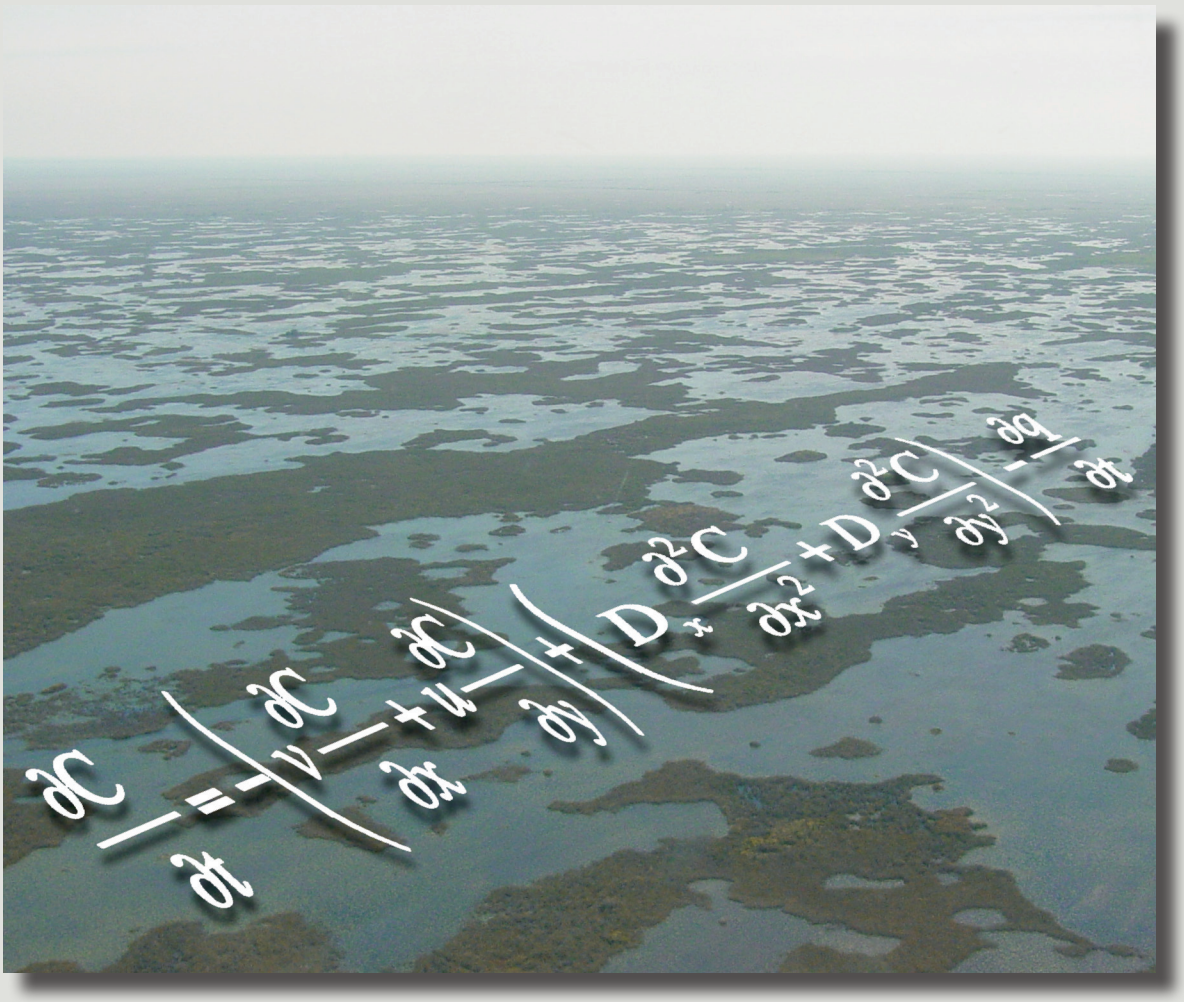
USGS
science for a changing world

Prepared in Cooperation with the South Florida Water Management District

Development and Implementation of a Transport Method for the Transport and Reaction Simulation Engine (TaRSE) based on the Godunov-Mixed Finite Element Method



Scientific Investigations Report 2009-5034

U.S. Department of the Interior
U.S. Geological Survey

Development and Implementation of a Transport Method for the Transport and Reaction Simulation Engine (TaRSE) based on the Godunov-Mixed Finite Element Method

By Andrew I. James[1,2], James W. Jawitz[1], and Rafael Muñoz-Carpena[1]

[1]University of Florida, Gainesville
[2]Soil and Water Engineering Technology, Inc., University of Florida, Gainesville

Prepared in cooperation with the
South Florida Water Management District

Scientific Investigations Report 2009–5034

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
KEN SALAZAR, Secretary

U.S. Geological Survey
Suzette M. Kimball, Acting Director

U.S. Geological Survey, Reston, Virginia: 2009

Suggested citation:
James, A.I., Jawitz, J.W., and Muñoz-Carpena, Rafael, 2009, Development and Implementation of a Transport Method for the Transport and Reaction Simulation Engine (TaRSE) based on the Godunov-Mixed Finite Element Method: U.S. Geological Survey Scientific Investigations Report 2009-5034, 40 p.

Contents

Figures

Tables

Conversion Factors

Multiply	By	To obtain
	Leakance	
foot per day per foot [(ft/d)/ft]	1	meter per day per meter
inch per year per foot [(in/yr)/ft]	83.33	millimeter per year per meter [(mm/yr)/m]
	Flow rate	
acre-foot per day (acre-ft/d)	0.01427	cubic meter per second (m^3/s)
acre-foot per year (acre-ft/yr)	1,233	cubic meter per year (m^3/yr)
	Density	
pound per cubic foot (lb/ft^3)	16.02	kilogram per cubic meter (kg/m^3)
pound per cubic foot (lb/ft^3)	0.01602	gram per cubic centimeter (g/cm^3)

Temperature in degrees Celsius (°C) may be converted to degrees Fahrenheit (°F) as follows:

°F=(1.8×°C)+32

Temperature in degrees Fahrenheit (°F) may be converted to degrees Celsius (°C) as follows:

°C=(°F-32)/1.8

Transmissivity: The standard unit for transmissivity is cubic foot per day per square foot times foot of aquifer thickness [(ft^3/d)/ft^2]ft. In this report, the mathematically reduced form, foot squared per day (ft^2/d), is used for convenience.

Acronyms and Abbreviations

ADE	advection-dispersion equation
ADRE	advection-dispersion-reaction equation
API	Application program interface
CFL	Courant-Friedrichs-Lewy
DTD	Document type definition
ELLAM	Eulerian-Lagrangian localized adjoint method
GMM	Godunov-mixed method
HSE	Hydrologic Simulation Engine (a component of the Regional Simulation Model)
RT_0	Raviart-Thomas space of order zero
RSM	Regional Simulation Model
SRP	Soluble reactive phosphorus
SFWMD	South Florida Water Management District
TaRSE	Transport and Reaction Simulation Engine

Additional abbreviated units

m	meter
s	second
g	gram
g/m^2	gram per square meter
g/m^3	gram per cubic meter
m^2/s	square meter per second

Development and Implementation of a Transport Method for the Transport and Reaction Simulation Engine (TaRSE) based on the Godunov-Mixed Finite Element Method

By Andrew I. James[1,2], James W. Jawitz[1], and Rafael Muñoz-Carpena[1]

Abstract

A model to simulate transport of materials in surface water and ground water has been developed to numerically approximate solutions to the advection-dispersion equation. This model, known as the Transport and Reaction Simulation Engine (TaRSE), uses an algorithm that incorporates a time-splitting technique where the advective part of the equation is solved separately from the dispersive part. An explicit finite-volume Godunov method is used to approximate the advective part, while a mixed-finite element technique is used to approximate the dispersive part. The dispersive part uses an implicit discretization, which allows it to run stably with a larger time step than the explicit advective step. The potential exists to develop algorithms that run several advective steps, and then one dispersive step that encompasses the time interval of the advective steps. Because the dispersive step is computationally most expensive, schemes can be implemented that are more computationally efficient than non-time-split algorithms. This technique enables scientists to solve problems with high grid Peclet numbers, such as transport problems with sharp solute fronts, without spurious oscillations in the numerical approximation to the solution and with virtually no artificial diffusion.

Introduction

Transport and reaction of solutes and other material in surface water and ground water have been studied for many years, and the advection-dispersion-reaction equation (ADRE) has been successfully applied to a wide range of problems of this type in many different settings. A number of solute transport and reaction problems with simple geometries and boundary and initial conditions have exact solutions. However, many applications of the ADRE in complex domains and with complex boundary and initial conditions do not have exact solutions. Therefore, numerical models are commonly used to simulate transport because they are more easily adaptable to complex domains and boundary conditions.

The U.S. Geological Survey and University of Florida, in cooperation with the South Florida Water Management District (SFWMD), worked in a collaborative effort to document the development and implementation of a transport model, Transport and Reaction Simulation Engine (TaRSE), for mobile components in surface-water bodies such as wetlands. Mobile components are defined to mean solutes and/or suspended material, such as plankton transported by water. This model separates the advective and dispersive parts of the transport equation and uses different approximating techniques on each part. The resulting numerical model approximates sharp fronts and conserves the mass in the system on a global and local scale. This approach is termed a time-split Godunov-mixed finite element method and follows the method of Mazzia and others (2000; 2002).

This study by the U.S. Geological Survey and University of Florida was conducted as part of the Cooperative Ecosystem Studies Units agreement (contract no. CESU 00212HS017). The project period was January 1, 2004, through April 30, 2007.

[1] University of Florida, Gainesville, FL 32611

[2] Soil and Water Engineering Technology, Inc., Gainesville, FL 32605

Purpose and Scope

The purpose of this report is to present a numerical method for simulating water quality within the SFWMD Regional Simulation Model (RSM), a two-dimensional surface-water and ground-water flow model (Lal and others, 2005). The RSM simulates surface-water and ground-water hydrodynamics on an unstructured triangular mesh using the Hydrologic Simulation Engine (HSE), which implements a finite-volume method to calculate water heads and flows. Initially, there was no means within the HSE to model either hydrologic transport of material or the reactions and transformations that may act on the mobile or non-mobile components. The computer model described in this report works with the RSM to simulate transport and reaction. As such, it uses hydrodynamics provided by the HSE to simulate transport in surface water.

Methods and Model Development

The initial attempt to add the capability to simulate mobile component transport to the RSM resulted in the implementation of a simple advective transport model that moved mobile components between elements (cells) in the RSM mesh. This attempt did not model transport using the ADRE, and was instead based on a simple finite-volume formulation that relied on an averaging technique to determine the concentration of the mobile component in each cell. It did not attempt to approximate the spatial distribution of transported material within a cell and diffusion/hydrodynamic dispersion was neglected. There were several advantages to this approach:

- If the element-to-element connectivity is available, then implementation should be straightforward because the only requirement is an explicit calculation at the end of every time step, followed by an updating of the concentration (element-to-element connectivity, however, was not provided by the RSM, so an inefficient workaround was required);
- This explicit method was inexpensive from a computational cost viewpoint, with no large linear system needing to be solved at each time step; and
- The method was simple to explain from a conceptual viewpoint.

These advantages are important; however, the simple advective transport method has deficiencies. In theory, advective transport can result in distribution of mobile components that have sharp solute fronts, as may happen when material is introduced across an inflow boundary into a domain where no material was initially present. A basic finite volume method under most conditions will introduce artificial spreading, where the material front becomes smeared out with time. This artificial diffusion is dependent on water velocity, time step, and grid size and (or) grid orientation. Diffusion and (or) hydrodynamic dispersion of mobile components transported through natural wetland systems is observable and measurable; unfortunately, the only means available to calibrate the model to match observed diffusion is to change the mesh geometry and (or) time step. It was thus determined that a purely advection based model was insufficient for the purposes of modeling material transport in wetlands. For problems with a significant diffusional component, a simple advective transport model serves a useful purpose as a first approximation, however, a model that directly incorporates a diffusion and (or) hydrodynamic dispersion term must be used if truly accurate numerical approximations are needed.

Consequently, work was undertaken to implement a numerical solution to the ADRE that could be used within the RSM. The model presented herein is the result of that work. The development of the model was largely based on the application of existing solution techniques and methods that have extensively developed theoretical underpinnings. This includes the mixed finite element technique to simulate hydrodynamic dispersion. Certain aspects of the methods used to simulate advection were developed in the course of this work. In either case, no suitable implementation of these methods in the form of computer code was found to be available. This current work is focused on the implementation of these methods as a computer code library for use in the RSM.

Previous Work

This section discusses several methods that have been used to either exactly solve or numerically approximate solutions to the ADRE. Discussion of the previous work has been divided into three parts. The first consists of previous work in finding exact solutions to the ADRE, whereas the other two focus on work developing numerical approximations to the ADRE. The latter is divided into two categories depending on how the time-dependent portion of the ADRE is treated. These categories are referred to as “Non-Operator-Splitting” and “Operator-Splitting” methods. In the following discussion, the form of the ADRE where a reaction component does not appear is referred to as the Advection-Dispersion Equation (ADE).

Exact Solutions

The advection-dispersion-reaction equation (ADRE) in one dimension can be written as (Bear, 1979):

$$\frac{\partial(\phi c)}{\partial t}+\frac{\partial}{\partial x}\left(c\,u-D\frac{\partial c}{\partial x}\right)+f_2 c=f_1 c_1 \quad , \tag{1}$$

where

t	is time,
x	is the spatial coordinate,
$c(x,t)$	is the concentration of a solute or suspended material [M L^{-3}],
$\phi(x,t)$	is the porosity (may equal 1 for surface water applications) [Ø],
$u(x,t)$	is the Darcy velocity/specific discharge [L T^{-1}],
$D(u[x,t])$	is the hydrodynamic dispersion tensor [L^2 T^{-1}], and
f_2	is a first-order sink/reaction term [T^{-1}].

The source term has rate f_1 [T^{-1}] and an associated concentration c_1. The dispersion tensor D combines molecular diffusion with hydrodynamic dispersion, and thus, is a function of u; because molecular diffusion is always greater than zero, D is positive definite (Arbogast and Wheeler, 1995). Equation 1 assumes that the water has constant density. Although the ADRE is formally classified as a parabolic partial differential equation, if the advective term u dominates the diffusion/dispersion term D so that $u \gg D$, it can become strongly hyperbolic in character.

There are several analytical solutions to equation 1 that are rather limited in application because they are restricted to steady-state flow problems with constant coefficients with regular boundaries. However, exact solutions are important for validating and testing numerical models, and are used for this purpose in the "Verification of the Algorithm" section of this report. Ogata and Banks (1961) solved the one-dimensional ADE (without a reaction term) in a semi-infinite column subject to a first-type (Dirichlet) boundary condition:

$$c(0,t)=C_0 \quad t\geq 0; \quad c(x,0)=0 \quad x>0 \quad , \tag{2}$$

where C_0 is the inlet concentration. They obtained the exact solution:

$$c(x,t)=\frac{C_0}{2}\operatorname{erfc}\left[\frac{x-u\,t}{\sqrt{4Dt}}\right]+\frac{C_0}{2}\exp\left(\frac{u\,x}{D}\right)\operatorname{erfc}\left[\frac{x+u\,t}{\sqrt{4Dt}}\right], \tag{3}$$

where erfc[·] is the complementary error function.

Gershon and Nir (1969) solved equation 1 subject to a third-type (Robin or Danckwerts) boundary condition (see also Bear, 1979) and the compilation of van Genuchten and Alves (1982) and arrived at the exact solution:

$$c(x,t)=C_0\left\{\frac{1}{2}\operatorname{erfc}\left[\frac{x-u\,t}{\sqrt{4Dt}}\right]+\left(\frac{u^2 t}{\pi D}\right)^{1/2}\exp\left[-\frac{(x-u\,t)^2}{4Dt}\right]-\frac{1}{2}\left(1+\frac{u\,x}{D}+\frac{u^2 t}{D}\right)\exp\left(\frac{u\,x}{D}\right)\operatorname{erfc}\left[\frac{x+u\,t}{\sqrt{4Dt}}\right]\right\}. \tag{4}$$

Non-Operator-Splitting Numerical Models

Traditionally, numerical approximations to the solution of the ADE relied on finite difference or finite element techniques (Pinder, 1973; van Genuchten and others, 1977; Pinder and Gray, 1977; Bear, 1979). These typically treat the advective and dispersive parts simultaneously, with the same discretization used for both parts of equation 1. If the dispersive term D is

sufficiently large when compared with the velocity u so that the grid Peclet number is sufficiently small, these numerical approximations perform well. In a simple one dimensional system, the grid Peclet number (Pe) is defined as:

$$\mathrm{Pe} = \frac{u\,\Delta h}{D}, \tag{5}$$

where Δh is the grid spacing. However, if the problem is advection dominated (i.e., has a high Peclet number), traditional solution methods are subject to nonphysical response, such as spurious oscillations near sharp solute fronts (Pe > 10). In this case, traditional methods of solution require very small grid spacings (Δh sufficiently small so that Pe < 10) to obtain accurate numerical approximations.

