directory. The Fortran subroutines that comprise the model are stored in separate files with '*.f*' file extensions (e.g. *main.f*, *input1.f*). Each of these files must be copied from the distribution tape (or diskette) into the 'source' directory. In addition to the subroutine files, two include files (*fmodules.inc* and *lda.inc*) and the *makefile* should be placed in the source directory.

Step 3 - Performing the 'Make'

After completing step two, position yourself in the source directory (i.e. *cd source*). The subroutines can now be compiled and linked using the *Make* utility. To do this simply type '*make*'. After the make is complete, an executable file named '*otis*' will be found in the run directory.

Step 4 - Program Execution

To execute the model, place all of the input files (see Section 4.4.1) in the run subdirectory. From this subdirectory type:

otis &

An output file for each solute is placed in the run directory during program execution.

5.7 Subroutine Structure

The OTIS solute code is composed of 47 subroutines. Figures 14-16 depict a subroutine hierarchy, wherein each box represents a Fortran subroutine. Arrows indicate which subroutines are called from a given routine. In general, subroutine calls proceed from left to right. The *main* program driver, for example, calls *ldainit, header, maininit, mainrun* and *closef*, respectively (see Figure 14). A short description of each subroutine follows.

5.7.1 Main Program Driver

The *main* routine is a driver that controls program execution. Subroutine calls are made to open files, read simulation parameters, initialize variables and conduct the simulation.

maininit

The *maininit* subroutine is called by the main program driver to obtain input parameters and initialize variables. If the steady-state option is in effect, *maininit* calls the steady-state solution routine, *sstate*.

mainrun

Based on the chemistry type (CHEM), the *mainrun* subroutine determines whether the chemistry is 'conservative' or 'decay'. An appropriate call is then made to the *timeloop* subroutine.

5.7.2 Input Routines

Several input subroutines are responsible for reading model parameters, initializing vectors, and validating the input data.

input1

This routine opens the *parameter file* and reads several parameters. Based on the value of the time-step (TSTEP), the simulation type (SIMTYPE) is set to 'Steady-State' or 'Dynamic'. The simulation type is later used to determine the appropriate solution technique.

input2

Input2 reads the time invariant parameters for each reach. The parameter vectors are filled so that each segment has its own dispersion coefficient, storage zone area, storage zone exchange coefficient, etcetera.

input3

Chemical specific parameters (e.g. decay rates) are read by *input3*. Based on the values of the first-order decay rates, the chemistry type (CHEM) is set to 'Conservative' or 'Decay'. The chemistry type is used by the *mainrun* subroutine to call the appropriate finite difference routine (see Section 5.4.1).

input4

Input4 reads the print locations (PRTLOC) and the upstream boundary conditions.

inputq

Finally, *inputq* opens the *flow file* and reads the model parameters that can potentially vary with time (e.g. flow rates and cross-sectional areas).

5.7.3 Initialization, Output and Miscellaneous Routines

The following routines initialize program variables, open files, output results and perform other miscellaneous functions.

closef

At the end of the simulation, this routine closes the *solute output files*, the *parameter file* and the *flow file*.

steady

Given the initial upstream boundary condition, this routine computes the steady-state solute concentrations. These concentrations are used as the initial conditions for the segments in the stream network.

distance

The subroutine *distance* computes the distance at the center of each segment. These distances are used to determine the exact location of the user-specified print and flow locations.

header

This routine writes a header to the *echo.out* output file. The header includes the date and time of the current run.

initialize

The *initialize* routine calls several routines to input simulation parameters and initialize program variables.

ldainit

This routine initializes the logical device assignments (*ldas*). Unit numbers (or '*ldas*') are assigned to each file used for input or output.

openfile

The *openfile* subroutine opens the *solute output files*. One file is opened for each solute.

outinit

For each print location (PRTLOC), *outinit* writes the initial conditions to the *echo.out* output file. The initial solute concentrations are also written to the *solute output files*.

output

After each print step (PSTEP), *output* writes the solute concentrations to the *solute output files*.

weights

Simulation results may be requested at various *print locations* (PRTLOC) along the stream. Linear interpolation is used to determine the solute concentration for a given print location. The subroutine *weights* calculates the interpolation weight (WT) for each print location.

5.7.4 Error Routines

As described in Section 5.3, the input routines validate the input data. If an error is found, one of three error routines (*error*, *error2* or *error3*) is called. These routines write an error message to the *echo.out* file and terminate program execution.

5.7.5 Steady Flow Routines

For the case of steady flow (QSTEP = 0), the following two subroutines are called. Subroutines specific to unsteady flow conditions are the subject of Section 5.7.6.

flowinit

The subroutine *flowinit* is called to initialize the volumetric flowrate. For each segment in the stream network, the flowrate is computed and stored in the vector Q. The flowrate at the center of each segment is given by:

$$Q_i = Q_{i-1} + \frac{(q_{LIN_{i-1}} - q_{LOUT_{i-1}})\Delta x_{i-1} + (q_{LIN_i} - q_{LOUT_i})\Delta x_i}{2} \quad (31)$$

where q_{LOUT} is the lateral outflow rate [meters^3/second-meter].

qsteady

This routine reads the flowrate at the upstream boundary (QSTART), the cross sectional areas (AREA) and the lateral flow parameters (QLATIN, QLATOUT, CLATIN).

5.7.6 Unsteady Flow Routines

When solute transport is subject to an unsteady flow regime (QSTEP > 0), the following subroutines are used.

readq

The *readq* subroutine is called each time the flow variables change (i.e. every QSTEP hours). *Readq* reads the lateral flow parameters, the cross-sectional areas and the flowrates from the *flow file*.

qainit

Using the weights provided by *qweights*, the *qainit* routine computes a flowrate and area for each stream segment. Linear interpolation is used to fill the flowrate vector, Q, and the area vector, AREA. The lateral inflow parameters (QLATIN and CLATIN) are also set for each segment.

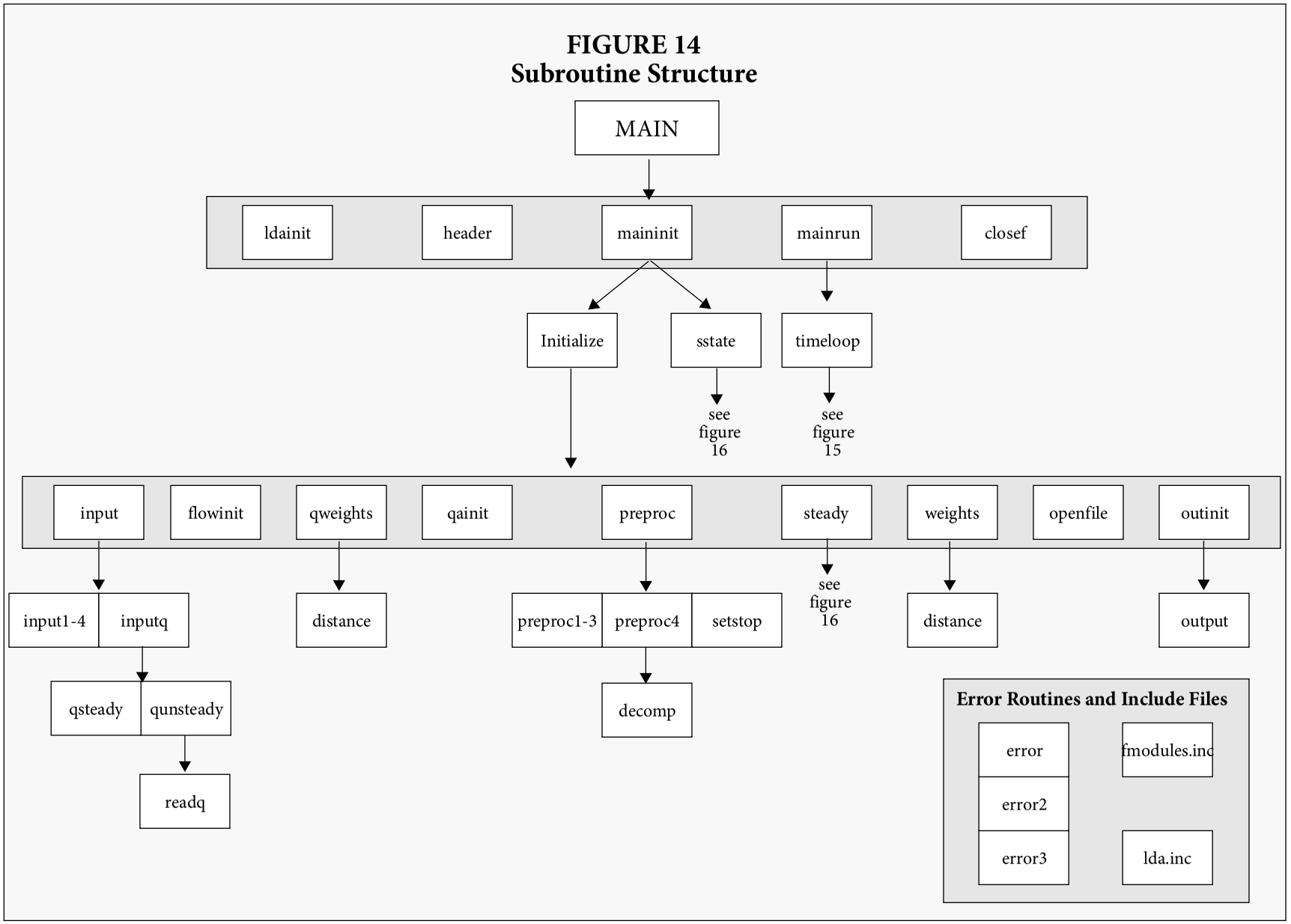
qunsteady

At the beginning of the simulation, *qunsteady* reads the flow locations and then calls *readq* to obtain the initial flow parameters.

qweights

Flow and area values are specified at various *flow locations* (FLOWLOC) along the stream. Linear interpolation is used to determine the flowrate and cross-sectional area for a given stream segment. The subroutine *qweights* calculates the interpolation weight (QWT) for each segment.

FIGURE 14
Subroutine Structure



5.7.7 Dynamic Simulation Routines

The following routines are called when a dynamic simulation is conducted.

conser

For the case of conservative transport, the *conser* subroutine computes the forcing function **{R}**. The in-stream and storage zone concentrations are then determined.

decay

When solutes undergo first-order decay, the subroutine *decay* is called to compute **{R}**. The in-stream and storage zone concentrations are then determined.

dynamic

This routine controls the dynamic simulation by updating boundary conditions and flow variables at the appropriate times.

partial

The *dynamic* subroutine is charged with the task of updating the upstream boundary conditions and the flow variables at the appropriate times. When these updates occur within the integration interval (i.e. Old Time < Time of Change < New Time), a partial time step is taken. The subroutine *partial* takes a partial time-step and then updates the boundary condition or flow variables, as appropriate.

setstop

As discussed above, the upstream boundary conditions and the flow variables change at specific times during the simulation. The *setstop* routine is responsible for setting the variable TSTOP equal to the time at which the next change occurs.

timeloop

The *timeloop* subroutine is used to control the simulation. After computing the number of time-steps within each print step, a loop is entered and solute concentrations are computed for each time-step. Simulation results are printed (via the *output* subroutine) at the appropriate times.

update

The *update* subroutine is called whenever the upstream boundary condition or the flow variables change at the beginning of the integration interval. Its task is to update the boundary conditions and/or the flow variables.