Operator-Splitting Methods

Several researchers, upon recognizing the importance of appropriately treating the hyperbolic part of the ADRE, have utilized characteristics-based solution techniques to develop methods that do not suffer from nonphysical results (Pinder and Cooper, 1970; Neuman, 1981; Douglas and Russell, 1982; Ewing and others, 1984). These methods approximate the advective component of transport using some type of a characteristics-based tracking algorithm, and approximate the dispersive component with a more standard finite difference or finite element method. More recently, Eulerian-Lagrangian Localized Adjoint Methods (ELLAM) have evolved from characteristics-based methods (Russell, 1985; Celia and others, 1990; Healy and Russell, 1993, 1998; Arbogast and Wheeler, 1995; Russell and Celia, 2002; Russell and others, 2003).

An alternative approach for approximating the advective step is to use what are termed Godunov-type methods, where the advective term is discretized by some form of a high-resolution finite volume method, and a finite element technique is used for the dispersive step (Bell and others, 1988; Dawson, 1990, 1991, 1993; Mazzia and others, 2000, 2002; James and Jawitz, 2007). Godunov techniques were originally developed to approximate purely hyperbolic nondissipative systems (Godunov, 1959; van Leer, 1979; Colella and Woodward, 1984). These methods are based on conservation of mass locally on each element. At each time step, the solution to the transport equation is approximated by a discontinuous piecewise polynomial, and the advective flux is approximated by a constant on each element edge (Dawson, 1993).

As Dawson (1993) pointed out, to solve the ADRE, it is logical to combine the Godunov solution technique for advection with a mixed finite element method for dispersion, using the lowest order Raviart-Thomas approximating space (Raviart and Thomas, 1977). The resulting methods, termed Godunov-mixed methods (GMMs), have several advantages over previous methods. These methods are numerically stable, introduce minimal numerical diffusion, and conserve mass both locally (i.e, on an element-by-element basis) and globally (Dawson, 1991, 1993; Mazzia and others, 2000; 2002). ELLAM and GMMs are collectively referred to as operator- or time-splitting techniques because they effectively "split" the time derivative between the advective and dispersive parts. This splitting results in two partial differential equations, which are solved sequentially at each time step. This permits independent discretizations to be applied to the two components, which in turn, allows a great deal of flexibility in selecting appropriate solution methods. For example, implicit and explicit time stepping can be applied to the diffusive and advective parts, respectively.

This work describes an operator-splitting algorithm that uses a Godunov method for the advective part of the ADRE (eq. 1) combined with a mixed finite element technique for the dispersive part. This technique is applied on an unstructured triangular mesh. This development closely follows the work of Mazzia and others (2000; 2002); significant differences arise only in the details of applying the Godunov method for the advective step. A computer model was implemented based on this algorithm, which was used to obtain the results (discussed later). Operation of the computer model is described in appendix 1.

Lastly, the method described in this document was extended and improved by James and Jawitz (2007), who developed a technique that more accurately approximates the advective step. Furthermore, they introduced a method that symmetrically splits the advective step so that half is applied before the dispersive part, and half afterwards. This allows a larger time step for the dispersive step without significantly diminishing accuracy.

Numerical Scheme Development

The specific form of the ADRE used for this model and the steps used to obtain a numerical approximation to the solution of the equation is discussed in the following sections. In the section immediately following, the splitting of the ADRE into advective and dispersive parts is discussed. The subsequent sections discuss the application of numerical approximations to the resulting split ADRE.

Transport Equations

Transport of a solute in a two-dimensional domain Ω with boundary Γ is described by the advection-dispersion-reaction equation:

$$\frac{\partial(\phi h c)}{\partial t} + \nabla\left(c\,h\,u - h\,D^{*}\cdot\nabla c\right) + h\,f_2 c = h\,f_1 c_1 \,, \tag{6}$$

where

t	is time,
$\mathbf{x}$	is the two-dimensional spatial coordinate with components (x_1, x_2);
$\phi(\mathbf{x},t)$	is the porosity of the medium (may equal 1 for surface-water applications, see fig. 1);
$h(\mathbf{x},t)$	is the water depth, or thickness of the saturated zone in ground-water flow [L];
$c(\mathbf{x},t)$	is the concentration [M L^{-3}];
$u(\mathbf{x},t)$	is the specific discharge (Darcy velocity in ground water) [L T^{-1}];
$D^* = D^*(u[\mathbf{x},t])$	is the hydrodynamic dispersion tensor (a function of u) [L^2 T^{-1}];
$f_1(\mathbf{x},t)$	is a source rate with associated concentration c_1 [T^{-1}]; and
$f_2(\mathbf{x},t)$	is a first-order decay rate [T^{-1}].

The density of the water is assumed to be constant. The domain is discretized into m triangles E_i, $i = 1, ..., m$ with n edges e_j, $j = 1, ..., n$.

A time-splitting GMM is used where the advective and diffusive parts of the governing equation 6 are discretized separately. Therefore, an advective flux $\mathbf{f}(\mathbf{x},t)$ term and a diffusive flux term $\mathbf{z}(\mathbf{x},t)$ (both [M L^{-1} T^{-1}]) are introduced:

$$\mathbf{f} - h\,u\,c = 0 \,, \tag{7}$$

$$\mathbf{z} + D\cdot\nabla c = 0 \,, \tag{8}$$

$$\frac{\partial(\phi h c)}{\partial t} + \nabla\cdot\left(\mathbf{f}+\mathbf{z}\right) = h\,f_1 c_1 - h\,f_2 c \,, \tag{9}$$

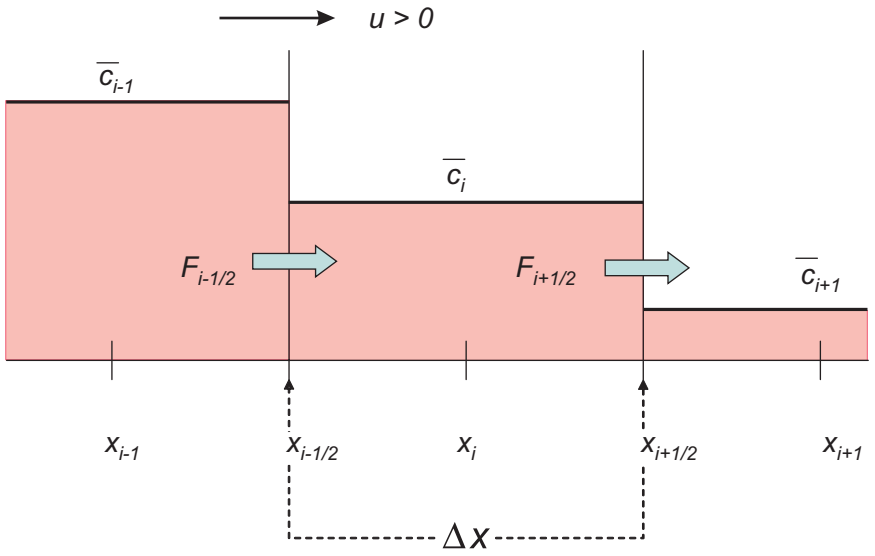


Figure 1. A simple one-dimensional solute advection problem. The spatial coordinate is x, Δx is the mesh spacing, $\overline{c}_i$ is the average concentration at x_i, u is the velocity, and $F_{i\pm1/2}$ is the approximation of mass flux at $x_{i\pm1/2}$.

where $D = h\,D^*$. The boundary conditions are specified for three situations: a known concentration at the boundary (Dirichlet), mixed (Robin; also known as a third-type boundary condition), and zero diffusive flux at the boundary (homogenous Neumann). These three conditions are written as:

$$c(\mathbf{x},t) = c_D \;\forall\, \mathbf{x} \in \Gamma_D\,, \tag{10}$$

$$z \cdot \mathbf{n} = \left(c_R(\mathbf{x},t) - c\right) h u \cdot \mathbf{n} \;\; \forall \mathbf{x} \in \Gamma_R\,, \tag{11}$$

$$z \cdot \mathbf{n} = 0 \;\; \forall \mathbf{x} \in \Gamma_N\,, \tag{12}$$

where $\mathbf{n}$ is the outward-pointing normal to the boundary and Γ_D, Γ_R, and Γ_N are the segments of the boundary with Dirichlet, Robin, and Neumann conditions, respectively. The known concentration on a Dirichlet boundary is given by c_D. The Robin boundary is useful for representing flow of water with a known concentration (c_R) from an external "reservoir" into the domain (where $u(x,t)\cdot\mathbf{n} < 0$) without specifying the concentration on the edge of the domain. The initial condition for the concentration c in Ω is specified by c_0:

$$c(\mathbf{x},0) = c_0(\mathbf{x})\,, \tag{13}$$

The fundamental idea in applying time-splitting to equation 9 is to first solve:

$$\frac{\partial(\phi h c)}{\partial t} = -\nabla \cdot \mathbf{f} + h f_1 c_1 - h f_2 c\,, \tag{14}$$

at the beginning of a time step. Then use the resulting solution, denoted by c^*, as an initial condition for the dispersive step:

$$\mathbf{z} + D \cdot \nabla c^* = 0\,, \tag{15}$$

$$\frac{\partial(\phi h c)}{\partial t} + \nabla \cdot \mathbf{z} = 0\,. \tag{16}$$

The source and sink terms are included with the advective step because it is more accurate to process these terms with the advective algorithm. The subsequent sections provide a detailed discussion of the solution technique applied to equation 6 with boundary and initial conditions 10-12.

Discretization and Calculation

The system of equations is solved using an explicit finite volume method to approximate the advective step and an implicit mixed finite element method to solve the dispersive step (Dawson 1993; Mazzia and others, 2000, 2002). The domain Ω is partitioned into m triangular elements E_i, $i = 1,\ldots, m$ with n edges e_j, $j = 1,\ldots, n$. The triangulation is denoted by $\mathcal{T}$. The concentration c, the diffusive flux $\mathbf{z}$, and the advective flux $\mathbf{f}$ are approximated using different approximation spaces. The different approximations for c and $\mathbf{z}$ are solved simultaneously in the dispersive step, hence the terminology "mixed finite element."

The concentration is approximated by functions that are constant on each element; thus, the resulting solution for c represents the average value of concentration within each element. Mathematically, the approximation space for the concentration on each element E_i is denoted by W_h, and on each element the approximation to concentration is represented by a real number α:

$$W_h(E) = \{\alpha \mid \alpha \in \mathbb{R}\}\,, \tag{17}$$

where $\mathbb{R}$ denotes the set of real numbers. This implies that the approximation to the concentration is discontinuous across element boundaries.

To represent the diffusive flux (a vector quantity), a more complex approximation space is required. This approximation space is composed of a set of vector-valued functions that have continuous zeroth spatial moments across each element boundary; that is, the integral along each element edge of the normal component (relative to the edge) of each vector-valued function is continuous between elements. The continuity of the zeroth moments ensures that the mass transfer due to diffusion/hydrodynamic dispersion balances from element to element. The vector space is denoted by $\mathbf{V}_h$ and functions from this space have three degrees of freedom on each element denoted by the scalars α_1, α_1, and β:

$$\mathbf{V}_h(E) = \left\{ \begin{bmatrix} \alpha_1 + \beta x_1 \\ \alpha_2 + \beta x_2 \end{bmatrix} \mid \alpha_i, \beta \in \mathbb{R} \right\}. \tag{18}$$

The approximation spaces W_h and $\mathbf{V}_h$ are together referred to as a mixed finite element space and were originally used by Raviart and Thomas (1977); this mixed finite element space is consequently referred to as the Raviart-Thomas space of order zero, or RT_0 (Raviart and Thomas, 1977; Brezzi and Fortin, 1991; Bergamaschi and Putti, 1999). A technique (termed *hybridization*) is discussed in a subsequent section, where the concentration on the edges of each element E will be approximated using an associated Lagrange multiplier space denoted by $\Lambda_h(\partial E)$, where ∂E denotes the boundary of E.

Using the approximating spaces W_h and $\mathbf{V}_h$ the approximations for $c(x,t)$ and $\mathbf{z}(x,t)$ are denoted by $\tilde{c}$ and $\tilde{\mathbf{z}}$, respectively, and are given by:

$$\tilde{c} = \sum_{j=1}^{m} c_j w_j,\ w_j \in W_h; \quad \tilde{\mathbf{z}} = \sum_{j=1}^{n} z_j \mathbf{v}_j,\ \mathbf{v}_j \in \mathbf{V}_h. \tag{19}$$

The basis function w_j is equal to 1 on element E_j and zero off of it, so each c_j is the coefficient representing the average concentration in E_j. Each basis function $\mathbf{v}_i$ is defined on edge e_j by the condition that the zeroth moment of $\mathbf{v}_i$ is equal to one on the ith edge, and 0 on all others:

$$\int_{e_j} \mathbf{v}_i \cdot \mathbf{n}_j \, ds = \delta_{ij} \quad i, j = 1, \ldots, n, \tag{20}$$

where $\mathbf{n}_j$ is the normal on e_j and $\delta_{ij} = 0$ if i ≠ j and $\delta_{ij} = 1$ if i = j. The basis function $\mathbf{v}_j$ is shared by the two elements sharing edge j, and the local support for $\mathbf{v}_j$ consists of those two elements. The quantity z_j represents the mass discharge (from dispersive flux only) normal to edge e_j. Each vector basis function has the form:

$$\mathbf{v}_i = \frac{1}{2|E_j|}(\mathbf{p} - \mathbf{p}_i) \quad i = 1,\ 2,\ 3, \tag{21}$$

where $|E_j|$ is the area of element E_j and $\mathbf{p} = [x_1, x_2]^T$ is a two-dimensional point within the element, and $\mathbf{p}_i$ is the ith vertex of E_j.

Discretization for the Advective Step

For the advective part of the ADRE, equation 14 is integrated over the time period $[t^k, t^{k+1}]$, with initial condition $\hat{c}^k = \tilde{c}^k$ (assuming the porosity ϕ remains constant with respect to time) gives:

$$\phi h^{k+1} \hat{c}^{k+1} = \phi h^k \hat{c}^k - \int_{t^k}^{t^{k+1}} \nabla \cdot \mathbf{f}(\mathbf{x}, t, \hat{c}) \, dt + \int_{t^k}^{t^{k+1}} h f_1 c_1 \, dt - \int_{t^k}^{t^{k+1}} h f_2 \hat{c} \, dt, \tag{22}$$

where $\hat{c}^{k+1}$ is the approximation at the end of the advective step, and is used as the initial condition for the dispersive step. It is assumed that the water depths and flows are known for both the current and the next time step, thus h^{k+1} is known at the kth step. The integration of equation 22 over a single element E_j gives:

$$\begin{aligned} \int_{E_j} \phi h^{k+1} \hat{c}^{k+1} \, d\Delta = {} & \int_{E_j} \phi h^k \hat{c}^k \, d\Delta - \int_{E_j} \int_{t^k}^{t^{k+1}} \nabla \cdot \mathbf{f}(\mathbf{x}, t, \hat{c}) \, dt \, d\Delta \\ & + \int_{E_j} \int_{t^k}^{t^{k+1}} h f_1 c_1 \, dt \, d\Delta - \int_{E_j} \int_{t^k}^{t^{k+1}} h f_2 \hat{c} \, dt \, d\Delta, \end{aligned} \tag{23}$$

where $d\Delta$ indicates that the integration is over the area of the triangle. The application of the divergence theorem to the integral containing $\nabla \cdot \mathbf{f}$, and the assumption that c_1, f_1 and f_2 are constant in the interval $[t^{k+1}, t^k]$ gives:

$$\int_{E_j} \phi h^{k+1}\, \hat{c}^{k+1}\, d\Delta = \int_{E_j} \phi h^k\, \hat{c}^k\, d\Delta - \int_{t^k}^{t^{k+1}} \int_{\partial E_j} \nabla \cdot \mathbf{f}\left(\mathbf{x}, t, \hat{c}\right) \cdot \mathbf{n}\, ds\, \mathrm{d}t + c_1 f_1 \int_{E_j} \int_{t^k}^{t^{k+1}} h\ \mathrm{d}t\, d\Delta - f_2 \int_{E_j} \int_{t^k}^{t^{k+1}} h\hat{c}\, \mathrm{d}t\, d\Delta \quad , \tag{24}$$

where $\mathbf{n}$ is the outward pointing normal of E_j and ds is an infinitesimal distance along the boundary ∂E_j. Execution of the integration over the element gives an expression for the mass of solute m_j in each element E_j:

$$m_j^{k+1} = m_j^k - \int_{t^k}^{t^{k+1}} \int_{\partial E_j} \mathbf{f}\left(\mathbf{x}, t, \hat{c}\right) \cdot \mathbf{n}\, ds\, \mathrm{d}t + \Delta t\, c_1 f_1 \left|E_j\right| \frac{h_j^{k+1} + h_j^k}{2} - \Delta t\, f_2\, \frac{m_j^{k+1} + m_j^k}{2} \quad , \tag{25}$$

where a midpoint approximation is applied for the integration over time for the last two terms, and h_j^k is the average depth/head over E_j at the kth time step. Equation 25 is a statement that the change in mass in element E_j is equal to the mass entering and leaving across the edges of the element, plus any sources and minus any first-order decay within the element. This is rewritten as:

$$m_j^{k+1} = \left(1 + \frac{\Delta t\, f_2}{2}\right)^{-1} \left(m_j^k - \int_{t^k}^{t^{k+1}} \int_{\partial E_j} \mathbf{f}\left(\mathbf{x}, t, \hat{c}\right) \cdot \mathbf{n}\, ds\, \mathrm{d}t + \Delta t\, c_1 f_1 \left|E_j\right| \bar{h}_j - \frac{\Delta t\, f_2}{2} m_j^k \right) \quad , \tag{26}$$

where

$$\bar{h}_j = \frac{\left(h_j^{k+1} + h_j^k\right)}{2} \quad , \tag{27}$$

The accuracy of the approximation for the advective part of the ADE is directly dependent on the accurate approximation of the term:

$$\int_{t^k}^{t^{k+1}} \int_{\partial E_j} \mathbf{f}\left(\mathbf{x}, t, \hat{c}\right) \cdot \mathbf{n}\, \mathrm{d}s\, \mathrm{d}t \quad . \tag{28}$$

The term $F_i^{k+1/2}\left(\mathbf{x}, \hat{c}^{k+1/2}\right)$ is used to denote the numerical approximation of the total mass flux $\mathbf{f} \cdot \mathbf{n}_i$ across side i (i = 1, 2, 3) of E_j at time $t^{k+1/2}$ so that:

$$\sum_{i=1}^{3} F_i^{k+1/2}\left(\mathbf{x}_j, \hat{c}^{k+1/2}\right) \approx \frac{1}{\Delta t} \int_{t^k}^{t^{k+1}} \int_{\partial E_j} \mathbf{f}\left(\mathbf{x}, t, \hat{c}\right) \cdot \mathbf{n}\, \mathrm{d}s\, \mathrm{d}t \quad , \tag{29}$$

where

$$t^{k+1/2} = \frac{\left(t^k + t^{k+1}\right)}{2} \quad . \tag{30}$$

The approximation for the mass in E_j becomes:

$$m_j^{k+1} = \left(1 + \frac{\Delta t\, f_2}{2}\right)^{-1} \left(m_j^k - \Delta t \sum_{i=1}^{3} F_i^{k+1/2}\left(\mathbf{x}_j, \hat{c}^{k+1/2}\right) + \Delta t\, c_1 f_1 \left|E_j\right| \bar{h}_j - \frac{\Delta t\, f_2}{2} m_j^k \right) \quad . \tag{31}$$

Calculation of the Numerical Advective Flux

The primary difference between the approach described herein and that of Mazzia and others (2000; 2002) is in the determination of the numerical advective flux term. Mazzia and others (2000; 2002) relied on the formulation of Liu (1993) that constructs a two-dimensional interpolation for the concentration in each element based on the concentration in the surrounding elements. Because each triangle is surrounded by three adjacent triangles (except along boundaries), this method requires constructing three different interpolations and using an algorithm to select the best one.

In contrast, the numerical flux approximation presented herein does not rely on an explicit geometric construction for element concentration. This method determines the flux approximation at each element edge by examining the analogous one-dimensional problem and generalizing it to a two-dimensional problem by applying a one-dimensional problem at each element edge. For a one-dimensional, solute, cell-centered advection problem (fig. 1) with steady-state velocity $u > 0$, Godunov's method gives the time averaged flux contribution between the ith and i+1th cell as:

$$F_{i+1/2}^{k} = u\,\overline{c}_i^{\,k} + \frac{u}{2}\left(1 - u\frac{\Delta t}{\Delta x}\right)\left(\overline{c}_i^{\,k} - \overline{c}_{i+1}^{\,k}\right), \tag{32}$$

where $\overline{c}_i^{\,k}$ is the volume averaged solute concentration in the ith cell at time step k, Δt is the time step, and Δx is the width of a cell. The first term on the right-hand side of equation 32 is a first-order upwinding term, whereas the second is a second-order correction. With no second-order correction, the total mass of solute moved from cell i to cell i+1 over one time step is:

$$\Delta t\, F_{i+1/2}^{k} = \Delta t\, u\, \overline{c}_i^{\,k}, \tag{33}$$

which is the average concentration multiplied by the volume of water transferred. Now, consider the second-order term in equation 32. Because $u\,\Delta t$ represents the volume of water moved from cell i to i+1, the term $u\,\Delta t/\Delta x$ is the fraction of water volume from cell i that moves into cell i+1. As Δt increases, this fraction approaches 1 and so the contribution from the second-order term diminishes. If the entire volume of water in cell i moves into cell i+1 over the time Δt, the contribution from the second-order term will vanish because the mean solute concentration times the volume of water in cell i is exactly the amount of solute that should be moved. Using the numerical flux approximation in equation 32, second-order results are obtained when the solution is sufficiently smooth. However, the approximation can give spurious nonphysical oscillations near discontinuities or near-discontinuities, such as sharp solute concentration fronts. To prevent this response, a limiter is used, which diminishes the contribution of the second-order term near discontinuities. For the case of representations such as equation 32, these are termed flux limiters because they limit the higher order flux contributions between cells. Usually, the limiter is some function, $\zeta(\theta)$, of the magnitude of jumps in concentration averages (denoted by θ) across an element. The resulting flux approximation takes the form:

$$F_{i+1/2}^{k} = u\,\overline{c}_i^{\,k} + \frac{u}{2}\left(1 - u\frac{\Delta t}{\Delta x}\right)\zeta(\theta)\left(\overline{c}_i^{\,k} - \overline{c}_{i+1}^{\,k}\right). \tag{34}$$

The flux limiter ranges from 0 (when the jumps in solute concentration across an element are large), to one (when the solution is sufficiently smooth). In this way, accuracy is maintained when the solution is smooth, and reduces to pure upwinding near sharp fronts to maintain stability.

The numerical flux approximation developed in almost the same manner as for the one-dimensional problem. First, conventions were given: water flow across an edge into an element is negative relative to the element; outflow is positive. The total inflow into an element across its edges is denoted by v_{in}; the total outflow is v_{out}. Now, given a volume of water $v_{j,i,}$ moving from element E_j to element E_i, the total mass flux across $e_{j,i}$ (the shared edge between E_j and E_i) is approximated as:

$$F_{j,i}^{k+1/2} = \frac{v_{j,i}}{\Delta t}\left[\hat{c}_j^{\,k} + \frac{1}{2}\left(1 - R_j\right)\zeta(\theta)\left(\hat{c}_i^{\,k} - \hat{c}_j^{\,k}\right)\right], \tag{35}$$

where $R_j = v_{j,i}/V_j$ is the fraction of volume V_j in E_j advected across $e_{j,i}$, and $\hat{c}_j^{\,k}$ is the volume-averaged concentration in each element, given by:

$$\hat{c}_j^{\,k} = \frac{1}{\phi h^k \left|E_j\right|}\int_{E_j} \phi h^k \hat{c}\left(\mathbf{x}, t^k\right) \mathrm{d}\Delta \quad \mathbf{x} \in E_j. \tag{36}$$

The function θ that measures the jumps in solute concentration across an element E_j is defined as:

$$\theta_j = \frac{\sum_{i|v_i<0} \frac{v_i}{|v_{\text{in}}|}\left(\hat{c}_i - \hat{c}_j\right)}{\sum_{i|v_i>0} \frac{v_i}{v_{\text{out}}}\left(\hat{c}_j - \hat{c}_i\right)}, \tag{37}$$

where the sums in the numerator are only taken over those adjacent elements E_i where the flow is from E_i into E_j, and the sums in the denominator are only over those adjacent elements E_i where the flow is out of E_j. If there is no flow from E_j to any other elements, θ is set equal to 1. The flux limiter used here is the van Leer limiter:

$$\zeta = \frac{\theta + |\theta|}{1 + |\theta|}. \tag{38}$$

Thus, $0 < \zeta < 2$ for all values of θ. An element E_j surrounded by three elements, E_i, E_l, and E_p is shown in figure 2. In this case, flow is from elements E_l and E_p into E_j, and then into E_i. The sum of fluxes is:

$$\begin{aligned} &v_{j,i}\left[\hat{c}_j^k + \frac{1}{2}\left(1 - R_j\right)\zeta\left(\theta_j\right)\left(\hat{c}_i^k - \hat{c}_j^k\right)\right] - v_{l,j}\left[\hat{c}_l^k + \frac{1}{2}\left(1 - R_l\right)\zeta\left(\theta_l\right)\left(\hat{c}_j^k - \hat{c}_l^k\right)\right] \\ &-v_{p,j}\left[\hat{c}_p^k + \frac{1}{2}\left(1 - R_p\right)\zeta\left(\theta_p\right)\left(\hat{c}_j^k - \hat{c}_p^k\right)\right] \end{aligned}. \tag{39}$$

The approximation of the mass after the advective step, m_j^{k+1}, becomes:

$$\begin{aligned} m_j^{k+1} = \left(1 + \frac{\Delta t\, f_2}{2}\right)^{-1} \Bigg\{ & m_j^k - \Delta t\, v_{j,i}\left[\hat{c}_j^k + \frac{1}{2}\left(1 - R_j\right)\zeta\left(\theta_j\right)\left(\hat{c}_i^k - \hat{c}_j^k\right)\right] \\ & + \Delta t\, v_{l,j}\left[\hat{c}_l^k + \frac{1}{2}\left(1 - R_l\right)\zeta\left(\theta_l\right)\left(\hat{c}_j^k - \hat{c}_l^k\right)\right] \\ & + \Delta t\, v_{p,j}\left[\hat{c}_p^k + \frac{1}{2}\left(1 - R_p\right)\zeta\left(\theta_p\right)\left(\hat{c}_j^k - \hat{c}_p^k\right)\right] \\ & + \Delta t\, c_1 f_1 \left|E_j\right| \bar{h}_j - \frac{\Delta t\, f_2}{2} m_j^k \Bigg\} \end{aligned}. \tag{40}$$

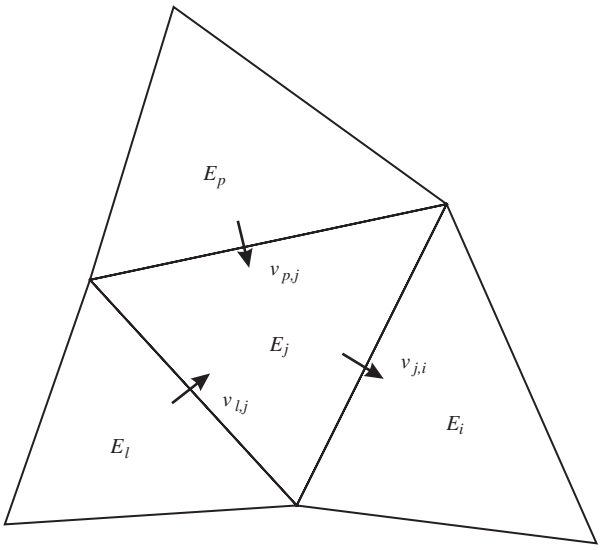


Figure 2. Flow between an element E_j and the surrounding three elements E_l, E_i, and E_p. The flow is denoted by $v_{a,b}$, where *a* and *b* represent the "to" and "from" elements, respectively (here, *i*, *j*, *l*, or *p*).

This value for the volume-averaged concentration $\hat{c}_j^{k+1}$ in each element is then given by:

$$\hat{c}_j^{k+1} = \frac{1}{\phi h_j^{k+1} \left| E_j \right|} m_j^{k+1} \quad , \tag{41}$$

This becomes the initial condition for the dispersion step in equations 15 and 16.

Hybridized Mixed Finite Element for Dispersion

A mixed finite element technique was applied to discretize the mixed form of the dispersive part of the ADE given in equations 15 and 16. This method uses functions w from the approximation space W_h (eq. 17) to approximate the concentration $c(\mathbf{x},t)$ within each element and functions $\mathbf{v}$ from the approximation space $\mathbf{V}_h$ (eq. 18) to approximate the dispersive flux $\mathbf{z}(\mathbf{x},t)$ across element edges.

Multiplying equations (15) and (16) by the functions $\mathbf{v}_i$ in $\mathbf{V}_h$ and w_i in W_h, respectively, and integrating over Ω, give the equations:

$$\int_\Omega D^{-1} \cdot \tilde{\mathbf{z}} \cdot \mathbf{v}_i \, d\Omega + \int_\Omega \nabla \tilde{c} \cdot \mathbf{v}_i \, d\Omega = 0 \quad i = 1, \ldots, n \quad , \tag{42}$$

$$\int_\Omega \frac{\partial \left(\phi h \tilde{c} \right)}{\partial t} w_i \, d\Omega + \int_\Omega \nabla \cdot \tilde{\mathbf{z}} \, w_i \, d\Omega = 0 \quad i = 1, \ldots, m \quad . \tag{43}$$

The discretized version of this system is a saddle point problem and consequently is difficult to solve efficiently (Benzi and others, 2005). One method to remedy this is to convert the problem into one with a positive definite coefficient matrix by introducing a space of Lagrange multipliers (denoted by Λ_h) for the concentration on the edge of each element (Brezzi and Fortin, 1991). This technique is referred to as hybridization. The Lagrange multipliers represent the average values of concentration on the edges of each element E, which is called the *trace* of the concentration $c(\mathbf{x},t)$ on ∂E (Brezzi and Fortin, 1991; Chavent and Roberts, 1991). The Lagrange multipliers are denoted by $\tilde{\lambda}$ and the space Λ_h is simply the space of constant functions on each edge in the triangulation:

$$\Lambda_h \left(\partial E \right) = \left\{ \mu \mid \mu \in \mathbb{R} \right\} \quad . \tag{44}$$

A basis function μ_j from this space is equal to 1 on edge e_j and 0 elsewhere.

The primary purpose of hybridization is to decouple the dispersive flux basis functions between elements. In the non-hybridized system, a single basis function $\mathbf{v}_j$ on an edge e_j is shared between the two elements that share that edge. This basis function approximates the dispersive flux that crosses e_j as it passes from one element to the adjoining element; thus, inter-element mass balance is assured. The hybridization technique replaces this one basis function for dispersive flux with two separate basis functions, each associated with only one of the adjoining elements, and has support only on that element (that is, each basis function is 0 off its "own" element). The dispersive flux $\tilde{\mathbf{z}}^k$ on element E_k is written as the sum of the three dispersive flux basis functions $\mathbf{v}_i$ (i = 1, 2, 3) defined for that element and the associated coefficients $\tilde{z}_i$ for that element:

$$\tilde{\mathbf{z}}^k = \sum_{j=1}^{3} \tilde{z}_j^k \mathbf{v}_j^k \quad . \tag{45}$$

Although this appears to significantly increase the number of basic functions required (thereby increasing computational time), the hybridization process will leave the final size of the linear system unaffected. Because the support for the vector basis functions is limited to a single element, for a particular basis function $\mathbf{v}_i^k$ defined on element E_k, the integral of $\mathbf{v}_i^k$ over the entire domain Ω is equal to the integral over E_k alone:

$$\int_\Omega v_i^k \, d\Omega = \int_{T_k} \mathbf{v}_i^k \, dA \quad , \tag{46}$$

Consequently, any two basis functions $\mathbf{z}_r$ and $\mathbf{z}_s$ that share an edge e_j but are defined separately on the two elements that share e_j, the associated coefficients $\tilde{z}_r$ and $\tilde{z}_s$ will not necessarily be equal; thus, decoupling the dispersive flux coefficients between adjoining elements (Beckie and others, 1993; Bergamaschi and Putti, 1999). To assure mass balance between the two elements by ensuring continuity of dispersive flux across the shared edge, the condition that across each edge $e_j = \partial E_i \cap \partial E_k$ was added:

$$\tilde{\mathbf{z}}^i \cdot \mathbf{n}_j^i + \tilde{\mathbf{z}}^k \cdot \mathbf{n}_j^k = 0 \,, \tag{47}$$

where $\mathbf{n}_j^i$ and $\mathbf{n}_j^k$ are the outward-pointing normals of elements i and k, respectively, on e_j. The integration over the shared edges in the triangulation gives:

$$\int_{e_j} \tilde{\mathbf{z}}^i \cdot \mathbf{n}_j^i \, ds + \int_{e_j} \tilde{\mathbf{z}}^k \cdot \mathbf{n}_j^k \, ds = 0 \quad j = 1, \ldots, n' \,, \tag{48}$$

where *n'* is the number of shared interior edges (those not lying on a Dirichlet boundary).