5.7.8 Pre-processing Routines

Several routines are called to pre-compute parameter groups that are constant with respect to time. This preprocessing significantly reduces the required execution time as the computations are not repeated for each time-step.

preproc1

Preproc1 computes several quantities that are functions of the segment length, DELTAX.

preproc2

Preproc2 computes two quantities that are functions of the in-stream decay coefficient and the storage zone exchange coefficient.

preproc3

Preproc3 computes two quantities that are introduced by the storage zone equation. Because these quantities are functions of the time-step, *preproc3* must be called whenever the time-step changes (e.g. when a partial time-step is taken).

preproc4

The matrix coefficients, **E**, **F** and **G**, are computed by this routine. Given these coefficients, the *decomp* routine is called to decompose the coefficient matrix. Because the matrix coefficients depend on the flow variables, *preproc4* must be called each time the flow variables change.

5.7.9 Routines for the Thomas Algorithm

Tridiagonal matrices such as the one shown in Equation (17) may be solved using the Thomas Algorithm (see Section 5.4.1). The subroutines to implement the Thomas Algorithm are discussed below.

decomp

This routine performs the decomposition phase of the Thomas Algorithm. For the simple case of conservative transport and steady flow, only one call to *decomp* is made.

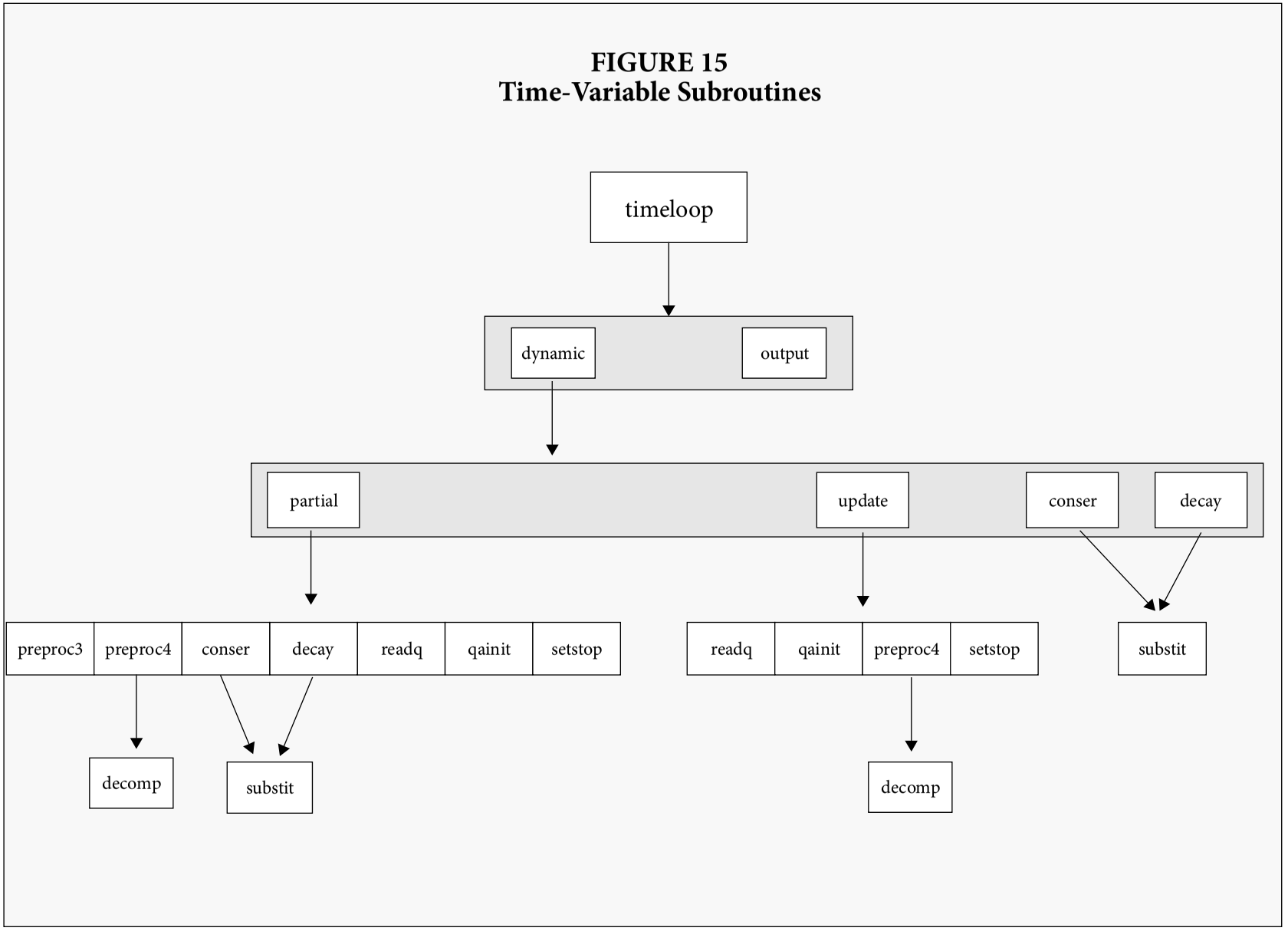
substit

The substitution phase of the Thomas Algorithm is completed by the *substit* routine.

matrix

The *matrix* subroutine conducts both phases of the Thomas Algorithm.

FIGURE 15
Time-Variable Subroutines



5.7.10 Steady-State Routines

When a steady-state solution is desired, the following routines are used.

sstate

Sstate calls the routines to compute the steady-state concentrations.

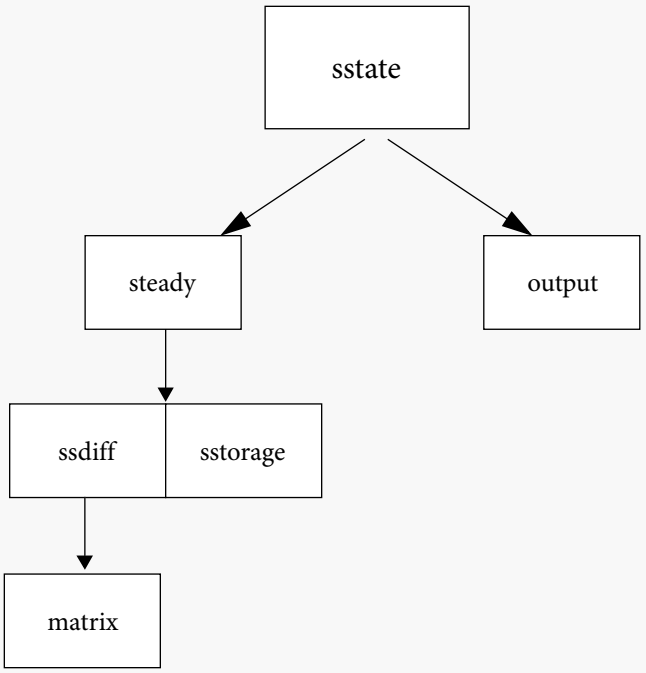
ssdiff

The steady-state finite difference routine, *ssdiff*, computes the matrix coefficients (**E**, **F** and **G**) and the forcing function (**R**) developed in Appendix D. The *matrix* subroutine is then called to compute the steady-state solute concentrations.

sstorage

This routine computes the steady-state concentration in the storage zone.

FIGURE 16
Steady-State Subroutines



6.0 MODEL APPLICATION

OTIS has been used to simulate the behavior of conservative and nonconservative solutes in a Rocky Mountain stream affected by acid mine drainage. The stream is St. Kevin Gulch, near Leadville, Colorado, where interdisciplinary studies have addressed physical, chemical, and biological processes governing the transport and fate of toxic metals in surface waters (Kimball, 1991).

In the applications shown below, OTIS is used in both dynamic and steady-state modes. These simulations are discussed in greater detail by Kimball et al. (1991). In Section 6.1, we describe a dynamic simulation of lithium chloride, an injected tracer. In Section 6.2, two steady-state simulations for iron are presented.

6.1 Application 1: Transient Profiles of a Tracer Salt

OTIS simulated concentration profiles of lithium and chloride during a tracer-dilution experiment in St. Kevin Gulch in August 1986. A 4.7 molar solution of lithium chloride was injected into the stream for 52 hours at an average rate of 27 milliliters per minute. Water samples were collected at six downstream locations.

Figure 17 shows the *control.inp* file for the simulation. The control file specifies a single run with constant parameters read from the file *params.inp* and flow information read from the file *flow.inp*. Simulated lithium and chloride concentrations are written to the files *lithium.out* and *chloride.out*, respectively.

FIGURE 17
Control File for Application 1

```
1
params.inp
flow.inp
lithium.out
chloride.out
```

The parameter file is shown in Figure 18. For this simulation, stream concentrations are printed every 0.1 hours. The time step is 0.01 hours, beginning at 13.9 hours and ending at 82.0 hours. The upstream boundary of the network is designated 0.0 meters. There is no dispersive flux at the downstream boundary.

The stream is divided into seven reaches, with reaches one through six ending at the sampling locations. The last reach extends an additional 100 meters downstream. This reach is included to reduce any error introduced by the downstream boundary approximation (see Section 3.6.2). Reaches vary in length, but each segment within the network is 1.0 meters long. Three reach-specific parameters - the dispersion coefficient, cross-sectional area of the storage zone, and the stream-storage exchange coefficient - are defined. Two solutes, lithium and chloride, are conservative throughout the network. Six print locations correspond to the sampling locations. The upstream boundary concentration is a block-shaped pulse of lithium chloride lasting from 14.0 hours until 66.0 hours.

FIGURE 18
Parameter File for Application 1

```
1   Application 1: Transient profile of a tracer salt
2   1
3   0.10
4   0.01
5   13.9
6   82.0
7   0.0
8   0.0
9   7
10   26    26.00   0.02   0.050   3.00E-05
10   458   458.00  0.02   0.050   2.00E-05
10   42     42.00  0.02   0.250   2.00E-05
10   422   422.00  0.02   0.100   1.50E-05
10   609   609.00  0.02   0.200   5.00E-05
10   247   247.00  0.02   0.200   3.00E-05
10   100   100.00  0.02   0.200   3.00E-05
11  2
12  0.00   0.00
13  0.00   0.00
12  0.00   0.00
13  0.00   0.00
12  0.00   0.00
13  0.00   0.00
12  0.00   0.00
13  0.00   0.00
12  0.00   0.00
13  0.00   0.00
12  0.00   0.00
13  0.00   0.00
12  0.00   0.00
13  0.00   0.00
14  6
15    26.00
15   484.00
15   526.00
15   948.00
15  1557.00
15  1804.00
16  3
17  12.0   0.005   0.200
17  14.0   2.363  13.360
17  66.0   0.005   0.200
```

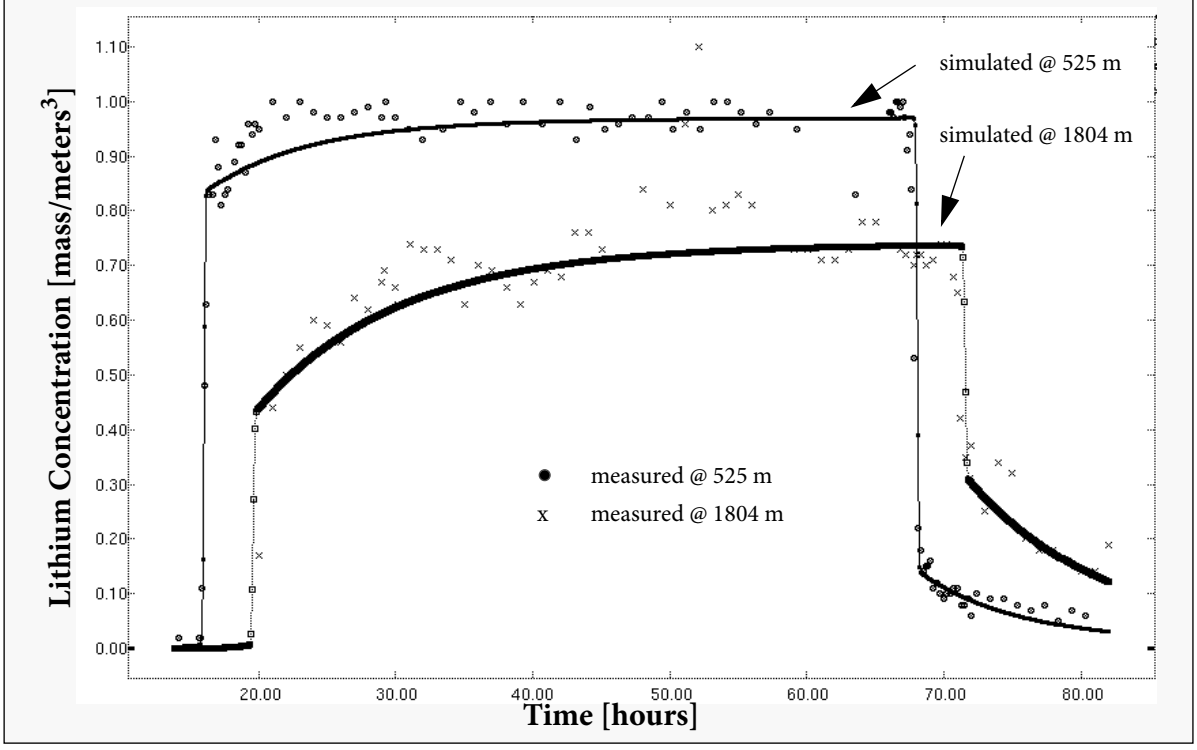
The flow file is shown in Figure 19. The zero value in the first line specifies steady-flow conditions. Flow at the upstream boundary is 6.12×10^{-3} cubic meters per second and increases downstream until the last two reaches, where there is a loss of water from the channel. Stream cross-sectional areas and concentrations of lithium and chloride in lateral inflow waters are specified.

FIGURE 19
Flow File for Application 1

```
1     0.00
2     6.12E-03
3     0.000E-00  0.000E-00  0.120  0.005  0.200
3     3.780E-06  0.000E-00  0.097  0.036  0.900
3     1.700E-04  0.000E-00  0.138  0.005  0.200
3     4.120E-06  0.000E-00  0.199  0.005  0.200
3     4.840E-06  0.000E-00  0.152  0.005  0.200
3     0.000E-00  2.030E-05  0.153  0.005  0.200
3     0.000E-00  2.030E-05  0.153  0.005  0.200
```

A post-processor program (see Section 4.6) converted the two solute output files into files with a data format suitable for display by Xgraph. This display for lithium at two sampling locations reveals a good fit between measured and simulated concentrations (Figure 20). A third output file, *echo.out*, contains a summary of the simulation parameters. This file is included as Appendix E.

FIGURE 20
Dynamic Simulation (Application 1)



6.2 Application 2: Steady-State Simulations of Iron Concentration

Two steady-state simulations for iron illustrate the reactive transport of solutes in St. Kevin Gulch. These simulations use hydraulic parameters calibrated from the simulation of lithium and chloride concentration during the tracer-dilution experiment described in Section 6.1. Iron enters St. Kevin Gulch with drainage from abandoned mine tailings that flank the stream along a zone approximately 363 to 484 meters downstream from the tracer injection site. One additional reach is delineated in this zone to simulate iron loading more accurately. Iron concentration in the drainage water is characterized by analysis of water samples collected from seeps near the tailings. Lateral inflow rates are calculated based on local dilution of the tracer salt.

The control file shown in Figure 21 specifies two runs with different parameter files but the same flow file. The two parameter files differ only in record types 12 and 13, which define first-order removal constants. In the file *feparams1.inp* (Figure 22), these constants are zero throughout the network, so that iron is modeled as a conservative substance. In the file *feparams2.inp* (Figure 23), the first-order removal constant varies from 4.0×10^{-5} to 6.0×10^{-4} per second. The flow file is shown in Figure 24. A plot of measured and simulated concentrations for the two runs indicates that iron is removed as it is transported downstream (Figure 25).

FIGURE 21
Control File for Application 2

```
2
feparams1.inp
feflow.inp
fe1.out
feparams2.inp
feflow.inp
fe2.out
```

FIGURE 22
Parameter File for Application 2 (Run 1)

```
1   Application 2:  Steady-state iron with no decay
2   1
3   0.00
4   0.00
5   0.00
6   0.00
7   0.0
8   0.0
9   8
10   26      26.00    0.02    0.050    3.00E-05
10   337   337.00    0.02    0.050    2.00E-05
10   121   121.00    0.02    0.050    2.00E-05
10    42     42.00    0.02    0.250    2.00E-05
10   422   422.00    0.02    0.100    1.50E-05
10   609   609.00    0.02    0.200    5.00E-05
10   247   247.00    0.02    0.200    3.00E-05
10   100   100.00    0.02    0.200    3.00E-05
11  1
12  0.00
13  0.00
12  0.00
13  0.00
12  0.00
13  0.00
12  0.00
13  0.00
12  0.00
13  0.00
12  0.00
13  0.00
12  0.00
13  0.00
12  0.00
13  0.00
14  7
15     26.00
15    363.00
15    484.00
15    526.00
15    948.00
15   1557.00
15   1804.00
16  1
17  0.00     0.640
```

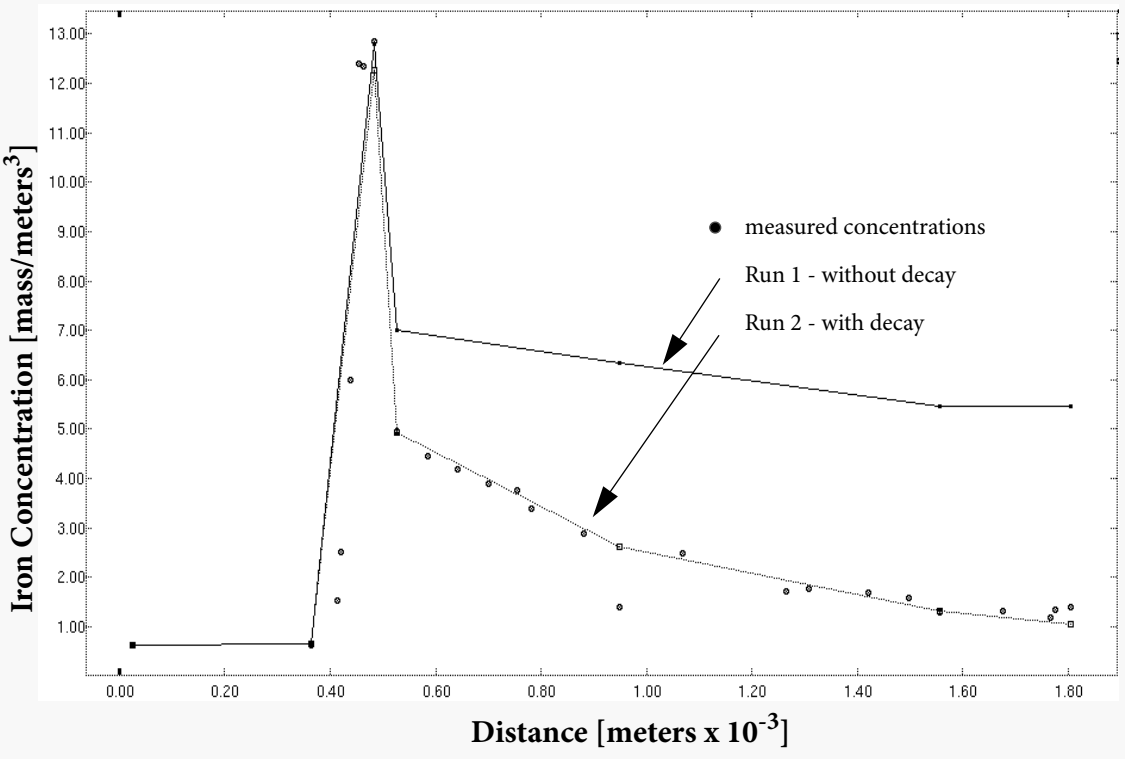
FIGURE 23
Parameter File for Application 2 (Run 2)

```
1   Application 2:  Steady-state iron with decay
2   1
3   0.00
4   0.00
5   0.00
6   0.00
7   0.0
8   0.0
9   8
10   26     26.00   0.02    0.050    3.00E-05
10   337   337.00   0.02    0.050    2.00E-05
10   121   121.00   0.02    0.050    2.00E-05
10    42    42.00   0.02    0.250    2.00E-05
10   422   422.00   0.02    0.100    1.50E-05
10   609   609.00   0.02    0.200    5.00E-05
10   247   247.00   0.02    0.200    3.00E-05
10   100   100.00   0.02    0.200    3.00E-05
11  1
12  0.00
13  0.00
12  0.00
13  0.00
12  4.E-5
13  4.E-5
12  6.E-4
13  6.E-4
12  9.E-5
13  9.E-5
12  8.E-5
13  8.E-5
12  8.E-5
13  8.E-5
12  8.E-5
13  8.E-5
14  7
15     26.00
15    363.00
15    484.00
15    526.00
15    948.00
15   1557.00
15   1804.00
16  1
17  0.00    0.64
```

FIGURE 24
Flow File for Application 2

```
1     0.00
2     6.12E-03
3     0.000E-00  0.000E-00  0.120  0.56E+0
3     3.780E-06  0.000E-00  0.097  0.56E+0
3     3.780E-06  0.000E-00  0.097  2.11E+2
3     1.700E-04  0.000E-00  0.138  0.56E+0
3     4.120E-06  0.000E-00  0.199  0.56E+0
3     4.840E-06  0.000E-00  0.152  0.56E+0
3     0.000E-00  2.030E-05  0.153  0.56E+0
3     0.000E-00  2.030E-05  0.153  0.56E+0
```

FIGURE 25
Steady-State Simulation (Application 2)



7.0 CONCLUDING REMARKS

The foregoing sections provide a thorough description of a solute transport model for small streams. As presently implemented, the model is capable of simulating the transport of multiple conservative solutes in advective surface water systems. Selected non-conservative substances may also be simulated through the specification of first-order decay rates. In addition to providing dynamic and steady-state simulation options, the code allows for the analysis of solute behavior under both steady and unsteady flow regimes.