When the separated vector basis function $\mathbf{v}^k$ is applied, the system given in equations 15 and 16 can then be written in the mixed-hybrid form:

$$\int_{E_k} D_k^{-1} \cdot \tilde{\mathbf{z}} \cdot \mathbf{v}_i^k \, d\Delta - \int_{E_k} c \nabla \cdot \tilde{\mathbf{v}}_i^k \, d\Delta + \int_{\partial E_k \setminus \Gamma_D^k} \tilde{\lambda} \mathbf{v}_i^k \cdot \mathbf{n}^k \, ds = \int_{\Gamma_D^k} c_D \mathbf{v}_i^k \cdot \mathbf{n}^k \, ds \quad i = 1, \ldots, n; \; k = 1, \ldots, m \,, \tag{49}$$

$$\int_{E_k} \phi_k \frac{\partial h_k \tilde{c}}{\partial t} w^k \, d\Delta + \int_{E_k} \nabla \cdot \tilde{\mathbf{z}} \, w^k \, d\Delta = 0 \quad k = 1, \ldots, m \,, \tag{50}$$

$$\int_{e_j} \tilde{\mathbf{z}}^i \cdot \mathbf{n}_j^i \, ds + \int_{e_j} \tilde{\mathbf{z}}^k \cdot \mathbf{n}_j^k \, ds = 0 \quad j = 1, \ldots, n' \,. \tag{51}$$

The term $\Gamma_D{}^k$ refers to the part of the Dirichlet boundary (where the concentration is given by c_D) that intersects E. The term $\partial E_k \setminus \Gamma_D{}^k$ in the lower limit of the integral in equation 49 indicates that the integration is over all the edges of element k, except those that lie on Γ_D. The function $\tilde{\lambda}$ in the boundary integrals is the Lagrange multiplier representing the concentration on the element boundaries discussed above. A linear system was constructed to solve for λ alone, and back-substitutes element by element to obtain element concentrations and dispersive fluxes. This eliminates the additional computation required when the additional dispersive flux basis functions are introduced.

The integration of each inner product in equations 49 to 51 element by element gives a set of block diagonal submatrices which are denoted $A = \mathrm{diag}[A_1, \ldots, A_m]$, $B = \mathrm{diag}[B_1, \ldots, B_m]$, and $G = \mathrm{diag}[G_1, \ldots, G_m]$. For the kth element, these submatrices are:

$$A_k \equiv a_{ij}^k = \int_{E_k} D_k^{-1} \, \mathbf{v}_i \cdot \mathbf{v}_k \, d\Delta; \quad B_k \equiv b_i^k = \int_{E_k} \nabla \cdot \mathbf{v}_i \, d\Delta \,, \tag{52}$$

$$G_k^* \equiv g_k = \int_{E_k} \phi_k \, d\Delta; \tag{53}$$

In addition, the matrix C (not block diagonal) results from the last integral on the left-hand side of equations 49 and 51:

$$C = \int_{\partial E_k \setminus \Gamma_D^k} \mathbf{v}_i \cdot \mathbf{n}_i^k \, ds \,. \tag{54}$$

This yields the system of equations:

$$\begin{aligned} A\tilde{\mathbf{z}} - B\tilde{\mathbf{c}} + C\tilde{\boldsymbol{\lambda}} &= -\mathbf{d} \\ B^T\tilde{\mathbf{z}} + G^* \frac{\partial \mathbf{h}\,\tilde{\mathbf{c}}}{\partial t} &= 0 \\ C^T\tilde{\mathbf{z}} &= 0 \end{aligned}, \tag{55}$$

where $\tilde{\mathbf{z}} \equiv \tilde{z}_i^k$ for $i = 1, 2, 3$ and $k = 1,\ldots, m$; and $\tilde{\lambda} \equiv (\tilde{\lambda}_i)$ for $i = 1,\ldots,$ n', and **h** is the vector of water depths in each element (assumed known at time k and k+1). The right-hand side term **d** results from integrating the right-hand side of equation 49 and is defined as:

$$\mathbf{d} \equiv (d_i) = \int_{\Gamma_D} c_D\, \mathbf{v}_i \cdot \mathbf{n}\, ds \,. \tag{56}$$

The time derivative is discretized using a first-order accurate backward Euler difference formula:

$$\frac{\partial h\tilde{c}}{\partial t} \approx \frac{\left(h^{k+1}\,\tilde{c}^{k+1} - h^k\,\hat{c}^{k+1}\right)}{\Delta t}, \tag{57}$$

where $\hat{c}^{k+1}$ is the intermediate solution after applying the advective step given by equation 41. For notational convenience, this was written as $\tilde{c}^k \equiv \hat{c}^{k+1}$. The fully discretized system of equations for the variables $\tilde{\mathbf{z}}$, $\tilde{\mathbf{c}}$, $\tilde{\boldsymbol{\lambda}}$ at the k+1 time step can be written in matrix form as:

$$\begin{pmatrix} A & -B & C \\ B^T & G & 0 \\ C^T & 0 & 0 \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{z}}^{k+1} \\ \tilde{\mathbf{c}}^{k+1} \\ \tilde{\boldsymbol{\lambda}}^{k+1} \end{pmatrix} = \begin{pmatrix} -\mathbf{d} \\ G\tilde{\mathbf{c}}^k \\ 0 \end{pmatrix}, \tag{58}$$

where $G = \mathbf{h}\, G^*/\Delta t$.

Simplification of the Solution Matrices

A technique known as static condensation (also known as a Schur complement method) can be applied to the system to reduce the number of variables to solve the linear system for $\tilde{\lambda}$ alone. The hybrid development is critical to this step, since it reduces the submatrix A to a block diagonal matrix that is easily invertible.

Multiplying the first row of submatrices out in equation 58 gives the equation:

$$A\tilde{\mathbf{z}}^{k+1} - B\tilde{\mathbf{c}}^{k+1} + C\tilde{\boldsymbol{\lambda}}^{k+1} = -\mathbf{d}\,. \tag{59}$$

Solving this for $\tilde{\mathbf{z}}^{k+1}$ yields:

$$\tilde{\mathbf{z}}^{k+1} = A^{-1}\left(B\tilde{\mathbf{c}}^{k+1} - C\tilde{\boldsymbol{\lambda}}^{k+1} - \mathbf{d}\right). \tag{60}$$

This leaves the remaining two equations:

$$\begin{aligned} B^T\left[A^{-1}\left(B\tilde{\mathbf{c}}^{k+1} - C\tilde{\boldsymbol{\lambda}}^{k+1} - \tilde{\mathbf{d}}\right)\right] + G\tilde{\mathbf{c}}^{k+1} &= G\tilde{\mathbf{c}}^k \\ C^T\left[A^{-1}\left(B\tilde{\mathbf{c}}^{k+1} - C\tilde{\boldsymbol{\lambda}}^{k+1} - \tilde{\mathbf{d}}\right)\right] &= 0 \end{aligned}\,. \tag{61}$$

Upon expansion, equation 61 gives the system:

$$\begin{aligned} B^T A^{-1} B\tilde{\mathbf{c}}^{k+1} + G\tilde{\mathbf{c}}^{k+1} - B^T A^{-1} C\tilde{\boldsymbol{\lambda}}^{k+1} &= B^T A^{-1}\mathbf{d} + G\tilde{\mathbf{c}}^{k} \\ C^T A^{-1} B\tilde{\mathbf{c}}^{k+1} - C^T A^{-1} C\tilde{\boldsymbol{\lambda}}^{k+1} &= C^T A^{-1}\mathbf{d} \end{aligned} . \tag{62}$$

This can be written in matrix form as:

$$\begin{pmatrix} B^T A^{-1} B + G & -B^T A^{-1} C \\ C^T A^{-1} B & -C^T A^{-1} C \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{c}}^{k+1} \\ \tilde{\boldsymbol{\lambda}}^{k+1} \end{pmatrix} = \begin{pmatrix} B^T A^{-1}\mathbf{d} + G\mathbf{c}^{k+1} \\ C^T A^{-1}\mathbf{d} \end{pmatrix} . \tag{63}$$

Now, solving the top row of equation 63 for $\tilde{\mathbf{c}}^{k+1}$ yields:

$$\tilde{\mathbf{c}}^{k+1} = \left(B^T A^{-1} B + G\right)^{-1} \left(B^T A^{-1} C\tilde{\boldsymbol{\lambda}}^{k+1} + B^T A^{-1}\mathbf{d} + G\tilde{\mathbf{c}}^{k}\right) . \tag{64}$$

Substituting this into the bottom row of equation 63 gives:

$$\begin{aligned} C^T A^{-1} B\left[\left(B^T A^{-1} B + G\right)^{-1} \left(B^T A^{-1} C\tilde{\boldsymbol{\lambda}}^{k+1} + B^T A^{-1}\mathbf{d} + G\tilde{\mathbf{c}}^{k}\right)\right] & \\ -C^T A^{-1} C\tilde{\boldsymbol{\lambda}}^{k+1} = C^T A^{-1}\mathbf{d} & \end{aligned} . \tag{65}$$

Solving this for $\tilde{\boldsymbol{\lambda}}^{k+1}$ gives:

$$\begin{aligned} &C^T \left[A^{-1} B\left(B^T A^{-1} B + G\right)^{-1} \left(B^T A^{-1}\right) - A^{-1}\right] C\tilde{\boldsymbol{\lambda}}^{k+1} = \\ &-C^T A^{-1} B\left[\left(B^T A^{-1} B + G\right)^{-1} \left(B^T A^{-1}\mathbf{d} + G\tilde{\mathbf{c}}^{k}\right)\right] + C^T A^{-1}\mathbf{d} \end{aligned} \tag{66}$$

Letting $S = A^{-1} B$, (so $S^T = B^T$, A^{-1}), and $J = S^T B + G$, this becomes:

$$\begin{aligned} &C^T \left[S J^{-1} S^T - A^{-1}\right] C\tilde{\boldsymbol{\lambda}}^{k+1} = \\ &C^T \left[\left(A^{-1} - S J^{-1} S^T\right)\mathbf{d} - S J^{-1}\left(G\tilde{\mathbf{c}}^{k}\right)\right] . \end{aligned} \tag{67}$$

Finally, letting $M = A^{-1} - S J^{-1} S^T$ gives the reduced linear equation for $\tilde{\boldsymbol{\lambda}}^{k+1}$:

$$C^T M C\tilde{\boldsymbol{\lambda}}^{k+1} = C^T \left[S J^{-1}\left(G\tilde{\mathbf{c}}^{k}\right) - M\mathbf{d}\right] . \tag{68}$$

This system is much smaller than the original in equation 58, with one unknown for each edge in the triangulation. The solution for $\tilde{\boldsymbol{\lambda}}^{k+1}$ is then used in a back-substitution procedure to find solutions for the element concentration and the dispersive flux. The advantage is that these can be found on an element-by-element basis, obviating the need for solving a large linear system. Equation 64 is used to obtain a solution for the updated concentration approximation $\tilde{\mathbf{c}}^{k+1}$:

$$\tilde{\mathbf{c}}^{k+1} = J^{-1}\left[S^T\left(C\tilde{\boldsymbol{\lambda}}^{k+1} + \mathbf{d}\right) + G\tilde{\mathbf{c}}^{k}\right] . \tag{69}$$

Lastly, a solution for the dispersive flux $\tilde{\mathbf{z}}^{k+1}$ is provided from equation 60, again on an element-by-element basis:

$$\tilde{\mathbf{z}}^{k+1} = A^{-1}\left(B\tilde{\mathbf{c}}^{k+1} - C\tilde{\boldsymbol{\lambda}}^{k+1} - \mathbf{d}\right) . \tag{70}$$

Notes on the Linear System

The linear system and equations describing the solution method is considerably less complex than it first appears. The matrix A is block diagonal, and for the two-dimensional system described above each block, A_k is 3 x 3:

$$\begin{aligned} A_k = a_{ij}^k &= \int_{T_k} D_k^{-1}\, \mathbf{v}_i \cdot \mathbf{v}_k \, d\Delta \\ &= \int_{T_k} \frac{1}{4\left|E_k\right|^2} \left(p - p_i\right) \cdot D_k^{-1} \cdot \left(p - p_j\right) d\Delta; \quad i, j = 1, 2, 3 \end{aligned} \tag{71}$$

Thus, A is a $3m$ x $3m$ matrix that looks like:

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & A_m \end{bmatrix}. \tag{72}$$

Because everything off the main block diagonal is 0, finding the inverse requires only inverting each 3 x 3 subblock:

$$A^{-1} = \begin{bmatrix} A_1^{-1} & 0 & \cdots & 0 \\ 0 & A_2^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & A_m^{-1} \end{bmatrix}. \tag{73}$$

The matrix B is also block diagonal, but each subblock is 3 x 1. Recalling the definition of B:

$$B_k \equiv b_i^k = \int_{T_k} \nabla \cdot \mathbf{v}_i \, d\Delta \tag{74}$$

This is the discretized divergence operator on each element. From the definition of $\mathbf{v}_i$, it can easily be determined that each entry $b_i = 1$, so each $B_k = [1, 1, 1]^{\mathrm{T}}$ and B is a $3m$ x m matrix, and B^{T} is an m x $3m$ matrix:

$$B = \begin{bmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_m \end{bmatrix}; \quad B^T = \begin{bmatrix} B_1^{\mathrm{T}} & 0 & \cdots & 0 \\ 0 & B_2^{\mathrm{T}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_m^{\mathrm{T}} \end{bmatrix}. \tag{75}$$

Thus, $A^{-1}B$ is a $3m$ x m matrix, and $B^{\mathrm{T}}A^{-1}B$ is m x m:

$$A^{-1}B = \begin{bmatrix} A_1^{-1}B_1 & 0 & \cdots & 0 \\ 0 & A_2^{-1}B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_m^{-1}B_m \end{bmatrix};$$

$$B^T A^{-1} B = \begin{bmatrix} B_1^{\mathrm{T}} A_1^{-1} B_1 & 0 & \cdots & 0 \\ 0 & B_2^{\mathrm{T}} A_2^{-1} B_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B_m^{\mathrm{T}} A_m^{-1} B_m \end{bmatrix}. \tag{76}$$

Calculating each subblock of $B^T A^{-1} B$ requires no multiplication because each matrix product $A_k^{-1} B_k$ sums each row of A_k^{-1} and $B_k^T A_k^{-1} B_k$ is 1 x 1 (that is, a scalar) and is simply a sum of all the entries of A_k^{-1}. Consequently, the matrix J is a diagonal matrix and J^{-1} is trivial to determine.

The matrix C has no simple block structure, but recalling the definition of each basis function $\mathbf{v}_i$ from equation 20 and the construction of C from equation 54, it can be seen that C is a matrix of zeros and ones. Each row of C has a maximum of two non-zero entries corresponding to the two adjacent basis functions for dispersive flux $\mathbf{v}_i$ and $\mathbf{v}_j$ that share a common edge in the triangulation. Thus, C maps the local edges of each triangle E to the global matrix numbering scheme. Because each entry is either a 0 or 1, no multiplication is required, and entries are taken from the local element matrices and added to the left-hand side of equation 68.

Verification of the Algorithm

The accuracy of the algorithm was verified using a suite of numerical tests. The first of these tests consisted of a series of simulations modeling one-dimensional transport, such as could happen in a column or in a long but narrow wetland. The modeled values were compared with a known exact solution to determine the error. A series of progressively finer meshes was used in these tests, and the reduction of error was measured when passing from one mesh to the next finest. Two sets of dispersion coefficients were tested. A similar testing methodology was used by James and Jawitz (2007) to demonstrate the greater accuracy of the extension to the method discussed here.

A two-dimensional domain was modeled in the second set of tests. Solute was released at a point within the domain, and again the modeled values were compared with an exact solution to determine the error. These tests were conducted on a series of meshes. Two sets of dispersion coefficients were used.