Future work efforts should be directed toward the following tasks:

- Non-conservative solutes are simulated using first-order decay rates. This empirical approach does not begin to consider the complex chemical, physical and biological processes that affect solute transport. In order to increase its utility, the model should be modified to incorporate kinetic and/or equilibrium controlled reactions. Of particular interest is the ability to simulate chemical speciation and sorption.
- The model described herein is a stand-alone computer program developed in Fortran-77. As such, the model uses conventional 'flat files' for input and output. In order to provide a more flexible modeling environment, an interface-driven version of the code is currently under development (OTIS-MHMS). Parallel development of OTIS and OTIS-MHMS should continue.
- Section 4.3 presents a model network in which the stream is a linear series of reaches. Future work efforts could include the modifications necessary to simulate solute transport in branching stream networks.
- Further testing of the unsteady flow option (Section 4.2.3) is needed.

References

Alley, W.M. and P.E. Smith, *Distributed Routing Rainfall-Runoff Model, Version II, Computer Program Documentation, User's Manual*, U.S. Geological Survey, Water Resources Division, Open-File Report 82-344, 1982.

Arden, B.W. and K.N. Astill, *Numerical Algorithms: Origins and Applications*, Addison-Wesley, Reading MA, 1970.

Bencala, K.E. and R.A. Walters, Simulation of Solute Transport in a Mountain Pool-and-Riffle Stream: A Transient Storage Model, *Water Resources Research*, 19(3), 718-724, 1983.

Bencala, K.E. and R.A. Walters, Dynamic Solute-Transport Simulation in Dispersive Streamflow, Fortran code obtained from the USGS Prime Computer, August 2, 1990.

Chapra, S.C. and K.H. Reckhow, *Engineering Approaches for Lake Management, Volume 2: Mechanistic Modeling*, Butterworth Publishers, 1983.

Chapra, S.C. and R.P. Canale, *Numerical Methods for Engineers, Second Addition*, McGraw-Hill, New York, 1988.

Fischer, H.B., E.J. List, R.C.Y. Koh, J. Imberger and N.H. Brooks, *Mixing in Inland and Coastal Waters*, Academic Press, San Diego, 1979.

Harrison, D., *Xgraph Version 11.3.2*, University of California, Berkeley, 1989.

Henderson, F.M., *Open Channel Flow*, Macmillan, New York, 1966.

Jackman, A.P., R.A. Walters and V.C. Kennedy, Transport and Concentration Controls for Chloride, Strontium, Potassium and Lead in Uvas Creek, A Small Cobble-bed Stream in Santa Clara County, California, U.S.A., 2. Mathematical Modeling, *Journal of Hydrology*, 75:111-141,1984.

Kimball, B.A., Physical, Chemical and Biological Processes in Waters Affected by Acid Mine Drainage: From Headwater Streams to Downstream Reservoirs, in U.S. Geological Survey Toxic Substances Hydrology Program, Abstracts of the Technical Meeting Monterey, California, March 11-15, 1991, Gail E. Mallard and David A. Aronson, compilers. U.S. Geological Survey Open-File Report 91-88, Page 51, 1991.

Kimball, B.A., R.E. Broshears, K.E. Bencala and D.M. McKnight, Comparison of Rates of Hydrologic and Chemical Processes in a Stream Affected by Acid Mine Drainage, in U.S. Geological Survey Toxic Substances Hydrology Program, Abstracts of the Technical Meeting Monterey, California, March 11-15, 1991, Gail E. Mallard and David A. Aronson, compilers. U.S. Geological Survey Open-File Report 91-88, Page 71, 1991.

Nordin, C.F. and B.M. Troutman, Longitudinal Dispersion in Rivers: The Persistence of Skewness in Observed Data, *Water Resources Research* 16(1), 123-128, 1980.

Thackston, E.L. and P.A. Krenkel, Longitudinal Mixing in Natural Streams, *Journal of the Sanitary Engineering Division, ASCE,* 93(SA5), 67-90, 1967.

Thackston, E.L. and K.B. Schnelle, Predicting Effects of Dead Zones on Stream Mixing, *Journal of the Sanitary Engineering Division, ASCE*, 96(SA2), 319-331, 1970.

Thomann, R.V. and J.A. Mueller, *Principles of Surface Water Quality Modeling and Control*, Harper & Row, New York, 1987.

Valentine, E.M. and I.R. Wood, Longitudinal Dispersion with Dead Zones, *Journal of the Hydraulics Division, ASCE*, 103(HY9), 975-990, 1977.

Appendix A

Derivation of the Governing Differential Equations

A1.0 INTRODUCTION

The primary purpose of this appendix is to derive the differential equations that are presented in Section 3.2. Implicit in this derivation are several underlying assumptions:

- The fate and transport of the modeled solute can be described by a one-dimensional system.
- Model parameters may be spatially and temporally variable. Spatial variation is assumed to occur in the longitudinal direction only.
- The mechanisms shown in Table 1 influence solute behavior in the stream channel and the storage zone, as indicated.
- The volume of the storage zones does not change in time.

Mechanistic models such as the one presented in this report are based on a fundamental engineering principle, *conservation of mass*. This principle states that for a finite volume of water, mass is neither created or destroyed (see Chapra and Reckhow, 1983). In short, mass conservation implies that:

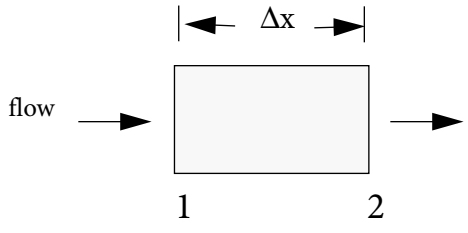
$$\text{accumulation} = \text{sources} - \text{sinks} \tag{A1}$$

or equivalently:

$$accumulation = in - out \tag{A2}$$

Equation (A2) is often called a mass balance equation. This mass balance must be applied to a finite volume of water, known as a control volume. A control volume for our one-dimensional system is shown below. Note that the terms '*segment*' (introduced in Section 3.2) and '*control-volume*' can be used interchangeably. In Figure A1, Δx represents the segment length, while '1' and '2' are used to demarcate the upstream and downstream boundaries of the control-volume.

FIGURE A1
1-D Control Volume



The remainder of this appendix is devoted to the formulation of differential equations based on Equation (A2). To accomplish this task, we first define '*accumulation*' and then proceed to develop the '*in*' and '*out*' terms for each process given in Table 1. This procedure is carried out for both the stream channel and the storage zone.

A2.0 MASS BALANCE - STREAM CHANNEL

I. Accumulation

By definition, accumulation is the change in mass with respect to time. For the main stream channel, accumulation is given by:

$$accumulation = \frac{\partial(VC)}{\partial t} \tag{A3}$$

where

C - in-stream solute concentration [mass/meters^3]

V - volume of the main channel segment [meters^3]

t - time [seconds]

We can now expand Equation (A3) using the chain rule, yielding:

$$\frac{\partial(VC)}{\partial t} = V\frac{\partial C}{\partial t} + C\frac{\partial V}{\partial t} = \Delta x A\frac{\partial C}{\partial t} + \Delta x C\frac{\partial A}{\partial t} \tag{A4}$$

where

Δx - segment length [meters]

A - stream channel cross-sectional area [meters^2]

II. Advection

We now begin to develop the '*in*' and '*out*' terms for each process. The mass advected into the control volume, i.e. across the upstream boundary, is given by:

$$In = (QC)_1 \tag{A5}$$

where Q is the volumetric flowrate [meters^3/second].

Next, the mass advected out of the control volume, i.e. across the downstream boundary, is given by:

$$Out = (QC)_2 = (QC)_1 + \frac{\partial}{\partial x}(QC)\Delta x \tag{A6}$$

Finally, we combine the 'In' and 'Out' terms, giving:

$$In - Out = -\frac{\partial}{\partial x}(QC)\Delta x = -\left(Q\frac{\partial C}{\partial x} + C\frac{\partial Q}{\partial x}\right)\Delta x \tag{A7}$$

III. Dispersion

In a similar fashion, we define the dispersive flux across the upstream and downstream boundaries:

$$In = -\left(AD\frac{\partial C}{\partial x}\right)_1 \tag{A8}$$

$$Out = -\left[\left(AD\frac{\partial C}{\partial x}\right)_1 + \frac{\partial}{\partial x}\left(AD\frac{\partial C}{\partial x}\right)\Delta x\right] \tag{A9}$$

$$In - Out = \frac{\partial}{\partial x}\left(AD\frac{\partial C}{\partial x}\right)\Delta x \tag{A10}$$

where D is the dispersion coefficient [meters^2/second].

IV. Lateral Flows

Here we are interested in mass entering the system via lateral inflow, and mass leaving through lateral outflow. Note that unlike the previous two mechanisms, the concentrations associated with mass inflows and outflows differ (C_L vs. C):

$$In = C_L q_{LIN}\Delta x \tag{A11}$$

$$Out = Cq_{LOUT}\Delta x \tag{A12}$$

$$In - Out = (C_L q_{LIN} - Cq_{LOUT})\Delta x \tag{A13}$$

where

C_L - solute concentration of the lateral inflow [mass/meters^3]

q_{LIN} - lateral inflow rate [meters^3/second-meter]

q_{LOUT} - lateral outflow rate [meters^3/second-meter]

V. Transient Storage

The final physical mechanism we consider for the stream channel is transient storage. The inflow and outflow terms are given by:

$$In = \alpha V C_S \tag{A14}$$

$$Out = \alpha V C \tag{A15}$$

$$In - Out = \alpha V(C_S - C) = \alpha \Delta x A(C_S - C) \tag{A16}$$

where

α - storage zone exchange coefficient [/second]

C_S - storage zone solute concentration [mass/meters^3]

Note that the exchange coefficient, α, is defined as the fraction of the main channel volume that is exchanged with the storage zone, per unit time.

VI. Decay

In addition to the foregoing physical processes, we also consider the loss of solute mass due to first order decay. Since decay represents a one-way loss from the system, our '*In*' term is zero. This results in:

$$In - Out = -\lambda VC \tag{A17}$$

where λ is the in-stream first-order decay rate [/second]. Note that if λ is a negative number, Equation (A17) yields a positive quantity. In this case, the decay term is actually a first-order production mechanism, wherein solute mass is added to the system.

VII. Accumulation = In - Out

Now that the individual processes have been elucidated, the results from I-VI may be used to develop the mass balance (Equation (A2)). Setting the accumulation term equal to the combined '*In - Out*' terms and dividing by Δx gives:

$$A\frac{\partial C}{\partial t} + C\frac{\partial A}{\partial t} = -\left(Q\frac{\partial C}{\partial x} + C\frac{\partial Q}{\partial x}\right) + \frac{\partial}{\partial x}\left(AD\frac{\partial C}{\partial x}\right) + (C_L q_{LIN} - Cq_{LOUT}) + \alpha A(C_S - C) - \lambda AC \tag{A18}$$

Although Equation (A18) represents a mass balance for a one-dimensional stream, it may be simplified using the water balance shown below.

VIII. Water Balance

Like the solute mass balance presented above, a water mass balance is based on conservation of mass. Here again we develop Equation (A2) for a one-dimensional control volume or stream segment. Using the assumption of incompressible flow, accumulation is given by:

$$Accumulation = \frac{\partial}{\partial t}(V\rho) = \rho\Delta x\frac{\partial A}{\partial t} \tag{A19}$$

where ρ is the density of water [mass/meters^3].

The inflow term for the water balance is the sum of the water advected across the upstream interface and the water entering via lateral inflow:

$$In = Q_1\rho + q_{LIN}\Delta x\rho \tag{A20}$$

Similarly, outflow is the sum of the water advected out and that leaving through lateral outflow:

$$Out = Q_2\rho + q_{LOUT}\Delta x\rho = \left(Q_1 + \frac{\partial Q}{\partial x}\Delta x\right)\rho + q_{LOUT}\Delta x\rho \tag{A21}$$

Finally, the water balance is completed by combining Equations (A19) through (A21), and dividing by ρΔx:

$$\frac{\partial A}{\partial t} = -\frac{\partial Q}{\partial x} + (q_{LIN} - q_{LOUT}) \tag{A22}$$

IX. Governing Equation - Stream Channel

Substituting Equation (A22) into Equation (A18) and dividing by A yields the governing differential equation for the stream channel:

$$\frac{\partial C}{\partial t} = -\frac{Q}{A}\frac{\partial C}{\partial x} + \frac{1}{A}\frac{\partial}{\partial x}(AD\frac{\partial C}{\partial x}) + \frac{q_{LIN}}{A}(C_L - C) + \alpha(C_S - C) - \lambda C \quad \text{(A23)}$$

This equation is presented in Section 3.2.1 as Equation (1).

A3.0 MASS BALANCE - STORAGE ZONE

As in Section A2.0, a mass balance procedure is used to develop the governing equation. Here the approach is greatly simplified, as the processes of advection, dispersion and lateral inflow do not affect the storage zone.

I. Accumulation

Accumulation of solute mass in the storage zone is given by:

$$\frac{\partial}{\partial t}(V_S C_S) = V_S\frac{\partial C_S}{\partial t} + C_S\frac{\partial V_S}{\partial t} = \Delta x A_S\frac{\partial C_S}{\partial t} \quad \text{(A24)}$$

where

V_S - volume of the storage zone segment [meters^3]

A_S - storage zone cross-sectional area [meters^2]

Note that the storage zone area, A_S, is assumed to be constant in time.

II. Transient Storage

The change in storage zone solute mass is given by:

$$In = \alpha V C \quad \text{(A25)}$$

$$Out = \alpha V C_S \quad \text{(A26)}$$

$$In - Out = \alpha \Delta x A (C - C_S) \quad \text{(A27)}$$

where α is the fraction of the main channel volume that is exchanged with the storage zone, per unit time.

III. Decay

First-order decay also occurs in the storage zone:

$$In - Out = -\lambda_S V_S C_S \quad \text{(A28)}$$

where λ_S is the storage-zone first-order decay coefficient [/second]. Note that the rate of decay in the storage zone is independent from that in the main channel.

IV. Governing Equation - Storage Zone

Combining Equations (A19), (A24), (A27) and (A28) we arrive at:

$$\frac{dC_S}{dt} = \alpha \frac{A}{A_S}(C - C_S) - \lambda_S C_S \qquad \text{(A29)}$$

This governing equation is presented in Section 3.2.1 as Equation (2).

Appendix B

Finite Difference Approximations and Boundary Conditions

B1.0 INTRODUCTION

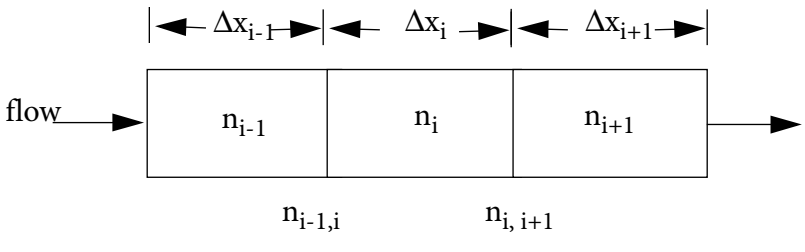
As discussed in Section 3.4, the spatial derivatives in the governing differential equations may be approximated using finite difference techniques. These techniques are the subject of this appendix. For a review of finite difference modeling, the reader is referred to Thomann and Mueller (1987).

We begin the discussion with a brief overview of the finite difference approximations required for the one-dimensional stream network. This overview is followed by a development of the equations for the individual segments or control-volumes that comprise the system. Two concluding sections discuss a special case of the approximations, *boundary conditions*.

B2.0 FINITE DIFFERENCE EQUATIONS (FDEs)

Equation (1), the mass balance equation for the stream channel, contains several spatial derivatives that may be approximated via finite differences. In general, spatial derivatives and model parameters need to be defined at both the center of the stream segment and at the segment's interfaces. These locations are defined in Figure B1, where an arbitrary quantity, *n*, is shown. Here we see that the subscript *'i'* is used to represent the quantity at the center of the segment. Quantities at the upstream interface, where segment *i-1* adjoins segment *i*, are indicated using the subscript *'i-1,i'*. In a similar fashion, the downstream interface is marked using the subscript *'i,i+1'*.

FIGURE B1
Central and Interfacial Quantities



I. The Center of the Segment

The first task is to evaluate the spatial derivative, ∂n/∂x, at the center of segment *i*. Using a centered difference, the spatial derivative of any variable is given by:

$$\left.\frac{\partial n}{\partial x}\right|_i = \frac{n_{i,\,i+1} - n_{i-1,\,i}}{\Delta x_i} \qquad \text{(B1)}$$

Note that Equation (B1) introduces the interfacial quantities, $n_{i-1,i}$ and $n_{i,i+1}$. Since the mass balance equations were developed for the center of each segment, the interfacial quantities should be defined in terms of n_{i-1}, n_i and n_{i+1}. To do this, we compute the interfacial quantities by interpolating linearly between the centers of the adjacent segments. This results in the following definitions:

$$n_{i-1,\,i} = \frac{\Delta x_i}{\Delta x_{i-1} + \Delta x_i} n_{i-1} + \frac{\Delta x_{i-1}}{\Delta x_{i-1} + \Delta x_i} n_i \tag{B2}$$

$$n_{i,\,i+1} = \frac{\Delta x_i}{\Delta x_{i+1} + \Delta x_i} n_{i+1} + \frac{\Delta x_{i+1}}{\Delta x_{i+1} + \Delta x_i} n_i \tag{B3}$$

II. The Segment Interfaces

We must also define the spatial derivatives at the upstream and downstream interfaces of the segment. These spatial derivatives are defined as the change in n divided by the distance over which the change occurs. For the upstream interface, this gives:

$$\left.\frac{\partial n}{\partial x}\right|_{i-1,\,i} = \frac{2(n_i - n_{i-1})}{\Delta x_i + \Delta x_{i-1}} \tag{B4}$$

The same approach is used to evaluate the spatial derivative at the downstream boundary. This yields:

$$\left.\frac{\partial n}{\partial x}\right|_{i,\,i+1} = \frac{2(n_{i+1} - n_i)}{\Delta x_i + \Delta x_{i+1}} \tag{B5}$$

B3.0 INTERIOR SEGMENTS - FDEs

Using the foregoing definitions, it is now possible to approximate the spatial derivatives in Equation (1). Note that the first term on the right-hand side of Equation (1) is the '*Advection*' term while the second is the '*Dispersion*' term. These two quantities are now approximated for the *interior segments* of the stream network. *Boundary segments*, i.e. the first and last segments, represent special cases that are discussed in Sections B4.0 and B5.0

I. Advection

Using definition (B1), the advective term may be approximated as:

$$\left(-\frac{Q}{A}\frac{\partial C}{\partial x}\right)_i \cong -\frac{Q_i}{A_i}\left(\frac{C_{i,\,i+1} - C_{i-1,\,i}}{\Delta x_i}\right) \tag{B6}$$

We now replace the interfacial concentrations ($C_{i,i+1}$ and $C_{i-1,i}$) with the definitions given by Equations (B2) and (B3). This substitution results in a centered difference approximation for $\partial C/\partial x$, given equal segment lengths:

$$= -\frac{Q_i}{A_i \Delta x_i}\left(\frac{\Delta x_i}{\Delta x_{i+1} + \Delta x_i} C_{i+1} + \frac{\Delta x_{i+1}}{\Delta x_{i+1} + \Delta x_i} C_i - \frac{\Delta x_i}{\Delta x_{i-1} + \Delta x_i} C_{i-1} - \frac{\Delta x_{i-1}}{\Delta x_{i-1} + \Delta x_i} C_i\right) \quad \text{(B7)}$$

II. Dispersion

Here again we use definition (B1) to define the spatial derivative:

$$\left[\frac{1}{A}\frac{\partial}{\partial x}\left(AD\frac{\partial C}{\partial x}\right)\right]_i \cong \frac{1}{A_i}\left[\frac{\left(AD\frac{\partial C}{\partial x}\right)_{i,\,i+1} - \left(AD\frac{\partial C}{\partial x}\right)_{i-1,\,i}}{\Delta x_i}\right] \quad \text{(B8)}$$

As the right side of Equation (B8) still contains spatial derivatives, we need the definitions given as Equations (B4) and (B5). Replacing the interfacial derivatives, we arrive at:

$$\cong \frac{1}{A_i \Delta x_i}\left[AD_{i,\,i+1}\frac{2(C_{i+1} - C_i)}{(\Delta x_i + \Delta x_{i+1})} - AD_{i-1,\,i}\frac{2(C_i - C_{i-1})}{(\Delta x_i + \Delta x_{i-1})}\right] \quad \text{(B9)}$$

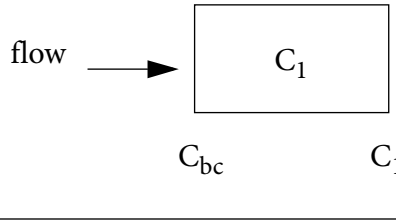
B4.0 THE FIRST SEGMENT - Upstream Boundary Condition

In the previous section, finite difference approximations for the spatial derivatives were developed. These approximations may be applied directly to all interior segments in the stream network. A problem arises, however, when we are dealing with the first segment, as all quantities involving the subscript *i-1* are undefined. Equations (B7) and (B9) must be modified accordingly.