One-Dimensional Problem

The first of these tests compared the analytical solution given in equation 4 to the one-dimensional problem described by equation 1 with the numerical approximation to the solution on a two-dimensional mesh. The analytical solution given in equation 4 is the solution for transport in a semi-infinite column, with a Robin (type three) boundary condition at $x = 0$ with $c_R = 1$, and the far-field condition $c(x, t) \rightarrow 0$ as $x \rightarrow \infty$. This was numerically approximated using a rectangular domain of length $x_1 = 1$ m, $x_2 = 0.1$ m, with the same type three boundary condition at $x_1 = 0$, uniform velocity $u = (u_1, 0)$, depth $h = 1$ m, and with dispersion tensor:

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}. \tag{77}$$

The initial condition was $c(x_1, 0) = 0$. A Neumann boundary condition was used for the boundary condition on the right side of the domain at $x_1 = 1$. To negate the effects of the differences in the boundary conditions on the right side of the domain, the ending time of the simulation was small enough to ensure that the difference between the exact solution and the approximation at $x_1 = 1$ was negligible. The simulations were conducted on a series of refined meshes (each refinement is considered a different "level") to evaluate the numerical convergence response of the algorithm. This simulation was meant to closely follow the similar one-dimensional test of Mazzia and others (2000; 2002), although the meshes used here were coarser at the lowest level. It is the same as the first simulation discussed in James and Jawitz (2007). The coarsest mesh subdivided the x_1 direction into 10 equal sections and the x_2 direction into 2, and each resulting rectangle was subdivided into two triangles. This mesh was subsequently refined to generate a series of meshes at 20 x 4, 40 x 8, 80 x 16, 160 x 32, and 320 x 64 sections. The L_1, L_2, and L_∞ relative error was calculated at each level, and the convergence rate was then computed from the error calculations. The L_1 error (relative sum of absolute differences) at the kth time step on level l is given by:

$$e_1^l = \frac{\sum_{j=1}^{m} \left| c\left(x_j, t^k\right) - \hat{c}_j^{l,k} \right|}{\sum_{j=1}^{m} \left| c\left(x_j, t^k\right) \right|}, \tag{78}$$

where $c(xj, tk)$ is the value of the analytical solution given by equation 4 calculated at the centroid of element E_j and $\hat{c}_j^{l,k}$ is the approximation to the solution at time step k on the level l mesh in E_j. The L_2 error (relative sum of squares) at level l is given by:

$$e_2^l = \frac{\sqrt{\sum_{j=1}^{m}\left[c\left(x_j,t^k\right)-\hat{c}_j^{l,k}\right]^2}}{\sqrt{\sum_{j=1}^{m}c\left(x_j,t^k\right)^2}} \quad . \tag{79}$$

Lastly the L_∞ error (maximum absolute difference) is given by:

$$e_\infty^l = \max_j \frac{\left|c\left(x_j,t^k\right)-\hat{c}_j^{l,k}\right|}{c\left(x_j,t^k\right)} \quad . \tag{80}$$

The convergence rate is defined as:

$$\text{convergence rate} = \frac{\log\left(e^l/e^{l+1}\right)}{\log 2} \quad . \tag{81}$$

For each simulation, the velocity u_1 = 1 m/s, while the time step was 0.02 s on the coarsest mesh and was halved on each subsequently refined mesh. The ending time for all the simulations was 0.2 s. For the first set of simulations, the dispersion coefficient $D_1 = D_2 = 4.0 \times 10^{-2}$ m^2/s. For the second set of simulations, the dispersion coefficient $D_1 = D_2 = 4.0 \times 10^{-3}$ m^2/s. The Courant-Friedrichs-Lewy (CFL) number can be defined (Liu, 1993; Mazzia and others, 2000) on each triangle E_j by:

$$\text{CFL} = \Delta t \sup\frac{\left|\partial E_j\right|}{\left|E_j\right|}\sup\left|\frac{d\mathbf{f}}{dc}\right| , \tag{82}$$

where $\left|\partial E_j\right|$ and $\left|E_j\right|$ are the perimeter and area, respectively, of E_j. The grid Peclet number Pe represents the ratio between advective flux and dispersive flux and can be defined as:

$$\text{Pe} = \frac{\text{CFL}}{|D|\,\Delta t \sup\left|1/E_j\right|} = \frac{\sup\left|\partial E_j\right|}{|D|}\sup\left|\frac{d\mathbf{f}}{dc}\right| , \tag{83}$$

where $|D|$ is the norm of the dispersion tensor. The CFL number for this problem was 1.37 on all the meshes, whereas the grid Peclet number on the coarsest mesh was 8.5 and was halved on each successive mesh. The L_1, L_2, and L_∞ errors and convergence rates for the first set of simulations (higher D) are shown in table 1. The convergence rates are somewhat better than first order over the first meshes, but decrease to first order as the mesh size decreases. These results are consistent with the first-order accuracy of the backward Euler time discretization for the dispersive step. As the mesh size decreases, the grid Peclet number decreases, and the dispersive step has a larger influence on the solution. The results to the same problem obtained by James and Jawitz (2007), who used an extension of the method discussed here, are significantly better, showing second-order convergence.

The results from the level 2 mesh (x_1-axis subdivided into 20 equal increments) are shown in figure 3. The model tracked the solute front quite well, with no oscillations.

The second set of simulations done for the one-dimensional problem used the same parameters and meshes as before, except using a smaller dispersion coefficient, $D_1 = D_2 = 4.0 \times 10^{-3}$ m^2/s. This makes the problem much more advection dominated, with a grid Peclet number of 85.4 on the coarsest mesh. Again, the grid Peclet number was halved on the next finer mesh. The convergence rates given in table 2 are relatively unchanged from the previous problem, showing that the model performs well even in strongly advection dominated situations, with no unphysical oscillations near the front (fig. 4).

Table 1. Convergence rates for the one-dimensional problem with D = 0.04 square meter per second.

Divisions in x	e_1^I	Convergence rate	e_2^I	Rate	e_∞^I	Rate
10	0.021691		0.01753		0.014429	
20	.006022	1.85	.004836	1.86	.004064	1.83
40	.001872	1.69	.001555	1.64	.001503	1.43
80	.000707	1.41	.000571	1.44	.000569	1.40
160	.000307	1.20	.000243	1.23	.000266	1.10
320	.000143	1.10	.000114	1.10	.00013	1.03

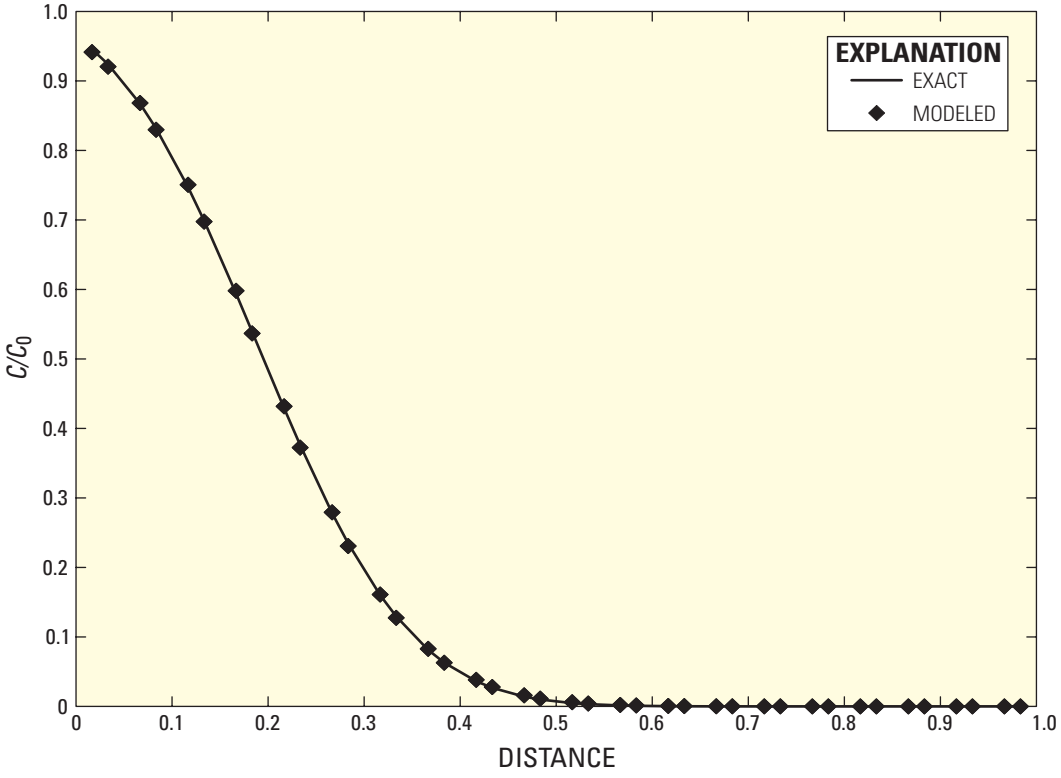


Figure 3. Model results for a solute (as the ratio of simulated concentration to initial concentration, C/C_0) on the level 2 mesh with D = 0.04 square meter per second.

Table 2. Convergence rates for the one-dimensional problem with D = 0.004 square meter per second.

Divisions in x	e_1^I	Convergence rate	e_2^I	Rate	e_∞^I	Rate
10	0.08655		0.088851		0.121609	
20	.020333	2.09	.023138	1.94	.041478	1.55
40	.006515	1.64	.00836	1.47	.017055	1.28
80	.002246	1.54	.003148	1.40	.006627	1.36
160	.000764	1.56	.000964	1.70	.002134	1.63
320	.000367	1.057	.000437	1.14	.000797	1.42

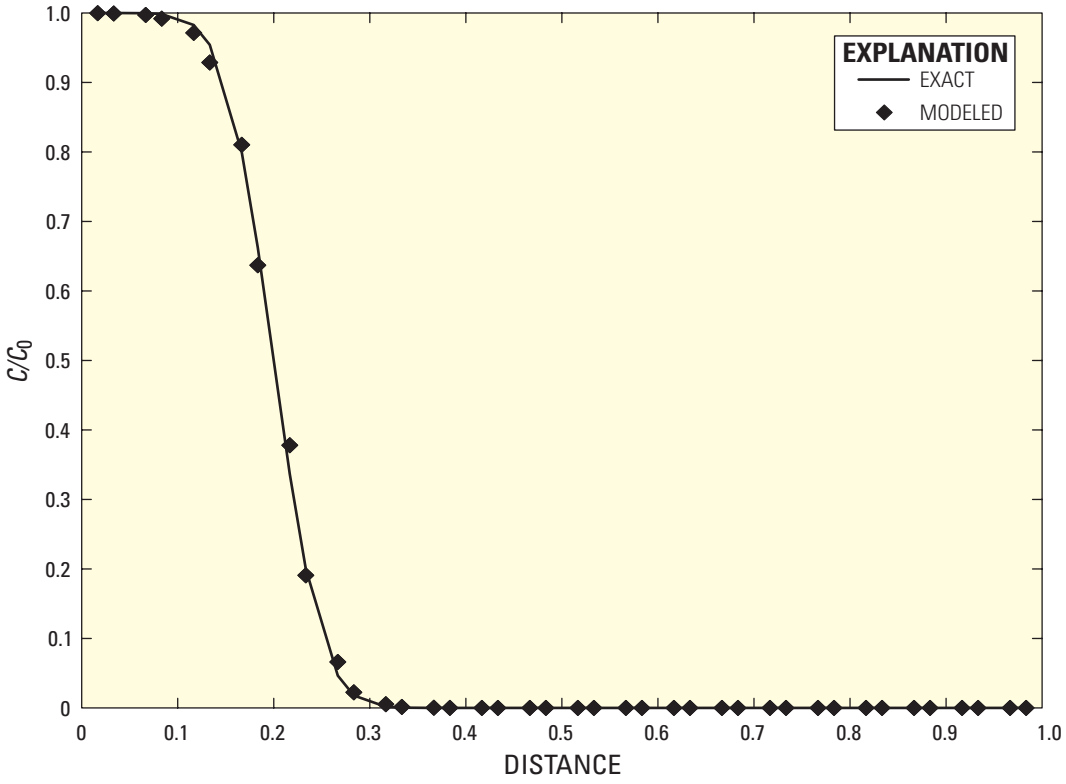


Figure 4. Model results for a solute (as the ratio of simulated concentration to initial concentration, C/C_0) on the level 2 mesh with D = 0.004 square meter per second.

Two-Dimensional Problem

A comparison with the analytical solution to the problem of a point input of mass was made to verify the model in two dimensions. The exact solution to this problem is given by:

$$c(x,y,t) = \frac{M}{4\pi d\sqrt{D_1 D_2 t}} \exp\left(-\frac{(x_1 - u_1 t)^2}{4D_1 t} - \frac{(x_1 - u_2 t)^2}{4D_2 t} \right), \quad (84)$$

where M is the mass of solute, and d is the thickness of the layer (equal to one). Because release of a point mass of solute cannot be accurately represented on a mesh, this solution at time $t_i = 0.2$ s was used to determine the initial conditions for the simulations. After a period of time t_p, the exact solution at $t_e = t_i + t_p$ was compared with the simulation at time t_p. A similar test simulation was conducted by James and Jawitz (2007), although their simulation included first-order decay so the results are not strictly comparable.

For the simulations, the meshes were 1 m x 1 m. As before, the simulation was performed on a series of progressively finer meshes, with the coarsest being 10 x 10, then 20 x 20, 40 x 40, 80 x 80, and lastly 160 x 160. The velocity $u_1 = 0.5$ m/s while $u_2 = 0$ m/s. The time step was set to 0.01 s on the coarsest mesh and was halved on each subsequent mesh. The boundary conditions were a Robin condition at $x_1 = 0$ with $c_R = 0.0$ (an inflow with zero concentration), a homogeneous Neumann condition at $x_1 = 1$, and no-flow boundaries at $x_2 = 0$ and 1. The Courant-Friedrichs-Lewy (CFL) number was 0.34 on all the meshes. The centroid of the input mass was located at $x_1 = 0.2$, $x_2 = 0.5$. The ending time for all the simulations (t_p) was 0.8 s. For the first set of simulations, the dispersion coefficient was $D_1 = D_2 = 2.0 \times 10^{-2}$ m^2/s; for the second set, the dispersion coefficient was $D_1 = 1.0 \times 10^{-2}$ and $D_2 = 5.0 \times 10^{-3}$ m^2/s. Thus, the grid Peclet numbers on the coarsest meshes were 8.53 and 54, respectively, and each were halved on each progressively finer mesh.

The convergence results are given in table 3. The errors have suboptimal convergence rates of less than one. The reason for this result is uncertain, although the initial condition is far from smooth, which contributes to the problem. Additionally, the solution given in equation 84 applies to a mass release far from any boundaries and probably contributes to the error as well, since it is difficult to simulate this effect with a finite-sized mesh.

Plots of the mass at $t_p = 0.005$s (after one time step) and at $t_p = 0.8$s (at the end of the simulation) on the 40 x 40 mesh are shown in figures 5 and 6. Note the difference in the z axis scales between the two plots is significant. At the end of the simulation, the center of mass has moved along the x_1-axis by 0.4 units (to $x_1 = 0.6$), as it should.

Results of the second simulation with dispersion coefficient $D_1 = 1.0 \times 10^{-2}$ and $D_2 = 5.0 \times 10^{-3}$ at time $t_p = 0.8$s are shown in figure 7. The solute distribution is clearly less dispersed, as expected with the lower values for the dispersion coefficients. The center of mass is again at $x_1 = 0.6$ at the end of the simulation. Errors and convergence rates for this problem are given in table 4. The convergence rates of the errors are again suboptimal, and are even poorer than in the previous problem.

The issue of suboptimal convergence was remedied by the study of James and Jawitz (2007), who significantly improved the algorithms developed here, and extended it to problems including first-order decay and mobile-immobile exchange. They were able to obtain second-order convergence for a wide range of problems.

Table 3. Errors and convergence rates for the two-dimensional problem with $D_1 = D_2 = 2.0 \times 10^{-2}$ square meter per second.

Divisions in x	e_1'	Convergence rate	e_2'	Rate	e_∞'	Rate
10	0.0536		0.05142		0.0643	
20	.0253	1.084	.02356	1.125	.0321	0.999
40	.0133	.931	.01229	.939	.0168	.932
80	.00695	.934	.00647	.925	.0088	.939
160	.0036	.958	.0033	.955	.0045	.967

Table 4. Errors and convergence rates for the two-dimensional problem with $D_1 = 1.0 \times 10^{-2}$ and $D_2 = 5.0 \times 10^{-3}$ square meter per second.

Divisions in x	e_1'	Convergence rate	e_2'	Rate	e_∞'	Rate
10	0.2151		0.2095		0.2343	
20	.0762	1.574	.0688	1.605	.0912	1.360
40	.0398	.936	.0373	.884	.0520	.810
80	.0224	.832	.0209	.836	.0287	.860
160	.0119	.907	.0112	.898	.0153	.902

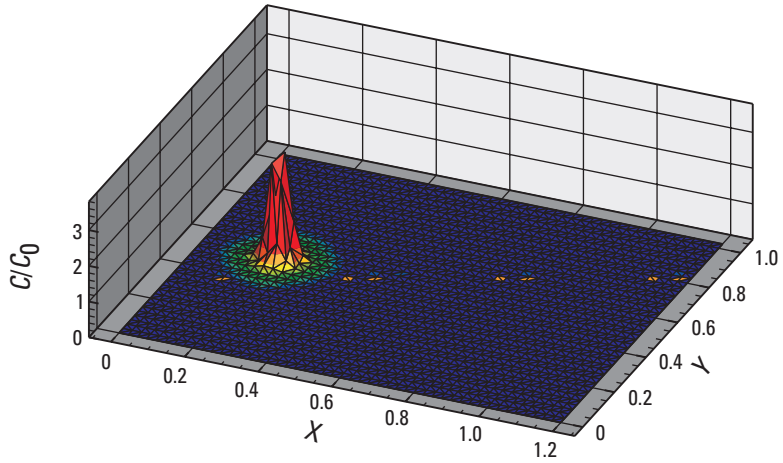


Figure 5. Solute concentrations (as the ratio of simulated concentration to initial concentration, C/C_0) at t_p = 0.005s, $D_1 = D_2 = 4.0 \times 10^{-2}$ square meter per second.

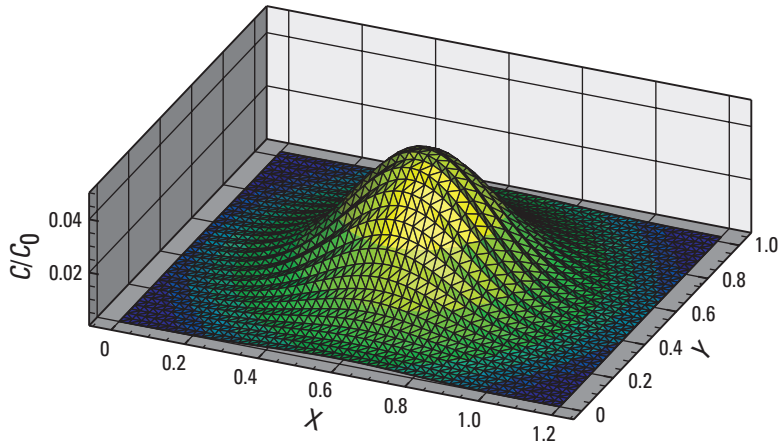


Figure 6. Solute concentrations (as the ratio of simulated concentration to initial concentration, C/C_0) at t_p = 0.8s, $D_1 = D_2 = 4.0 \times 10^{-2}$ square meter per second.

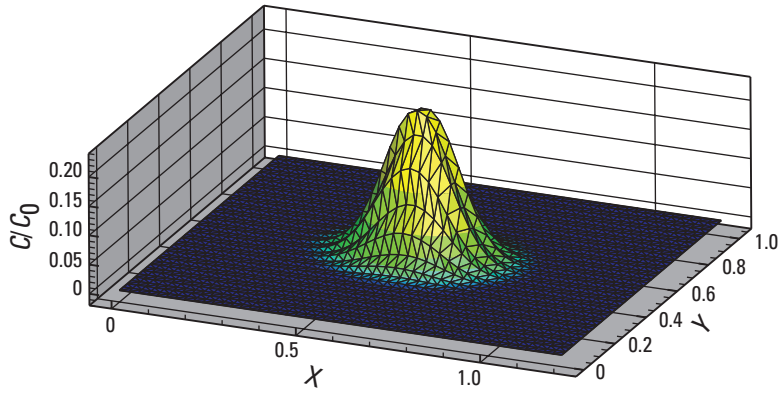


Figure 7. Solute concentrations (as the ratio of simulated concentration to initial concentration, C/C_0) at t_p = 0.8s, $D_1 = 1.0 \times 10^{-2}$, $D_2 = 5.0 \times 10^{-3}$ square meter per second.

Summary

This report presents a time-splitting Godunov-mixed finite element approach for the numerical solution of the two-dimensional advection-dispersion equation on unstructured triangular meshes. This method is applicable to transport of solutes in both surface water and ground water; however, only results with porosities of 1 are provided, corresponding to surface-water problems. The method is developed by separately discretizing the advective and dispersive parts of the advection-dispersion equation. The advective term is treated using an explicit high resolution Godunov-type finite volume method, whereas the dispersive part is approximated using an implicit hybrid-mixed finite element method. Because the dispersive part uses an implicit discretization, it can run stably with a much larger time step than the advective step, allowing algorithms to run several advective steps, then one dispersive step that encompasses the time interval of the advective steps. Because the dispersive step is computationally the most expensive, this allows schemes to be implemented that are much more computationally efficient than non-time-split algorithms.

The explicit Godunov-type method captures sharp solute fronts. When used in concert with the implicit hybrid-mixed finite element method, the resulting scheme does not suffer from limitations on the value of the grid Peclet number and shows virtually no excess numerical diffusion. The method has the limitations of assuming uniform density of the fluid in time and space, and the method works best when the fluid flows are calculated using a finite volume method such as with the regional simulation model/hydrologic simulation engine. Additionally, approximately first-order convergence is observed on the simulation of two-dimensional problems that have exact solutions in unbounded domains. Future work could include extending the method using higher order numerical schemes.

References Cited

Arbogast, Todd, and Wheeler, M.F., 1995, A characteristics-mixed finite-element method for advection-dominated transport problems: SIAM Journal on Numerical Analysis, v. 32, p. 404-424.

Bear, Jacob, 1979, Hydraulics of groundwater: New York, McGraw-Hill, 544 p.

Beckie, R.D., Wood, E.F., and Aldama, A.A., 1993, Mixed finite-element simulation of saturated groundwater-flow using a multigrid accelerated domain decomposition technique: Water Resources Research, v. 29, p. 3145-3157**.**

Bell, J.B., Dawson, C.N., and Snubin, G.R., 1988, An unsplit, higher-order Godunov method for scalar conservation-laws in multiple dimensions: Journal of Computational Physics, v. 74, p 1-24.

Benzi, Michele, Golub, G.H., and Liesen, Jörg, 2005, Numerical solution of saddle point problems: Acta Numerica, p. 1-137,

Bergamaschi, Luca, and Putti, Mario, 1999, Mixed finite elements and Newton-type linearizations for the solution of Richards' equation: International Journal for Numerical Methods in Engineering, v. 45, p. 1025-1046.

Brezzi, Franco, and Fortin, Michel, 1991, Mixed and hybrid finite element methods: New York, Springer-Verlag, 350 p.

Celia, M.A., Russell, T.F., Herrera, Ismael, and Ewing, R.E., 1990, An Eulerian-Lagrangian localized adjoint method for the advection-diffusion equation: Advances in Water Resources, v. 13, p. 187-206.

Chavent, Guy, and Roberts, J.E., 1991, A unified physical presentation of mixed, mixed-hybrid finite-elements and standard finite-difference approximations for the determination of velocities in waterflow problems: Advances in Water Resources, v. 14, p. 329-348.

Colella, Philip, and Woodward, P.R., 1984, The piecewise parabolic method (PPM) for gas-dynamical simulations: Journal of Computational Physics, v. 54, p. 174-201.

Dawson, C.N., 1990, Godunov mixed methods for immiscible displacement: International Journal for Numerical Methods in Fluids, v. 11, p. 835-847.

Dawson, C.N., 1991, Godunov-mixed methods for advective flow problems in one space dimension: SIAM Journal on Numerical Analysis, v. 28, p. 1282-1309.

Dawson, C.N., 1993, Godunov-mixed methods for advection-diffusion equations in multidimensions: SIAM Journal on Numerical Analysis, v. 30, p. 1315-1332.

Douglas, Jim Jr., and Russell, T.F., 1982, Numerical methods for convection-dominated diffusion-problems based on combining the method of characteristics with finite-element or finite-difference procedures: SIAM Journal on Numerical Analysis, v. 19, p. 871-885.

Ewing, R.E., Russell, T.F., and Wheeler, M.F., 1984, Convergence analysis of an approximation of miscible displacement in porous-media by mixed finite-elements and a modified method of characteristics: Computer Methods in Applied Mechanics and Engineering, v. 47, p. 73-92.

Gershon, N.D., and Nir, A., 1969, Effects of boundary conditions of models on tracer distribution in flow through porous mediums: Water Resources Research, v. 5, p. 830.

Godunov, S.K., 1959, A difference method for the numerical calculation of discontinuous solutions of hydrodynamic equations: Mat.Sbornik, v. 47, p. 271-306.

Healy, R.W., and Russell, T.F., 1993, A finite-volume Eulerian-Lagrangian localized adjoint method for solution of the advection-dispersion equation: Water Resources Research, v. 29, p. 2399-2413.

Healy, R.W., and Russell, T.F., 1998, Solution of the advection-dispersion equation in two dimensions by a finite-volume Eulerian-Lagrangian localized adjoint method: Advances in Water Resources, v. 21, p. 11-26.

James, A.I., and Jawitz, J.W., 2007, Modeling two-dimensional reactive transport using a Godunov-mixed finite element method: Journal of Hydrology, v. 338, p. 28-41.

Lal, A.M.W., Van Zee, R., and Belnap, M., 2005, Case study: Model to simulate regional flow in south Florida: Journal of Hydraulic Engineering, v. 131, no. 4, p. 247-258.

Liu, X.D., 1993, A maximum principle satisfying modification of triangle based adaptive stencils for the solution of Scalar hyperbolic conservation-laws: Siam Journal on Numerical Analysis, v. 30, p. 701-716.

Mazzia, Annamaria, Bergamaschi, Luca, Dawson, C.N., and Putti, Mario, 2002, Godunov mixed methods on triangular grids for advection-dispersion equations: Computational Geosciences, v. 6, p. 123-139.

Mazzia, Annamaria, Bergamaschi, Luca, and Putti, Mario, 2000, A time-splitting technique for the advection-dispersion equation in groundwater: Journal of Computational Physics, v. 157, p. 181-198.

Neuman, S.P., 1981, A Eulerian-Lagrangian numerical scheme for the dispersion-convection equation using conjugate space-time grids: Journal of Computational Physics, v. 41, p. 270-294.

Ogata, Akio, and Banks, R.B., 1961, A solution of the differential equation of longitudinal dispersion in porous media (edited): Washington, D.C., U.S. Government Printing Office.

Pinder, G.F., 1973, A Galerkin-finite element simulation of groundwater contamination on Long Island, New York: Water Resources Research, v. 9, p. 1657-1669.

Pinder, G.F., and Cooper, H.H., 1970, A numerical technique for calculating transient position of saltwater front: Water Resources Research, v. 6, p. 875.

Pinder, G.F., and Gray, W.G., 1977, Finite element simulation in surface and subsurface hydrology: New York, Academic Press, 295 p.

Raviart, P.A., and Thomas, J.M., 1977, A mixed finite element method for 2nd order elliptic problems, *in* Mathematical Aspects of the Finite Element Method Proceedings of Conference; Rome 1975, (edited): Berlin, Springer-Verlag.

Russell, T.F., 1985, Time-stepping along characteristics with incomplete iteration for a Galerkin approximation of miscible displacement in porous-media: Siam Journal on Numerical Analysis, v. 22, p. 970-1013.

Russell, T.F., and Celia, M.A., 2002, An overview of research on Eulerian-Lagrangian localized adjoint methods, ELLAM, Advances in Water Resources, v. 25, p. 1215-1231.

Russell, T.F., Heberton, C.I., Konikow, L.F., and Hornberger, G.Z., 2003, A finite-volume ELLAM for three-dimensional solute-transport modeling: Ground Water, v. 41, 258-272.

van Genuchten, M.T., and W.J. Alves., 1982, Analytical solutions of the one-dimensional convective-dispersive solute transport equation: U.S. Department of Agriculture, Agricultural Research Service, Technical Bulletin 1661.

van Genuchten, M.T., Pinder, G.F., and Frind, E.O., 1977, Simulation of two-dimensional contaminant transport with isoparametric Hermitian finite-elements: Water Resources Research, v. 13, p. 451-458.

van Leer, Bram, 1979, Towards the ultimate conservative difference scheme: A second-order sequel to Godunov's method: Journal of Computational Physics, v. 32, p. 101-136.

Appendix 1: Transport Model Structure and Application Details

Model Features

The transport model developed for this project, TaRSE, numerically approximates the solution to the advection-dispersion-reaction equation (ADRE). It treats the advective and dispersive and reaction parts of the equation separately, solving the advective part first and then solving for the dispersive step. It uses a Godunov-Mixed method (GMM), where the advective movement of solute is approximated by an explicit finite volume technique, and a hybridized mixed finite element method is used to solve for dispersion. This combination has several advantages:

- The explicit method used for the advective step is not computationally expensive because it does not require solving a large system of equations. If only a quick approximation to a solute transport problem is required, the model can bypass the computationally more expensive dispersive step. Alternatively, it should be possible to implement independent time steps for the advective and dispersive parts, where the advective algorithm can be run for several time steps, and the dispersive part run for a single aggregate time step.
- The explicit method gives stable approximations to problems with high grid Peclet numbers. Typically, when the advection-dispersion equation is solved numerically, the grid Peclet number must be less than about 10m or non-physical oscillations will develop near sharp fronts. The GMM avoids this difficulty.
- If the advective step is solved for separately, the solution to the matrix problem for the dispersive step resulting linear system can be reduced to a symmetric positive-definite form. If the two parts are solved together, the discretized linear system is non-symmetric, which is more difficult and computationally expensive to solve.

Program Features

The computer program that implements the numerical model has several features that make it readily adaptable to a wide range of problems. It solves a finite element problem on an unstructured triangular mesh, which allows it to be easily applied to problems with irregular boundary geometries and boundary conditions. It does not solve for the hydrodynamic equations; thus, a separate hydrodynamic model must be run to obtain the velocities and (or) flow between elements. Element vertex and adjacency information, along with boundary condition information, must be supplied to the model, either directly through a library call, or through an input file. In its current form, it does not run as a stand-alone program, but must be controlled by another program, such as a hydrodynamic model. A starting time, an ending time, and a time step must be given by that program.

Units

The units for the model are:

- Length: meters (m);
- Time: seconds (s);
- Mass: grams (g);
- Concentration (transported material): grams per cubic meter (g/m^3); and
- Concentration (non-transported material): grams per square meter (g/m^2).

The concentration units are applicable for all values input as boundary conditions or initial distributions, and also apply to output values. The difference in units between transported and non-transported material *must* be kept in mind when formulating equations, as no checking is done. A small degree of unit conversion can be specified in the input files for rate constants, but nothing else.

Water-Quality Data

The model allows a large number (as much as available memory and computational time allow) of solutes and stabile material (*stabile* material is not mobile, and thus, is not transported). The names and initial distributions of solutes and stabile material are given as part of the input file, which is written in XML. In addition, specifications for chemical parameters and parameters relating to physical soil properties are given in this file. Also, other data pertaining to boundary conditions, program control, and output of results is given in this file. The section titled "Structure of the Water-Quality Input File" discusses the structure of this input file.

Chemical-Reaction Equations

One of the problems with developing a water-quality model is that, as understanding of a hydrologic system evolves, the number of mobile and stabile components as well as the number and character of chemical and physical parameters used to describe the system can change dramatically. If the reactions between components all have similar characteristics (are first order), adding or removing components without requiring recompilation is relatively straightforward. However, if the system has non-linear relations between components, changing the system is more difficult. As a first step toward allowing more sophisticated descriptions of these relations, a system is implemented that describes reactions between components using a structured XML file (see section "Structure of the Equation Input File").

Interface to Hydrodynamic Models

The water-quality model needed to be run by more than one hydrodynamic model. The model was designed as a library with small, easy-to-write interfaces that call the appropriate functions to instantiate a water-quality object and call functions on that object to initialize the data structures, and provide model information (such as starting and ending times, time step, water depth, and flow, and so forth). Functions are provided that enable the water-quality model object to read data files containing solute types and reaction information (see section "Linkage with Hydrodynamic Models"). Any hydrodynamic model can use the water-quality model if it implements an existing interface, or if a new interface is written and compiled into the hydrodynamic model. An example is given in appendix 2.

Structure of the Water-Quality Input File

The input file for the model information is formatted in XML. This choice allows the use of what has been termed "self-describing data" because the names of the nodes, node attributes, and actual content of the nodes in many cases will give a sufficient description of the document itself to be able to make changes in the data. A big advantage to using XML is that the input file can be validated against a Document Type Definition (DTD) or a schema. XML is a well known standard, and additional details can be found at *http://www.w3.org/XML/* (accessed February 22, 2008) or in many books.

Schema

The input files need to follow a data scheme which specifies that the necessary parts are present and that data values and formats conform to some predefined design. The input file is validated against the schema, either during construction of the input file (the best option) or at runtime. This validation process ensures that all necessary data to run the model are contained in the file, and (in more advanced cases) can ensure that input values are of the proper type (for example, character data for variable names or floating point numbers for input values) and are within some acceptable range (greater than 0). The schema is written in RELAX NG (*http://www.relaxng.org/* accessed on February 22, 2008), which has a number of advantages over traditional DTDs or W3C schema. Translation between a RELAX NG schema is easy when using several available translators (see trang at *http://thaiopensource.com/relaxng/trang.html/* accessed on February 22, 2008).

The elements of a basic input file are discussed below, followed by a simple example, and the RELAX NG compact schema.

Primary Elements

The first element in an XML file must be the XML version information, given by:

```
<?xml version="1.0"?>
```

Such an entry is required by all XML files, and is not unique to the water-quality input. The *root node* of the input file (which is the node that contains all other nodes) is called "wq", and is required:

```
<wq version="0.1">
```

The input file must end with a closing tag, so the file will have the form:

```
<wq version="0.1">
    ...
    ...
</wq>
```

All of the other nodes will be contained by the opening and closing root nodes. These nodes are denoted by the paired opening and closing tags:

```
<control>...</control>
<components>...</components>
<parameters>...</parameters>
<mesh>...</mesh>
<output>...</output>
```

Most, but not all, of these nodes can contain (and may be required to contain) other nodes as discussed in the subsequent sections.