I. Fixing the Upstream Boundary Condition

To effectively implement the finite difference equations for the first segment, an upstream boundary condition must be defined. This condition is shown in Figure B2, where the interfacial concentration at the upstream boundary is set to a user-supplied concentration, C_{bc}. This quantity is known as the upstream boundary condition.

FIGURE B2
Segment 1 - Upstream B.C.



II. Advection

Given the definition of C_{bc}, Equation (B6) can be modified for use in the first segment:

$$\left(-\frac{Q}{A}\frac{\partial C}{\partial x}\right)_1 = -\frac{Q_1}{A_1}\left(\frac{C_{1,2} - C_{bc}}{\Delta x_1}\right) \tag{B10}$$

As with the interior segments, replace the interfacial concentration ($C_{1,2}$) using Equation (B3):

$$= -\frac{Q_1}{A_1 \Delta x_1}\left(\frac{\Delta x_1}{\Delta x_1 + \Delta x_2} C_2 + \frac{\Delta x_2}{(\Delta x_1 + \Delta x_2)} C_1 - C_{bc}\right) \tag{B11}$$

III. Dispersion

Using the equation developed for the interior segments, the dispersive term is given by:

$$\left[\frac{1}{A}\frac{\partial}{\partial x}\left(AD\frac{\partial C}{\partial x}\right)\right]_i \cong \frac{1}{A_1}\left[\frac{\left(AD\frac{\partial C}{\partial x}\right)_{1,2} - \left(AD\frac{\partial C}{\partial x}\right)_{0,1}}{\Delta x_1}\right] \tag{B12}$$

The first interfacial derivative (at 1,2) is defined as before using Equation (B5). The second term (at 0,1), requires an additional equation:

$$\left.\frac{\partial C}{\partial x}\right|_{0,1} = \frac{C_1 - C_{bc}}{\frac{1}{2}\Delta x_1} \tag{B13}$$

Note that the forward difference approximation shown above must be used in lieu of Equation (B4) to avoid introducing C_0, the concentration in the non-existent segment zero. Equations (B5) and (B13) may now be used:

$$\cong \frac{1}{A_1 \Delta x_1}\left[AD_{1,2}\frac{2(C_2 - C_1)}{\Delta x_1 + \Delta x_2} - AD_{0,1}\frac{2(C_1 - C_{bc})}{\Delta x_1}\right] \tag{B14}$$

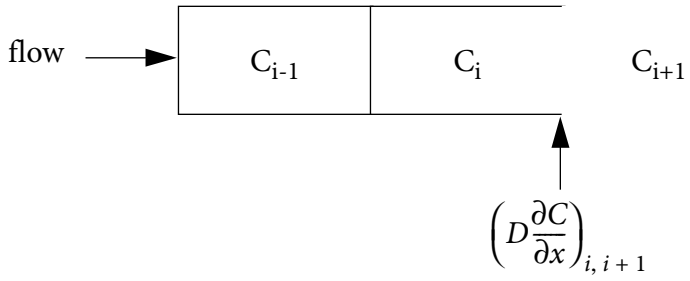
B5.0 THE LAST SEGMENT - Downstream Boundary Condition

The finite difference equations developed in Section B3.0 must also be modified to accommodate the last segment in the stream network. The required modifications to Equations (B7) and (B9) are shown below.

I. Fixing the Downstream Boundary Condition

Application of Equations (B7) and (B9) to the last segment results in the introduction of an solute concentration, C_{i+1}. This concentration is undefined as segment *i+1* does not exist. The last segment, *i*, and the fictitious segment, *i+1,* are shown in Figure B3.

FIGURE B3
Downstream Boundary Condition



To eliminate C_{i+1} from the finite difference equations, the dispersive flux at the interface between segments *i* and *i+1* is defined as in Equation (27):

$$\left(D\frac{\partial C}{\partial x}\right)\Bigg|_{i,\,i+1} = DSBOUND \tag{B15}$$

where DSBOUND is a user-supplied value for the dispersive flux. Note that if DSBOUND is set equal to zero, Equation (27) is a *zero gradient boundary condition* (Arden and Astill, 1970).

Using the definition of the interfacial first derivative given by Equation (B5), Equation (27) is solved for C_{i+1}, yielding:

$$C_{i+1} = \frac{(\Delta x_i + \Delta x_{i+1})DSBOUND}{2D_{i,\,i+1}} + C_i \tag{B16}$$

Equation (B16) is now used to eliminate C_{i+1} from Equations (B7) and (B9).

II. Advection

Substituting Equation (B16) into (B7) and replacing Δx_{i+1} with Δx_i, we arrive at the finite difference approximation of the advection term:

(B17)

$$\left(\frac{-Q}{A}\frac{\partial C}{\partial x}\right)_i = -\frac{Q_i}{A_i\Delta x_i}\left(\frac{\frac{1}{2}\Delta x_i DSBOUND}{D_{i,\,i+1}} + C_i - \frac{\Delta x_i}{(\Delta x_{i-1} + \Delta x_i)}C_{i-1} - \frac{\Delta x_{i-1}}{(\Delta x_{i-1} + \Delta x_i)}C_i\right)$$

III. Dispersion

The finite difference approximation of the dispersion term is now developed by substituting Equation (B16) into (B9) and replacing Δx_{i+1} with Δx_i:

$$\left[\frac{1}{A}\frac{\partial}{\partial x}\left(AD\frac{\partial C}{\partial x}\right)\right]_i = \frac{1}{A_i \Delta x_i}\left[A_{i,\,i+1} DSBOUND - AD_{i-1,\,i}\frac{2(C_i - C_{i-1})}{(\Delta x_i + \Delta x_{i-1})}\right] \tag{B18}$$

Appendix C

Crank-Nicolson Matrix Coefficients
Numerical Solution of the Dynamic Equations

C1.0 INTRODUCTION

As discussed in Section 3.4, the Crank-Nicolson method is used to solve the dynamic solute transport equations (Equations (2) and (6)). The Crank-Nicolson solution of the main channel equation (Equation (6)) is the subject of this appendix. A similar discussion for the storage zone (Equation (2)) is given in the main body of this report (Section 3.4.4).

The governing equation for the stream channel is given by:

$$\frac{dC}{dt} = L[C, Q, A, D, \Delta x] + \frac{q_{LIN}}{A_i}(C_L - C_i) + \alpha(C_S - C_i) - \lambda C_i \tag{C1}$$

where *L[C,Q,A,D,Δx]* is the finite difference approximation of the advection and dispersion terms, as given in Appendix B. In order to attain second order accuracy in both time and space, Equation (6) is evaluated at the intermediate time level, *j+1/2*. To this end, the time derivative is approximated at the intermediate time level using a centered finite difference:

$$\left.\frac{dC}{dt}\right|_{j+\frac{1}{2}} = \frac{C_i^{j+1} - C_i^j}{\Delta t} \tag{C2}$$

where

Δt - the integration time step [seconds]

j - denotes the value of a parameter or variable at the current time

j+1 - denotes the value of a parameter or variable at the advanced time

Furthermore, the right-hand side at *j+1/2* is simply the average of the terms evaluated at times *j* and *j+1*. Combining this average with Equation (C2), Equation (6) becomes:

$$\frac{C_i^{j+1} - C_i^j}{\Delta t} = \frac{G[C, C_L, C_S, \ldots]^{j+1} + G[C, C_L, C_S, \ldots]^j}{2} \tag{C3}$$

where

$$G[C, C_L, C_S, Q, A, D, q_{LIN}, \Delta x, \alpha, \lambda] = L[C, Q, A, D, \Delta x] + \frac{q_{LIN}}{A_i}(C_L - C_i) + \alpha(C_S - C_i) - \lambda C_i \tag{C4}$$

Because Equation (9) is dependent on the solute concentrations in the neighboring segments at the advanced time level (C_{i-1}, C_{i+1} at time $j+1$), it is not possible to explicitly solve for C_i^{j+1}. In the sections that follow, Equation (9) is rearranged so that all of the known quantities (i.e. those at time j) appear on the right-hand side and all of the unknown quantities (i.e. those at time $j+1$) appear on the left. This rearrangement yields:

$$\boldsymbol{E}_i C_{i-1}^{j+1} + \boldsymbol{F}_i C_i^{j+1} + \boldsymbol{G}_i C_{i+1}^{j+1} = \boldsymbol{R}_i \tag{C5}$$

where **E**, **F** and **G** are matrix coefficients and **R** is a forcing function.

We now proceed to develop the matrix coefficients and the forcing function by examining each term in Equation (9). In general, unknown quantities in each term contribute to the matrix coefficients while known quantities make up the forcing function. After going through each term, individual contributions are summed (see Section C4.0) to obtain **E**, **F**, **G** and **R**. This analysis assumes that Q, C_L, q_{LIN}, A, λ and α are constant in time. This assumption may introduce some error for simulations of solute transport under unsteady flow conditions.

C2.0 LEFT-HAND SIDE OF EQUATION C3 - TIME DERIVATIVE

As noted above, Equation (9) must be rearranged to develop the coefficients in Equation (12). We begin this process by considering the left side of Equation (9), the time derivative. Leaving the unknown concentration, C_i^{j+1}, on the left side and moving the known concentration to the right side, the contributions are given by:

$$\boldsymbol{F}_{dt} = \frac{1}{\Delta t} \tag{C6}$$

$$\boldsymbol{R}_{dt} = \frac{1}{\Delta t} C_i^j \tag{C7}$$

C3.0 RIGHT-HAND SIDE OF EQUATION C3

The right side of Equation (9) must also be rearranged such that the unknown quantities appear on the left side of Equation (12) and the known quantities appear on the right. As shown in Equation (C4), $G[C,C_L,C_S,...]$ is made up of several terms. Each term in $G[]$ is examined below.

C3.1 Advection

In Equation (C4), the operator $L[C,Q,A,D,\Delta x]$ represents the finite difference approximation of the spatial derivatives describing advection and dispersion. The contribution of the advection term is the subject of this section, while the dispersion term is discussed subsequently.

Using the finite difference equations for the advection term (Equations (B7), (B11), and (B17)), we develop the contributions to the matrix coefficients shown below. Note that due to the presence of boundary conditions, the contributions depend on the type of segment for which Equation (12) is developed.