The <control> Tag

The control node contains some basic information that the model needs to run. Some of this information is optional or may only be used for testing purposes. Currently, this node contains only attributes and has no subnodes, although this may change with future versions. The attributes of the tag are:

- use_operator_splitting: This attribute tells the model to solve the advection and dispersion parts of the advection-dispersion equation separately. The default is "true."
- linear_solver_type: This attribute tells the PETSc solver which type of linear system solver to use when solving the dispersive part of the ADRE. The default value is "cg" (Conjugate Gradient). See the PETSc manual for further information available at *http://www-unix.mcs.anl.gov/petsc/petsc-as/*, accessed February 22, 2008.
- linear_preconditioner_type: This attribute tells the PETSc solver which type of preconditioner to use when solving the linear system for the dispersive part of the ADRE. The default version is "ilu" (Incomplete LU).
- chemistry_solver_type: This attribute tells the PETSc solver which type of linear system solver to use when solving the chemistry reactions. Because this system can by nonsymmetric, the default is "gmres" (Generalized Minimal Residual).
- chemistry_preconditioner_type: The analogous value for solving the chemistry reactions. The default is also "ilu."
- equation_filename: This is the name of the XML file that contains the description of the chemical reactions. A filename must be specified; there is no default value.
- fixed_velocity: This is an optional feature used for testing. If this is set to "true," velocities and flows obtained from the governing hydrodynamic model will be bypassed, and the values for velocity and depth specified in the next three attributes will be used. The values are constant for the duration of the simulation. The default is set to "false" and the attribute does not need to appear if this feature is not used.
- x_vel: If "fixed_velocity" is set to "true," a value (with [L]/[T] in m/s) must be given to specify the velocity in the x-direction. The default value is 0.
- y_vel: If "fixed_velocity" is set to "true," a value (also in m/s) must be given to specify the velocity in the y-direction. The default value is 0.
- depth: If "fixed_velocity" is set to "true," a value (with [L] in m) must be given to specify the depth of water. The default value is 1.

The <components> Tag

This node contains information about the components (for example, solutes) used in the model. The type of components are specified by a <mobile> tags for components that are transported, or <stabile> tags for components that are not transported, but are used in the chemistry model. Each of these subnodes, in turn, has their own subnodes to specify the name (the <name> tag) and the initial distribution of the component (the <initial_distribution> tag). The <name> tag has an optional attribute "symbol" to specify a short symbol for the component. For example, if the component is named "soluble_reactive_p," a symbol could be "srp." At this stage, the symbol value is read, but nothing is done with it in the model.

The <initial_distribution> tag has a mandatory attribute ("type") which must be set to either "constant" or "variable." If the attribute is set to "constant," the node must contain a value (which can be 0) which sets the initial value for every finite element in the mesh. Alternatively, if the attribute is set to "variable," the node must contain a string with the name of the filename with the initial distribution data. For example, a mobile component named "soluble_reactive_phosphorus" with a constant initial distribution, and a stabile component "macrophytes" that had a variable initial distribution (contained in the file "initial_macrophyte_dist.dat" would be specified:

```
<components>
    <mobile>
        <name symbol="srp">soluble_reactive_phosphorus</name>
        <initial_distribution type="constant">12.8
         </initial_distribution>
    </mobile>
    <stabile>
        <name symbol="mp">macrophytes</name>
        <initial_distribution type="variable">
            initial_macrophyte_dist.dat
        </initial_distribution>
    </stabile>
</components>
```

If a variable initial distribution is specified, the file must contain pairs of numbers on each line with the finite element number (0 offset) followed by a space or tab, and the value in that element. For a mesh with *n* elements, the input file would look like:

```
0       13.2
1       43.3
...
n-1      0.1
```

Again, the initial values need to be specified in grams per cubic meter for mobile components, and grams per square meter for stabile components.

The <parameters> Tag

This node contains information about the parameter used by the water-quality model. There are two types of parameters specified by either the <chemical> tag or the <physical> tag. Chemical parameters are those parameters that are not associated with finite elements, such as rate constants or distribution coefficients. In this version of the model, these are both temporally and spatially constant. There can be any number of chemical parameters. Physical parameters, on the other hand, are associated with specific finite elements in the mesh, and each element can have a different initial value. Unlike chemical parameters, there is a fixed set of physical parameters that are required to be specified in the input file, and no extra physical parameters can be added except by adding to the code base.

As with the components described in the previous section, each parameter node has subnodes to specify the name and the initial distribution of the parameter. Additionally, each <parameter> tag has an attribute called "units" that specifies the units associated with the parameter. For now, this has limited functionality: if the parameter is a rate constant, the value is converted to units of inverse seconds. Recognized formats for rate constants are "per_day," "per_hour," "per_minute," and "per_second." Any one of these other than "per_second" are converted to "per_second" by applying the appropriate conversion factors after the value is read by the model.

The <name> tag for parameters also has an optional attribute "symbol" to specify a short symbol for the parameter, which is also unused for now. The <initial_distribution> tag has the mandatory attribute ("type") which must be set to "constant" for <chemical> parameters, but can be either "constant" or "variable" for <physical> parameters.

As an example, this parameters node has two parameters--a chemical parameter (a settling coefficient) called “k_settling” with symbol “k_st” and a value of 2.0 day^{-1}, and the physical parameter “soil_porosity” that has a distributed value:

```
 <parameters>
    <chemical units=”per_day”>
        <name symbol=”k_st”>k_settling</name>
        <initial_distribution type=”constant”>2.0
        </initial_distribution>
    </chemical>
    <physical>
        <name>soil_porosity</name>
        <initial_distribution type=”variable”>
            initial_porosity_dist.dat
        </initial_distribution>
    </physical>
</parameters>
```

Note that for brevity, the “soil_porosity” is the only mandatory physical parameter that is shown. The physical parameters that are required to be specified in the input file are:

- “soil_porosity”: The value of the soil porosity (dimensionless).
- “surface_porosity”: The value of the porosity above the land surface. Normally equal to 1, but may be less than 1 if large quantities of vegetation are present, such as in wetlands (dimensionless).
- “active_soil_depth: The depth of the “active” soil layer; that is, depth of soil that is participating in reactions with the surface water ([L] in meters).
- “fraction_organic_soil: The fraction of total soil in the active soil layer that is composed of organic material (dimensionless).
- “fraction_inorganic_soil”: 1.0 – fraction_organic_soil. The value input here is not used, because when this is needed, the model will calculate it from the value of fraction_organic_soil. Some value, however, is required (dimensionless).
- “bulk_density”: The bulk density of the soil relative to water. **Units note:** In the water-quality input file, this is a dimensionless variable. However, this is converted into units of grams per cubic meter internally (by multiplying by $1x10^6$), so when specifying “bulk_density” in the equation file, note that it represents units of grams per cubic meter in the equation.
- “long_disp”: The longitudinal dispersivity ([L] in meters).
- “trans_disp”: The transvers dispersivity ([L] in meters).

The <mesh> Tag

This node contains information about solute boundary conditions and sources. Two possible types of subnodes are contained with the mesh node: the <bc> tag defines nodes containing the boundary condition information, and the <source> tag defines nodes containing information about sources within the mesh.

Boundary Condition Data

The boundary condition tag <bc> has three required attributes. The first attribute is “section” which indicates to the model how the boundary condition is to be applied within the model. The possible values are “ol,” “gw,” and “ol_gw” which refer to overland flow, ground water, and both overland flow and ground water, respectively. At this stage, the water-quality model is used only for surface-water flow (extension to ground water is underway), so the only value should be “ol.”

The second attribute is “type” which refers to the type of boundary condition – fixed concentration, fixed flux, or mixed, indicated by “dirichlet,” “neumann,” and “robin,” respectively (see the chapter on numerical development for more details). Generally, the best choice for inflow boundaries is “robin” since it is the most conceptually satisfying when using a GMM. Currently, Neumann boundary conditions are not implemented for flux values other than 0. Note that inflow boundaries must coincide with a hydrodynamic inflow boundary, so the controlling hydrodynamic program must have the facility to export this information in some fashion.

The third attribute is “node_id_file” which must contain a string giving the filename containing the nodes on the boundary. This input file requires two integers on each line, delineated by spaces or tabs that contain the vertex numbers of each segment on the boundary. For example, if the boundary to be specified had three segments, each connected to the next with vertex numbers 1, 12, 18, and 57, the file would look like:

```
1    12
12   18
18   57
```

When this file is read, the model finds each of the segments specified and associates a boundary value to the segment. This value is specified by a subnode, defined by the tag <data>. The names of the solutes the data are applied to are defined by the tags <for> (that is, the boundary condition is “for” the named solute).

The <data> tag has the attribute “type” which can take two values: “constant” or “variable.” If it is “constant,” the node contains the value (as a floating point number) for the solute. If it is “variable,” the node contains a string with the filename containing the data. In this case, the tag has a second attribute called “format” that has the format of the data. For now, the only format is “gwq_time_series” (for “general water-quality time series”) that has the data in a specific time series format. Other formats will be added in the future. The “gwq_time_series” data look like:

```
1994-Jun-09 0:00:00  0.069
1994-Jun-23 0:00:00  0.108
1994-Jul-07 0:00:00  0.11
1994-Aug-21 0:00:00  0.028
```

The date is specified in YYYY-MMM-DD format (four-digit years, abbreviated month name, and two-digit days), followed by a space, then the time in HH:MM:SS, followed by another space and the value of the solute concentration (in grams per cubic meter). Note that the time intervals do not need to be constant – the data are linearly interpolated between specified dates. Furthermore, if a value is needed either before the beginning or after the end of the time series, the beginning or end value, respectively, is used (no warning is printed, so the user needs to be certain that this is intended). Note that using a single date-time-value is, thus, equivalent to a constant boundary condition.

Source Data

The second way that solute can be input to the water-quality model is by using a node defined by the <source> tag. These have similar formats to the <bc> tag. If a source for solute concentrations is specified, a corresponding source must also be present in the hydrodynamic model. Therefore, the controlling hydrodynamic program must have the facility to export this information, as with the boundary condition information. The <source> tag has a “section” attribute with the same format and meaning as in the “bc” node, but it has no “type” attribute, and instead of a “node_id_file” it has a “cell_id_file” attribute that contains the integer ID numbers (offset from 0) of the finite element that have the solute sources. A source node has the same <for> and <data> tags as a <bc> tag, and reads its data in the same fashion.

An example mesh node for a mixed (Robin) boundary condition with a constant value and a solute source with variable values would look like:

```
<mesh>
  <bc type=”robin” section=”ol” node_id_file=”robin_nodes.dat”>
    <for>soluble_reactive_p</for>
    <data type=”constant”>100.0</data>
  </bc>
  <source section=”ol” cell_id_file=”cell_ids.dat”>
    <for>particulate_p</for>
    <data type=”variable”
          format=”gwq_time_series”>source_vals.dat</data>
 </source>
</mesh>
```

The <output> Tag

This section contains information to specify how model results will be output. There is a single type of subnode defined by the <monitor> tag contained by the output node. A <monitor> tag has a "type" attribute, which can be "cell," "global," or "edge." "Cell" monitors print out concentrations of mobile or stabile material (in grams per cubic meter for mobile material and grams per square meter for stabile) for a specific finite element (cell). For "cell" monitors, the "id" attribute is a single integer value that gives the integer ID of the finite element for which to print out data. "Edge" monitors print out the total mass of mobile material that passes across the edge over a time step. The "id" attribute for "edge" monitors is a string that contains two integer values corresponding to the vertex numbers at either end of the edge segment. Lastly, "global" monitors print out concentrations of mobile or stabile material in every finite element in the mesh.

All monitor types have subnodes defined by the <for> tag that list all the materials for which to print out data. Any particular monitor can have one or more <for> tags so that multiple concentrations can be output.

All monitor types also have a subnode defined by the <destination> tag that specifies the file to which the data is written. The <destination> tag has several attributes to control the output type, how often data should be output, and formats for the written data. These attributes are:

- type: For now, the type is either "text" for simple columnar ASCII data or "tecplot" for data that can be viewed using the program Tecplot. Other formats will be added in the future.
- time_format: This is a formatting string that is used to put the times in specific formats. For example, when applied to the date April 7, 2002, at 3:53:05 p.m., the formatting string "%Y-%b-%d:%H:%M" would print out "2002-Apr-07:03:53." A complete list is given at the end of the appendix. The formatting is from the C/C++ function strftime( ).
- every: This is an integer value that tells the model how often to print out data. If every="1," it will print out data at every time step; every="4" will print out data at every 4^{th} time step; and every="-1" will only print out data at the conclusion of a run. Note that for "edge" type monitors, this will only print out the mass transferred across an edge for the last time step, not accumulated values.
- delimiter: This is the character used to delimit values in "text" (not "tecplot") output files. The default value is a tab. For example delimiter=",“ would produce a comma delimited data file.
- spatial_format: This specifies the format of the location for "global" type output files. There are two values "barycenter" and "indexed." If it is set to "barycenter," the x and y coordinates of the barycenter (center point) of the finite element is written out, followed by the data values; if "indexed" is used, only the element number is written, followed by the data values.
- title: This is a string describing the monitor; for example, title="cell4_data." It is not required.

Examples

A full example input file in XML format is used for one mobile component, soluble reactive phosphorus (SRP) in the surface water column, named "soluble_reactive_sw" in the input file, and one stabile material, soil phosphorus, named "soil_p". Conceptually, the modeled exchange between the two components is a simple uptake and release, where uptake refers to the transfer from SRP to soil P, and release is the opposite, with transfer from soil P to SRP. The uptake and release coefficients (units of 1/[T], that is, s^{-1}) are named "k_up" and "k_rs," respectively. In addition to the chemical parameters, all of the required physical parameters are given in the example file. Appendix 2 contains more example model details.

Structure of the Equation Input File

The equation definition file is used to define the chemical reactions between mobile and stabile components. The definition file is in a structured XML format, not a standard "equation" style format. An example input file is presented later in appendix 2. Each equation consists of a variable on the left-hand side of the equation, and a (potentially unlimited) series of terms on the right-hand side. Each equation listed is assumed to be a differential equation for the variable on the left-hand side of the equation with respect to time. All of the terms on the right-hand side consist of a mobile or stabile system component (one of the components defined in the water-quality input file) and are multiplied or divided by an arbitrary number of factors, which are the chemical or physical parameters that are defined in the water-quality input file. Four types of terms are defined so far, and additional ones can be added when deemed necessary. The four are "zero_order," "first_order," "sorption_desorption," and "monod_growth."

An Example System of Equations

As an example, given the system of equations describing the uptake and release:

$$\frac{\partial(\phi c)}{\partial t}+\frac{\partial}{\partial x}\left(c\,u-D\frac{\partial c}{\partial x}\right)+f_2 c=f_1 c_1 \qquad \text{A.1.1}$$

$$u \gg D\,, \qquad \text{A.1.2}$$

where z_w is the water depth and z_{as} is the active soil depth. The *terms* on the right-hand side of the equations are first-order terms that add or subtract from the right-hand side, and are in turn made up of multiplicative or divisive *factors*. For example, the first term on the right-hand side of the first equation is $-k_{up}C_{sw}{}^{P}$ and the *action* of this term is subtractive ("sub."). This term is comprised of one factor, k_{up} (with action "mult") and one variable, $C_{sw}{}^{P}$ (variables are always assumed to be multiplicative). Note that if a factor refers to a physical parameter, the attribute "type" must be set equal to "physical"; otherwise, the "type" attribute can be left out. The factor "depth" (which requires 'type="physical"') in the equation input file refers to the water depth in the element. This is an intrinsic variable of the model and does not need to be defined in the water-quality input file.

The Root Node: The <eqs> Tag

The system of equations is contained in the root node denoted by the opening and closing tags <eqs>...</eqs>. The different equations are defined between these tags.

The <equation> Tag

This node contains all the elements of one equation. The left-hand side is defined, and then as many right-hand side terms as needed are defined.

The <lhs> Tag

This tag contains only one subnode enclosed by the <variable> tag that assigns the named component to the left-hand side of the equation. Again, it is assumed that this is a differential equation of the named component with respect to time.

The <term> Tag

This tag defines the type of the term that is added or subtracted from the right-hand side. The action of adding or subtracting is determined by the attribute "action," which is set to either "add" or "sub." The attribute "type" sets the type of the term, of which there are four currently available: "zero_order," "first_order," "sorption_desorption," and "monod_growth."
Zero_order—This is a simple loading or removal term that is not dependent on a variable. Examples could be atmospheric deposition into the system, which is not dependent on the value of any variable within the system; therefore, no variable is specified in the term definition. For example, a zero order term (with arbitrary multiplicative factor "factor_1") would look like:

```
<term action="add" type="zero_order">
    <factor action="mult">factor_1</factor>
</term>
```

First_order—This is the most commonly used term, where there is a first-order dependence of the term on a variable. For example, a first-order term with arbitrary variable "var_1," multiplicative factor representing a rate constant "k_1" and dividing by an arbitrary factor "factor_1" would look like:

```
<term action="add" type="first_order">
    <variable>var_1</variable>
    <factor action="mult">k_1</factor>
    <factor action="div">factor_1</factor>
</term>
```

Sorption_desorption—Sorption/desorption terms exchange material between a solid phase and an aqueous phase. The sorption/desorption term assumes instantaneous equilibrium of the solute mass between the two phases. Non-equlibrium sorption can be represented by two coupled first-order terms if desired. The <variable> is always the solid phase, if the term is on the right-hand side of the equation for the solid phase it is added, if the term is on the right-hand side of the equation for the aqueous phase, it is subtracted. It is necessary to multiply and divide by the appropriate distribution coefficient, and also by the physical parameters soil_porosity and bulk_density (note that these later require the "type" attribute to be explicitly set to "physical.")

On the right-hand side of the equation for the solid phase, the sorption/desorption term has the form:

```
<term action="add" type="sorption_desorption">
    <variable>soild_phase</variable>
    <factor action="div" type="physical">soil_porosity</factor>
    <factor action="mult" type="physical">bulk_density</factor>
    <factor action="mult">k_d</factor>
</term>
```

The corresponding form on the right-hand side of the aqueous phase would be:

```
<term action="sub" type="sorption_desorption">
    <variable>soild_phase</variable>
    <factor action="mult" type="physical">soil_porosity</factor>
    <factor action="div" type="physical">bulk_density</factor>
    <factor action="div">k_d</factor>
</term>
```

Monod_growth—A Monod growth term models a growth rate of some material based on a growth coefficient, a variable, and a limiting variable. This is basically a first-order term for the variable modified by a limiting factor. For example, the growth G^o of some type of organic material ("o") as a function of the concentration of the organic material C^o and is limited by the phosphorus concentration C^p can be described by the Monod growth term:

$$c(0,t) = C_0 \quad t \geq 0; \quad c(x,0) = 0 \quad x > 0 \tag{A.1.3}$$

where $k_g{}^o$ is the growth rate coefficient of the material (s^{-1}) and $k_{1/2}{}^o$ (also grams per cubic meter) is the half-saturation constant for the material. This example will limit the growth of material if C^p is small compared to $k_{1/2}{}^o$. But, if C^p is significantly greater $k_{1/2}{}^o$, plankton would grow at nearly the rate of the first-order term $k_g{}^o\, C^o$. This term would be written in the equation input file as:

```
<term type="monod_growth_function" action="add">
    <variable>organic_material</variable>
    <limiter>phos</limiter>
    <k_half_sat>half_sat_o</k_half_sat>
    <factor action="mult">growth_coeff</factor>
</term>
```

where "organic material" represents C^o, "phos" represents C^p, "half_sat_o" is $k_{1/2}{}^o$, and "growth_coeff" is $k_g{}^o$.

Linkage with Hydrodynamic Models and Example

Currently, the model runs with the Hydrologic Simulation Engine (HSE) of the South Florida Water Management District's (SFWMD) Regional Simulation Model (RSM). Documentation for the RSM is available from the SFWMD website at *http://www.sfwmd.gov/*. The RSM runs on Linux platforms, as does the water-quality model. The water-quality model is compiled into a static library which can be linked to by the RSM. Input data are provided to the RSM through an XML input file which defines the control parameters, hydrologic boundary conditions, sources, initial conditions, and so forth.

Three files need to be included during compilation for the RSM to utilize the water-quality program, which add two classes to the RSM code base. One of these classes is termed a "water-quality process module," which is derived from the HSE type "hydrologic process module." This object serves as a conduit for information to be passed from the HSE to the water-quality model and vice-versa. The second file is a header file that contains the Application Program Interface (API), which provides the functions that run the water-quality model to the RSM.

Example:

The input file for the HSE needs two lines added for the water-quality model to be run by RSM:

- An attribute to the <control> node that tells the RSM to actually run the model. This attribute is named "run_water_quality" and is set to either "true" or "false." If this attribute is set to "true," the water-quality model will run. If the default is "false" and the attribute is not present, the water-quality model will not be called to run.
- An attribute to the <control> node that gives the name of the water-quality XML input file. This attribute is named "waterqual_file" and should contain the name of the water-quality input file.

The example RSM/HSE XML input file is shown in section 0.

Appendix 2: Example Input Files

The RSM hydrodynamic input file is shown below, the full example water-quality input file in XML format follows after, and the corresponding equation definition file is shown in after that. This input file has one mobile component "soluble_reactive_sw," and one stabile material "soil_p". Conceptually, the modeled exchange between the two components is a simple uptake and release, where uptake refers to the transfer from soluble_reactive_sw to soil_p, and release is in the opposite direction. The uptake and release coefficients (units of 1/[T], that is, s^{-1}) are named "k_up" and "k_rs," respectively. In addition to the chemical parameters, all of the required physical parameters are also given in the example file.

In the example file, it is assumed that the hydrodynamic model is running on a rectangular area that is subdivided into a square mesh of 20x4, and each square is subdivided into two right triangles. The example mesh is shown in figure A1. The flow is from left to right, and the boundary condition for soluble_reactive_sw is applied on the left-hand side along segments 1-12, 12-23, 23-34, and 34-45. Cell monitors for both components are given at the leftmost and rightmost cells in the next to bottom row of elements; that is, the element with vertices 12-13-23 (element 20 in a zero-offset numbering scheme) and the element with vertices 22-33-32 (element 39). In addition, a global monitor with values located by the barycenters of the elements is also in the input file. The input vertex numbers for the Robin boundary condition on the left-hand side of the mesh are in the file robin_nodes.dat.

RSM Input File

To set up the hydrodynamics for the example problem, an XML file must be created for the RSM. This is shown below. Inputs for the time step length and type are required, as are the starting and ending times of the simulation. Boundary conditions are specified under the <mesh_bc> tag. The inflow boundaries in the water-quality input must coincide with these inflow boundaries, but no-flow and outflow boundaries do not need to be specified for the water-quality model because it gets this information directly from the HSE. It is required to specify a "hydrologic process module" with the <hpModules> tag, and the type of module with the line:

```
<wqpmodule>wqmodule</wqpmodule>
```

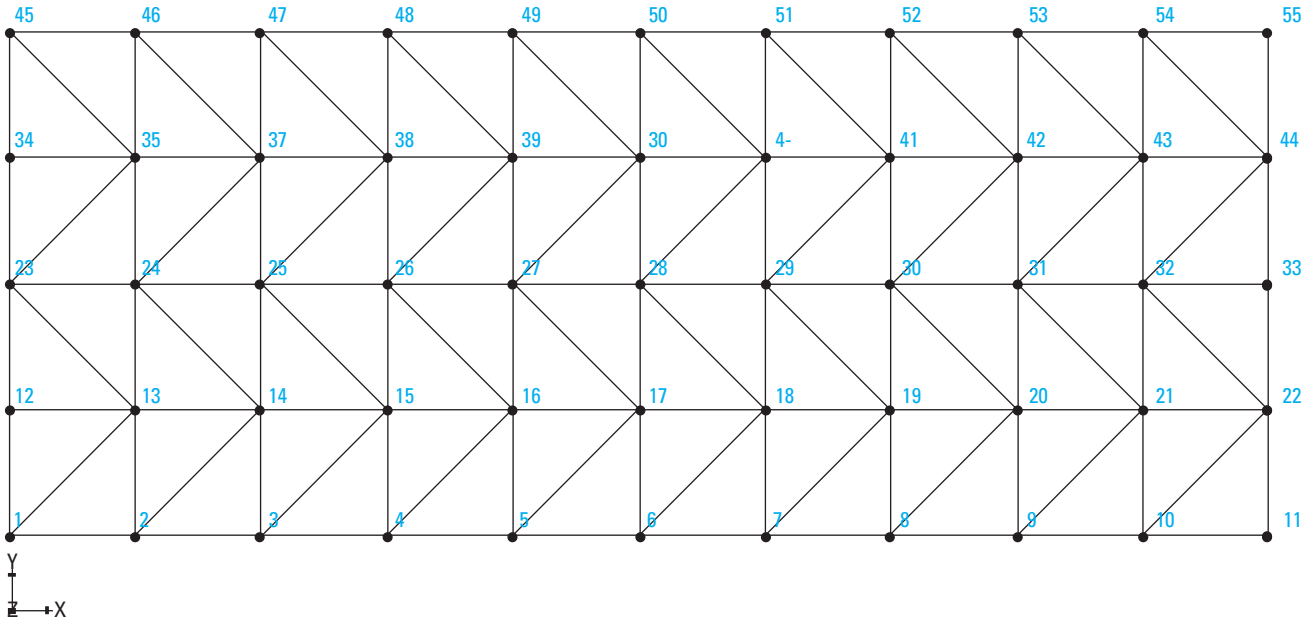


Figure A1. Example 10 x 4 mesh. Vertex numbers are shown. Flow is from left to right, and Robin boundary conditions are defined on the edges with vertex numbers 1-12, 12-23, 23-34, and 34-45.

Other initial conditions and parameters must also be specified. See the documentation provided by the SFWMD, which is available at their website at *http://www.sfwmd.gov/*.

```
<?xml version="1.0" ?>
<!DOCTYPE hse SYSTEM "../hse.dtd" []>
<hse version="0.1">
  <basins></basins>
  <control
    tslen="6"
    tstype="hour"
    startdate="01jan2000"
    starttime="0000"
    enddate="28feb2000"
    endtime="2400"
    alpha="0.500"
    solver="PETSC"
    method="gmres"
    precond="ilu"
    waterquality="true"
    waterqual_file="wq_input.xml">
  </control>
  <mesh>
    <geometry file="mesh10x4.2dm"> </geometry>
    <mesh_bc>
      <noflow section="ol_gw">
        <nodelist> 1 2 3 4 5 6 7 8 9 10 11 </nodelist>
      </noflow>
      <noflow section="ol_gw">
        <nodelist> 45 46 47 48 49 50 51 52 53 54 55 </nodelist>
      </noflow>
      <wallhead section="ol_gw">
        <nodelist> 11 22 33 44 55 </nodelist>
        <uniform><const value = "1.0"></const></uniform>
      </wallhead>
      <wallhead section="ol_gw">
        <nodelist> 1 12 23 34 45 </nodelist>
        <uniform><const value = "1.001"></const></uniform>
      </wallhead>
    </mesh_bc>
    <shead><const value="1.0"></const></shead>
    <bottom>  <const value="0.0">   </const> </bottom>
    <surface> <const value="0.0"> </const> </surface>
    <hpModules>
      <wqpmodule>wqmodule</wqpmodule>
    </hpModules>
    <conveyance>
      <mannings a="0.1233" detent="0.00001"></mannings>
    </conveyance>
    <transmissivity>
      <unconfined k = "0.02"> </unconfined>
    </transmissivity>
    <svconverter>
      <constsv sc="0.2"> </constsv>
    </svconverter>
  </mesh>
</hse>
```

Water-Quality Input File

This is the input file "wq_input.xml" for the water-quality model. The name of the input file must correspond to the "water-qual_file" attribute in the <control> section.

```
<?xml version="1.0" ?>
<wq version="0.1">
  <control
    use_operator_splitting="true"
    linear_solver_type="gmres"
    linear_preconditioner_type="ilu"
    chemistry_solver_type="gmres"
    chemistry_preconditioner_type="ilu"
    equation_filename="eqs.xml" >
  </control>
  <components>
    <mobile>
      <name symbol="srp">soluble_reactive_sw</name>
      <initial_distribution type="constant">0.005
   </initial_distribution>
    </mobile>
    <stabile>
      <name symbol="sp">soil_p</name>
      <initial_distribution type="constant">100.0
        </initial_distribution>
    </stabile>
  </components>
  <parameters>
    <chemical units="per_second">
      <name>k_up</name>
      <initial_distribution type="constant">1.0e-3
        </initial_distribution>
    </chemical>
    <chemical units="per_second">
      <name>k_rs</name>
      <initial_distribution type="constant">1.0e-7
        </initial_distribution>
    </chemical>
    <physical units="none">
      <name>soil_porosity</name>
      <initial_distribution type="constant">0.4</initial_distribution>
    </physical>
    <physical units="none">
      <name>surface_porosity</name>
      <initial_distribution type="constant">1.0</initial_distribution>
    </physical    <physical units="meter">
      <name>active_soil_depth</name>
      <initial_distribution type="constant">0.1</initial_distribution>
    </physical>
    <physical units="none">
      <name>fraction_organic_soil</name>
      <initial_distribution type="constant">0.98</initial_distribution>
    </physical>
    <physical units="none">
      <name>fraction_inorganic_soil</name>
      <initial_distribution type="constant">0.02</initial_distribution>
```

```
    </physical>
    <physical units="none">
      <name>bulk_density</name>
      <initial_distribution type="constant">0.9</initial_distribution>
    </physical>
    <physical units="meter">
      <name>long_disp</name>
      <initial_distribution type="constant">10.0</initial_distribution>
    </physical>
    <physical units="meter">
      <name>trans_disp</name>
      <initial_distribution type="constant">10.0</initial_distribution>
    </physical>
  </parameters>
  <mesh>
    <bc type="robin" section="ol" node_id_file="robin_nodes.dat">
      <for>soluble_reactive_sw</for>
      <data type="constant">0.005</data>
    </bc>
  </mesh>
  <output>
    <monitor type="cell" cell_id="20" title="Element 20 WQ monitor">
      <for>soluble_reactive_sw</for>
      <for>soil_p</for>
      <destination type="text"
                   time_format="%Y-%B-%d %H:%M"
                   every="2">./output/cell_20_output.dat</destination>
    </monitor>
    <monitor type="cell" cell_id="39" title="Element 39 WQ monitor">
      <for>soluble_reactive_sw</for>
      <for>soil_p</for>
      <destination type="text"
                   time_format="%Y-%B-%d %H:%M"
                   every="2">./output/cell_39_output.dat</destination>
    </monitor>
    <monitor type="global" title="water quality monitor">
      <for>soluble_reactive_sw</for>
      <for>soil_p</for>
      <destination type="text"
                   time_format="%Y-%B-%d %H:%M"
                   every="1"
                   spatial_format="barycenter">./output/srp_output.dat
      </destination>
    </monitor>
  </output>
</wq>
```

Equation Input File

This is the equation file corresponding to the system of equations A.1.1 and A.1.2, with the variables defined in the proceeding water-quality input file. The name of this input file must correspond to the "equation_filename" attribute in the <control> section of the water-quality input file, in this case, "eqs.xml." The full input file is:

```
<?xml version="1.0" ?>
<eqs version="0.1">
  <equation>
    <lhs>
      <variable>soluble_reactive_sw</variable>
    </lhs>
    <term action="sub" type="first_order">
      <variable>soluble_reactive_sw</variable>
      <factor action="mult">k_up</factor>
    </term>
    <term action="add" type="first_order">
      <variable>soil_p</variable>
      <factor action="mult">k_rs</factor>
      <factor action="div" type="physical">active_soil_depth</factor>
    </term>
  </equation>
  <equation>
    <lhs>
      <variable>soil_p</variable>
    </lhs>
    <term action="add" type="first_order">
      <variable>soluble_reactive_sw</variable>
      <factor action="mult">k_up</factor>
      <factor action="mult" type="physical">depth</factor>
    </term>
    <term action="sub" type="first_order">
      <variable>soil_p</variable>
      <factor action="mult">k_rs</factor>
    </term>
  </equation>
</eqs>
```

Time Formatting Characters

These are the time formatting characters used for the monitors in the output node. These are defined in the c function strftime( ):

%a	is replaced by the locale's abbreviated weekday name.
%A	is replaced by the locale's full weekday name.
%b	is replaced by the locale's abbreviated month name.
%B	is replaced by the locale's full month name.
%c	is replaced by the locale's appropriate date and time representation.
%C	is replaced by the century number (the year divided by 100 and truncated to an integer) as a decimal number [00,99].
%d	is replaced by the day of month as a decimal number [01,31].
%D	is the same as %m/%d/%y.
%e	is replaced by the day of the month as a decimal number [1,31]; a single digit is preceded by a space.
%h	is the same as %b.
%H	is replaced by the hour in 24-hour format [00,23].
%I	is replaced by the hour in 12-hour format [01,12].
%j	is replaced by the day of year as a decimal number [001,366].
%k	is replaced by the hour in 24-hour format as a decimal number [0,23]]; a single digit is preceded by a space.
%l	is replaced by the hour in 12-hour format as a decimal number [1,12]; a single digit is preceded by a space.
%m	is replaced by the month as a decimal number [01,12].
%M	is replaced by the minute as a decimal number [00,59].
%n	is replaced by a newline character.
%p	is replaced by the locale's equivalent of either A.M./P.M. indicator for 12-hour clock.
%r	is replaced by the time in A.M. and P.M. notation; in the POSIX locale, this is equivalent to %I:%M:%S %p.
%R	is replaced by the time in 24 hour notation (%H:%M).
%s	is replaced by the time in seconds since the epoch as a decimal number.
%S	is replaced by the second as a decimal number [00,59].
%t	is replaced by a tab character.
%T	is replaced by the time (%H:%M:%S).
%u	is replaced by the weekday as a decimal number [1,7], with 1 representing Monday.
%U	is replaced by the week of the year as a decimal number, with Sunday as first day of week [00,53].
%V	is replaced by the week of the year as a decimal number, with Monday as the first day of the week (01 - 53). If the week containing January 1st has four or more days in the new year, then it is considered week 1. Otherwise, it is week 53 of the previous year, and the next week is week 1.
%w	is replaced by the weekday as a decimal number [0,6]; Sunday is 0.
%W	is replaced by the week of the year as a decimal number, with Monday as first day of week [00,53]. All days in a new year preceding the first Monday are considered to be in week 0.
%x	is replaced by the locale's appropriate date representation.
%X	is replaced by the locale's appropriate time representation.
%y	is replaced by the year without century, as a decimal number [00,99].
%Y	is replaced by the year with century, as a decimal number.
%z, %Z	is replaced by the time zone name or abbreviation, or by no characters if no time zone information exists.
%%	is replaced by %.