Interior Segments

Consulting Equation (B7), we see that:

$$\boldsymbol{E}_{adv} = -\frac{Q_i}{2A_i(\Delta x_{i-1} + \Delta x_i)} \tag{C8}$$

$$\boldsymbol{F}_{adv} = \frac{Q_i}{2A_i\Delta x_i}\left(\frac{\Delta x_{i+1}}{\Delta x_{i+1} + \Delta x_i} - \frac{\Delta x_{i-1}}{\Delta x_{i-1} + \Delta x_i}\right) \tag{C9}$$

$$\boldsymbol{G}_{adv} = \frac{Q_i}{2A_i(\Delta x_{i+1} + \Delta x_i)} \tag{C10}$$

$$\boldsymbol{R}_{adv} = -\boldsymbol{E}_{adv}C^j{}_{i-1} - \boldsymbol{F}_{adv}C^j_i - \boldsymbol{G}_{adv}C^j_{i+1} \tag{C11}$$

where **E**, **F** and **G** are given by Equations (C8) through (C10).

First Segment

Using Equation (B11):

$$\boldsymbol{F}_{adv} = \frac{Q_1\Delta x_2}{2A_1\Delta x_1(\Delta x_1 + \Delta x_2)} \tag{C12}$$

$$\boldsymbol{G}_{adv} = \frac{Q_1}{2A_1(\Delta x_1 + \Delta x_2)} \tag{C13}$$

$$\boldsymbol{R}_{adv} = -\boldsymbol{F}_{adv}C^j_1 - \boldsymbol{G}_{adv}C^j_2 + \frac{Q_1}{A_1\Delta x_1}C_{bc} \tag{C14}$$

Last Segment

Evaluating Equation (B17) at *j+1/2* yields:

$$\boldsymbol{E}_{adv} = -\frac{Q_i}{2A_i(\Delta x_{i-1} + \Delta x_i)} \tag{C15}$$

$$\boldsymbol{F}_{adv} = \frac{Q_i}{2A_i\Delta x_i}\left(1 - \frac{\Delta x_{i-1}}{\Delta x_{i-1} + \Delta x_i}\right) \tag{C16}$$

$$\boldsymbol{R}_{adv} = -\boldsymbol{E}_{adv}C^j{}_{i-1} - \boldsymbol{F}_{adv}C^j_i - \frac{Q_i}{2A_i}\frac{DSBOUND}{D_{i,\,i+1}} \tag{C17}$$

C3.2 Dispersion

As with the advection term, the finite difference equations given in Appendix B (Equations (B9), (B14) and (B8)) are used to determine the contributions to the matrix coefficients.

Interior Segments

Equation (B9), evaluated at *j+1/2*, yields:

$$\boldsymbol{E}_{disp} = -\frac{(AD)_{i-1,\,i}}{A_i \Delta x_i (\Delta x_i + \Delta x_{i-1})} \tag{C18}$$

$$\boldsymbol{F}_{disp} = \frac{1}{A_i \Delta x_i}\left(\frac{AD_{i,\,i+1}}{\Delta x_i + \Delta x_{i+1}} + \frac{AD_{i-1,\,i}}{\Delta x_i + \Delta x_{i-1}}\right) \tag{C19}$$

$$\boldsymbol{G}_{disp} = -\frac{(AD)_{i,\,i+1}}{A_i \Delta x_i (\Delta x_i + \Delta x_{i+1})} \tag{C20}$$

$$\boldsymbol{R}_{disp} = -\boldsymbol{E}_{disp} C_{i-1}^{j} - \boldsymbol{F}_{disp} C_i^{j} - \boldsymbol{G}_{disp} C_{i+1}^{j} \tag{C21}$$

First Segment

Using the finite difference equation for the first segment (Equation (B14)):

$$\boldsymbol{F}_{disp} = \frac{1}{A_1 \Delta x_1}\left(\frac{AD_{1,\,2}}{\Delta x_1 + \Delta x_2} + \frac{AD_{0,\,1}}{\Delta x_1}\right) \tag{C22}$$

$$\boldsymbol{G}_{disp} = -\frac{1}{A_1 \Delta x_1}\left[\frac{(AD)_{1,\,2}}{\Delta x_1 + \Delta x_2}\right] = -\frac{(AD)_{1,\,2}}{A_1 \Delta x_1 (\Delta x_1 + \Delta x_2)} \tag{C23}$$

$$\boldsymbol{R}_{disp} = -\boldsymbol{F}_{disp} C_1^{j} - \boldsymbol{G}_{disp} C_2^{j} + \frac{2AD_{0,\,1}}{A_1 \Delta x_1^2} C_{bc} \tag{C24}$$

Last Segment

Finally, Equation (B8) at *j+1/2* yields:

$$\boldsymbol{E}_{disp} = -\frac{(AD)_{i-1,\,i}}{A_i \Delta x_i (\Delta x_i + \Delta x_{i-1})} \tag{C25}$$

$$\boldsymbol{F}_{disp} = \frac{(AD)_{i-1,\,i}}{A_i \Delta x_i (\Delta x_i + \Delta x_{i-1})} \tag{C26}$$

$$\boldsymbol{R}_{disp} = -\boldsymbol{E}_{disp} C_{i-1}^{j} - \boldsymbol{F}_{disp} C_i^{j} + \frac{A_{i,\,i+1} DSBOUND}{A_i \Delta x_i} \tag{C27}$$

C3.3 Lateral Inflow

The third term in *G[]*, lateral inflow, is also evaluated at *j+1/2*:

$$\left(\frac{q_{LIN}}{A}(C_L - C)\right)^{j+\frac{1}{2}} = \frac{\left(\frac{q_{LIN}}{A}(C_L - C_i)\right)^{j} + \left(\frac{q_{LIN}}{A}(C_L - C_i)\right)^{j+1}}{2} \tag{C28}$$

Assuming q_{LIN}, C_L and A_i to be time invariant, Equation (C28) becomes:

$$= \frac{q_{LIN}}{2A_i}(2C_L - C_i^j - C_i^{j+1}) \tag{C29}$$

The contributions are thus:

$$\boldsymbol{F}_{lat} = \frac{q_{LIN}}{2A_i} \tag{C30}$$

$$\boldsymbol{R}_{lat} = \frac{q_{LIN}}{A_i}C_L - \frac{q_{LIN}}{2A_i}C_i^j \tag{C31}$$

C3.4 Transient Storage

The transient storage term is now evaluated at *j+1/2*:

$$\alpha(C_s - C_i)^{j+\frac{1}{2}} = \frac{\alpha(C_s - C_i)^j + \alpha(C_s - C_i)^{j+1}}{2} = \frac{\alpha}{2}(C^j{}_s - C_i^j + C_S^{j+1} - C_i^{j+1}) \tag{C32}$$

To decouple the governing transport equations (see Section 3.4.5), Equation (19) is used to eliminate C_S^{j+1}. Equation (C32) thus becomes:

$$= \frac{\alpha}{2}\left(C_S^j - C_i^j - C_i^{j+1} + \frac{2 - \gamma - \Delta t\lambda_s}{2 + \gamma + \Delta t\lambda_s}C_s^j + \frac{\gamma}{2 + \gamma + \Delta t\lambda_s}C_i^j + \frac{\gamma}{2 + \gamma + \Delta t\lambda_s}C_i^{j+1}\right) \tag{C33}$$

The contributions to the matrix coefficients are given by:

$$\boldsymbol{F}_{stor} = \frac{\alpha}{2}\left(1 - \frac{\gamma}{2 + \gamma + \Delta t\lambda_s}\right) \tag{C34}$$

$$\boldsymbol{R}_{stor} = -\boldsymbol{F}_{stor}C_i^j + \frac{\alpha}{2}\left(1 + \frac{2 - \gamma - \Delta t\lambda_s}{2 + \gamma + \Delta t\lambda_s}\right)C_s^j \tag{C35}$$

C3.5 First-Order Decay

The final term in *G[]* represents first-order decay or production. This term is evaluated at *j+1/2*:

$$(-\lambda C)^{j+\frac{1}{2}} = \frac{-(\lambda C)^{j} - (\lambda C)^{j+1}}{2} = -\frac{\lambda}{2}(C_i^j + C_i^{j+1}) \tag{C36}$$

The contributions are given by:

$$\boldsymbol{F}_{decay} = \frac{\lambda}{2} \tag{C37}$$

$$\boldsymbol{R}_{decay} = -\boldsymbol{F}_{decay} C_i^j \tag{C38}$$

C4.0 MATRIX COEFFICIENTS

The matrix coefficients (**E**, **F**, **G**) and the forcing function (**R**) from Equation (12) may now be developed by summing the contributions given in Sections C2.0 and C3.0, and multiplying by Δt:

$$\boldsymbol{E}_i = \Delta t(\boldsymbol{E}_{adv} + \boldsymbol{E}_{disp}) \tag{C39}$$

$$\boldsymbol{F}_i = \Delta t(\boldsymbol{F}_{dt} + \boldsymbol{F}_{adv} + \boldsymbol{F}_{disp} + \boldsymbol{F}_{lat} + \boldsymbol{F}_{stor} + \boldsymbol{F}_{decay}) \tag{C40}$$

$$\boldsymbol{G}_i = \Delta t(\boldsymbol{G}_{adv} + \boldsymbol{G}_{disp}) \tag{C41}$$

$$\boldsymbol{R}_i = \Delta t(\boldsymbol{R}_{dt} + \boldsymbol{R}_{adv} + \boldsymbol{R}_{disp} + \boldsymbol{R}_{lat} + \boldsymbol{R}_{stor} + \boldsymbol{R}_{decay}) \tag{C42}$$

Appendix D

Steady State Matrix Coefficients

The purpose of this appendix is to develop the matrix coefficients (**E**, **F**, **G**) and the forcing function (**R**) from Equation (26). In the text that follows, the contributions to **E**, **F**, **G** and **R** are given on a term-by-term basis. The final values are determined by summing the individual contributions.

D1.0 ADVECTION

Equations (B7), (B11), and (B17) are used to approximate the spatial derivative describing advective transport. These equations contribute to the matrix coefficients as detailed below. Due to boundary conditions, the contributions depend on the type of segment for which Equation (26) is developed.

Interior Segments

$$\boldsymbol{E}_{adv} = -\frac{Q_i}{A_i}\left(\frac{1}{\Delta x_{i-1} + \Delta x_i}\right) \tag{D1}$$

$$\boldsymbol{F}_{adv} = \frac{Q_i}{A_i \Delta x_i}\left(\frac{\Delta x_{i+1}}{\Delta x_i + \Delta x_{i+1}} - \frac{\Delta x_{i-1}}{\Delta x_i + \Delta x_{i-1}}\right) \tag{D2}$$

$$\boldsymbol{G}_{adv} = \frac{Q_i}{A_i}\left(\frac{1}{\Delta x_i + \Delta x_{i+1}}\right) \tag{D3}$$

First Segment

$$\boldsymbol{F}_{adv} = \frac{Q_1}{A_1 \Delta x_1}\left(\frac{\Delta x_2}{\Delta x_1 + \Delta x_2}\right) \tag{D4}$$

$$\boldsymbol{G}_{adv} = \frac{Q_1}{A_1}\left(\frac{1}{\Delta x_1 + \Delta x_2}\right) \tag{D5}$$

$$\boldsymbol{R}_{adv} = \frac{Q_1}{A_1}\frac{C_{bc}}{\Delta x_1} \tag{D6}$$

Last Segment

$$\boldsymbol{E}_{adv} = -\frac{Q_i}{A_i}\left(\frac{1}{\Delta x_i + \Delta x_{i-1}}\right) \tag{D7}$$

$$\boldsymbol{F}_{adv} = \frac{Q_i}{A_i \Delta x_i}\left(1 - \frac{\Delta x_{i-1}}{\Delta x_i + \Delta x_{i-1}}\right) \tag{D8}$$

$$\boldsymbol{R}_{adv} = -\frac{Q_i}{A_i}\frac{DSBOUND}{2D_{i,\,i+1}} \tag{D9}$$

D2.0 DISPERSION

As with the advection term, the finite difference equations given in Appendix B (Equations (B9), (B14) and (B8)) are used to determine the contributions to the matrix coefficients.

Interior Segments

$$\boldsymbol{E}_{disp} = -\frac{2(AD)_{i-1,\,i}}{A_i \Delta x_i (\Delta x_i + \Delta x_{i-1})} \tag{D10}$$

$$\boldsymbol{F}_{disp} = \frac{2}{A_i \Delta x_i}\left[\frac{AD_{i,\,i+1}}{\Delta x_i + \Delta x_{i+1}} + \frac{AD_{i-1,\,i}}{\Delta x_i + \Delta x_{i-1}}\right] \tag{D11}$$

$$\boldsymbol{G}_{disp} = -\frac{2AD_{i,\,i+1}}{A_i \Delta x_i (\Delta x_i + \Delta x_{i+1})} \tag{D12}$$

First Segment

$$\boldsymbol{F}_{disp} = \frac{2}{A_1 \Delta x_1}\left(\frac{AD_{1,\,2}}{\Delta x_1 + \Delta x_2} + \frac{AD_{0,\,1}}{\Delta x_1}\right) \tag{D13}$$

$$\boldsymbol{G}_{disp} = -\frac{2AD_{1,\,2}}{A_1 \Delta x_1 (\Delta x_1 + \Delta x_2)} \tag{D14}$$

$$\boldsymbol{R}_{disp} = \frac{1}{A_1}\frac{2AD_{0,\,1}C_{bc}}{\Delta x_1^2} \tag{D15}$$

Last Segment

$$\boldsymbol{E}_{disp} = -\frac{2AD_{i-1,\,i}}{A_i \Delta x_i (\Delta x_i + \Delta x_{i-1})} \tag{D16}$$

$$\boldsymbol{F}_{disp} = \frac{2AD_{i-1,i}}{A_i \Delta x_i (\Delta x_i + \Delta x_{i-1})} \tag{D17}$$

$$\boldsymbol{R}_{disp} = \frac{A_{i,i+1} DSBOUND}{A_i \Delta x_i} \tag{D18}$$

D3.0 LATERAL INFLOW

$$\boldsymbol{F}_{lat} = \frac{q_{LIN}}{A_i} \tag{D19}$$

$$\boldsymbol{R}_{lat} = \frac{q_{LIN}}{A_i} C_L \tag{D20}$$

D4.0 TRANSIENT STORAGE

$$\boldsymbol{F}_{stor} = \frac{\alpha A_s \lambda_s}{\alpha A + A_s \lambda_s} \tag{D21}$$

D5.0 DECAY

$$\boldsymbol{F}_{decay} = \lambda \tag{D22}$$

D6.0 MATRIX COEFFICIENTS

The matrix coefficients (**E**, **F**, **G**) and the forcing function (**R**) from Equation (26) may now be developed by summing the contributions given in Sections D1.0 through D5.0:

$$\boldsymbol{E}_i = \boldsymbol{E}_{adv} + \boldsymbol{E}_{disp} \tag{D23}$$

$$\boldsymbol{F}_i = \boldsymbol{F}_{adv} + \boldsymbol{F}_{disp} + \boldsymbol{F}_{lat} + \boldsymbol{F}_{stor} + \boldsymbol{F}_{decay} \tag{D24}$$

$$\boldsymbol{G}_i = \boldsymbol{G}_{adv} + \boldsymbol{G}_{disp} \tag{D25}$$

$$\boldsymbol{R}_i = \boldsymbol{R}_{adv} + \boldsymbol{R}_{disp} + \boldsymbol{R}_{lat} \tag{D26}$$

Appendix E

The echo.out file from Application 1

DYNAMIC SOLUTE TRANSPORT MODEL
Version: MOD27 (July 23,1991)

Runtime Information
Number of Runs: 1
Date and Time : Thu Sep 26 10:17:55 1991

Input Data for Run Number 1
St. Kevin Gulch - 1986 Lithium Injection Experiment

PARAMETER FILE: params.inp

Print Option : 1
Requested Print Interval [hour] : 0.10000D+00
Integration Time Step [hour] : 0.10000D-01
Starting Time [hour] : 0.13900D+02
Ending Time [hour] : 0.82000D+02
Starting Distance [meters] : 0.00000D+00
Downstream Boundary Condition [mass/m^2-s] : 0.00000D+00
Simulation Type [dynamic or steady-state] : Dynamic

Number of Reaches : 7

Reach Information

Reach No.	Start Seg.	End Seg.	Start Distance [m]	End Distance [m]
1	1	26	0.000	26.000
2	27	484	26.000	484.000
3	485	526	484.000	526.000
4	527	948	526.000	948.000
5	949	1557	948.000	1557.000
6	1558	1804	1557.000	1804.000
7	1805	1904	1804.000	1904.000

Number of Spatial Segments : 1904

Reach Parameters

Reach No.	No. Segs.	Reach Length [m]	Segment Length [m]	Dispersion Coeff.[m^2/s]
1	26	26.000	1.000	0.20000D-01
2	458	458.000	1.000	0.20000D-01
3	42	42.000	1.000	0.20000D-01
4	422	422.000	1.000	0.20000D-01
5	609	609.000	1.000	0.20000D-01
6	247	247.000	1.000	0.20000D-01
7	100	100.000	1.000	0.20000D-01

Storage Parameters

Reach No.	Storage Zone Area [m^2]	Storage Rate [/sec]
1	0.50000D-01	0.30000D-04
2	0.50000D-01	0.20000D-04
3	0.25000D+00	0.20000D-04
4	0.10000D+00	0.15000D-04
5	0.20000D+00	0.50000D-04
6	0.20000D+00	0.30000D-04
7	0.20000D+00	0.30000D-04

Number of Solutes: 2
Chemistry : Conservative

Decay Coefficients, Solute # 1

Reach No.	Coefficient Channel [/second]	Coefficient Storage [/second]
1	0.000D+00	0.000D+00
2	0.000D+00	0.000D+00
3	0.000D+00	0.000D+00
4	0.000D+00	0.000D+00
5	0.000D+00	0.000D+00
6	0.000D+00	0.000D+00
7	0.000D+00	0.000D+00

Decay Coefficients, Solute # 2

Reach No.	Coefficient Channel [/second]	Storage [/second]
1	0.000D+00	0.000D+00
2	0.000D+00	0.000D+00
3	0.000D+00	0.000D+00
4	0.000D+00	0.000D+00
5	0.000D+00	0.000D+00
6	0.000D+00	0.000D+00
7	0.000D+00	0.000D+00

Number of Print Locations : 6

Printing Information

Print Location [m]
26.000
483.000
526.000
948.000
1557.000
1804.000

Number of Upstream Boundary Conditions: 3

Upstream Boundary Conditions, Solute # 1

Begin Time [hour]	Concentration [mass/m^3]
0.1200D+02	0.0000D+00
0.1400D+02	0.2363D+01
0.6600D+02	0.0000D+00

Upstream Boundary Conditions, Solute # 2

Begin Time [hour]	Concentration [mass/m^3]
0.1200D+02	0.0000D+00
0.1400D+02	0.1336D+02
0.6600D+02	0.0000D+00

Flow and Area Data

FLOW FILE: flow.inp

Flow Option : Steady
Flow interval [hour] : 0.00000D+00

Initial Flow and Area Values

Flowrate at Upstream Boundary [m^3/sec] : 0.61200D-02

Reach No.	Lateral Inflow [m^3/s-m]	Lateral Outflow [m^3/s-m]	X-sect. Area [m^2]
1	0.00000D+00	0.00000D+00	0.12000D+00
2	0.37800D-05	0.00000D+00	0.97000D-01
3	0.17000D-03	0.00000D+00	0.13800D+00
4	0.41200D-05	0.00000D+00	0.19900D+00
5	0.48400D-05	0.00000D+00	0.15200D+00
6	0.00000D+00	0.20300D-04	0.15300D+00
7	0.00000D+00	0.20300D-04	0.15300D+00

Initial Lateral Inflow Concentrations, Solute # 1

Reach No.	Inflow Concent. [mass/m^3]
1	0.50000D-02
2	0.36000D-01
3	0.50000D-02
4	0.50000D-02
5	0.50000D-02
6	0.50000D-02
7	0.50000D-02

Initial Lateral Inflow Concentrations, Solute # 2

Reach No.	Inflow Concent. [mass/m^3]
1	0.20000D+00
2	0.90000D+00
3	0.20000D+00
4	0.20000D+00
5	0.20000D+00
6	0.20000D+00
7	0.20000D+00

Output Files for Solutes

Solute #	File Name
1	lithium.out
2	chloride.out

Initial Conditions at the Print Locations

Location [m]	Prev. Seg.	Inter. Weight	Flow [m^3/sec]
26.00	26	0.500000D+00	0.612094D-02
483.00	483	0.500000D+00	0.784746D-02
526.00	526	0.500000D+00	0.149498D-01
948.00	948	0.500000D+00	0.167301D-01
1557.00	1557	0.500000D+00	0.196712D-01
1804.00	1804	0.500000D+00	0.146633D-01

Initial Conditions, Solute # 1

Location [m]	Concentration Channel [mass/m^3]	Storage [mass/m^3]
26.00	0.000D+00	0.000D+00
483.00	0.000D+00	0.000D+00
526.00	0.000D+00	0.000D+00
948.00	0.000D+00	0.000D+00
1557.00	0.000D+00	0.000D+00
1804.00	0.000D+00	0.000D+00

Initial Conditions, Solute # 2

Location	Concentration Channel	Storage
[m]	[mass/m^3]	[mass/m^3]
26.00	0.000D+00	0.000D+00
483.00	0.000D+00	0.000D+00
526.00	0.000D+00	0.000D+00
948.00	0.000D+00	0.000D+00
1557.00	0.000D+00	0.000D+00
1804.00	0.000D+00	0.000D+00

Appendix F

Program Listing

For a copy of the OTIS solute transport code, contact:

Robert L. Runkel
CU-CADSWES
2945 Center Green Court
Suite B
Boulder, Colorado 80301

Robert E. Broshears
U.S. Geological Survey
Water Resources Division
Mail Stop 415
Denver Federal Center
Lakewood, Colorado 80225