8-3-2002

A Web-Based High Performance Simulation System for Transport and Retention of Dissolved Contaminants in Soils

Honghai Zeng

A WEB-BASED HIGH PERFORMANCE SIMULATION SYSTEM FOR TRANSPORT AND RETENTION OF DISSOLVED CONTAMINANTS IN SOILS

By

Honghai Zeng

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Engineering
in the Department of Computational Engineering

Mississippi State, Mississippi

December 2002

A WEB-BASED HIGH PERFORMANCE SIMULATION SYSTEM FOR TRANSPORT AND RETENTION OF DISSOLVED CONTAMINANTS IN SOILS

By

Honghai Zeng

Approved:

Jianping Zhu Professor of Applied Mathematics Major Professor	Ioana Banicescu Associate Professor of Computer Science Committee Member
William L. Kingery Professor of Plant and Soil Sciences Committee Member	Chunhua Sheng Associate Research Professor of Computational Engineering Committee Member
Boyd Gatlin Associate Professor of Aerospace Engineering and Engineering Mechanics Graduate Coordinator of Computational Engineering	A. Wayne Bennett Dean of the College of Engineering

Name: Honghai Zeng

Date of Degree: December 13, 2002

Institution: Mississippi State University

Major Field: Computational Engineering

Major Professor: Dr. Jianping Zhu

Title of Study: A WEB-BASED HIGH PERFORMANCE SIMULATION SYSTEM FOR TRANSPORT AND RETENTION OF DISSOLVED CONTAMINANTS IN SOILS

Pages in Study: 135

Candidate for Degree of Doctor of Philosophy

Groundwater, the major source of human drinking water, is susceptible to contamination from industrial and agricultural activities. This research develops a web-based simulation system of remote high performance computing model for contaminant transport and retention in soils. A three-dimensional advection-dispersion-reaction MRTM model, based on previous experimental and theoretical studies, is proposed to analyze the transport and retention of chemical contaminants in groundwater flowing through soils. Since three-dimensional experiments are difficult to implement and verify, this simulation system provides scientists an alternative to trace the contaminant movement in soils outside laboratories.

The alternating direction implicit (ADI) algorithm is used in this study to reduce the computational complexity. Although the ADI method is very efficient to solve the governing advection-dispersion-adsorption equations in the three-dimensional MRTM model, achieving higher order accuracy with different boundary conditions remains a difficult research topic. This research develops a new numerical scheme to achieve second-order accuracy with the Neumann-type boundary conditions. Furthermore, parallel computing is used to achieve high performance using powerful multiprocessor computers.

A web-based simulation system provides users a friendly interface for remote access to the system through Internet browsers, so as to utilize remote computing resources transparently and efficiently. In the client-side computing one-dimensional MRTM simulation system, the legacy code written in FORTRAN and C are wrapped and reused with Java code, which provides the web-based graphic user interface (GUI). The server-side computing three-dimensional MRTM simulation system integrates the remote high performance computing resources, database management systems, online visualization functionality, and web-based user-friendly GUIs. Given access to the Internet, users can execute and manage remote high performance computing jobs anywhere anytime, even through a web browser from a laptop personal computer.

In summary, this research has the following four contributions:

- Extending the one-dimensional MRTM transport and retention model to three-dimensional applications.
- Using the alternating direction implicit (ADI) numerical algorithm to reduce the computational complexity and achieving second-order accuracy with the Neumann-type boundary conditions.
- Using parallel computing to achieve high performance on multiprocessor parallel computers.
- Integrating the web-based system with user-friendly interfaces, which provide client-side and server-side computing services and web-based visualization functionality.

DEDICATION

To my parents and wife.

ACKNOWLEDGEMENTS

I am indebted to my advisor Dr. Jianping Zhu, who provides me patient assistance and kindly support all the time. This dissertation could not have been done without his continual encouragement. Thanks to all the professors who serve on my dissertation committee: Dr. Ioana Banicescu, Dr. William Kingery, and Dr. Chunhua Sheng. Sincere appreciation is due to Dr. Ioana Banicescu for her guidance and support on my research project on high performance computing topics. I would like to express my gratitude to Dr. William Kingery for his guidance, which enriches my background in soil chemistry greatly.

I definitely could not have done this work without the unselfish help and support from Dr. Vladimir Alarcon. He has been giving me invaluable advice all the time. Thanks to Dr. Wesley Brewer, who helped me a lot with this dissertation format.

I greatly appreciate the assistance I received from NSF Engineering Research Center, Mississippi State University.

Thanks to my parents and wife, who support and encourage me to pursue a PhD.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

TABLE Page

TABLE Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

NOMENCLATURE

Symbols:

ρ	soil bulk density
Θ	soil volumetric moisture content
C	solution concentration
D	dispersion coefficient
k_1	forward kinetic reaction rate
k_2	backward kinetic reaction rate
k_3	forward kinetic reaction rate
k_4	backward kinetic reaction rate
k_5	forward kinetic reaction rate
k_6	backward kinetic reaction rate
k_d	distribution coefficient
k_s	irreversible reaction rate
NEQ	Freundlich parameter
q	Darcy water flux
S	solute retained by soil
T_p	parallel computing time
T_s	sequential computing time
U	non-linear kinetic parameter
W	non-linear kinetic parameter

Abbreviations:

ADI	Alternating Direction Implicit
API	Application Programming Interface
AWT	Abstract Window Toolkit
EJB	Enterprise JavaBean
GUI	Graphical User Interface
HPC	High Performance Computing
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
J2EE	Java 2 Enterprise Edition
JDK	Java Development Kit
JFC	Java Foundation Classes
JNI	Java Native Interface
JSP	Java Server Page
JVM	Java Virtual Machine
MPI	Message Passing Interface
MRM	Multi Reaction Model
MRTM	Multi Reaction Transport Model
ODE	Ordinary Differential Equation
PBS	Portable Batch System
PDE	Partial Differential Equation
XML	Extension Markup Language

CHAPTER I

INTRODUCTION

1.1 Motivation

This research develops a web-based simulation system of remote high performance computing model for contaminant transport and retention in soils. A three-dimensional advection-dispersion-reaction model, based on previous experimental and theoretical studies, is proposed to analyze the transport and retention of chemical contaminants in groundwater flowing through soils. Efficient numerical algorithms are also developed to solve the governing equations in the physical model. Furthermore, parallel computing technologies are applied to achieve high performance using powerful computing resources. Finally, web-based integration provides simulation users friendly remote access to this system through Internet browsers, so as to utilize remote computing resources transparently and efficiently.

1.2 Transport and Retention in Soils

Groundwater, the major source of human drinking water, is susceptible to contamination from industrial and agricultural activities. Modern agriculture uses chemicals such as pesticides and herbicides heavily in order to prevent competition from either insects or other plants for the same resources needed by crops. Although these chemicals contribute to the product of sufficient yields of crops, their movement into the groundwater causes environmental problems. They may subsequently move down through the soil profile by agricultural operations such as irrigation, and natural phenomena such as rain. The dissolved chemicals can then be transported within the soil matrix and eventually reach groundwater or surface water reservoirs (Figure 1.1). Such contamination may not only produce human health hazards, but also cause global environmental disasters. For example, the intensive use of DDT, a persistent chemical with a field half-life of 2-15.6 years, incurred a global environmental disaster in the last century [1]. Therefore,

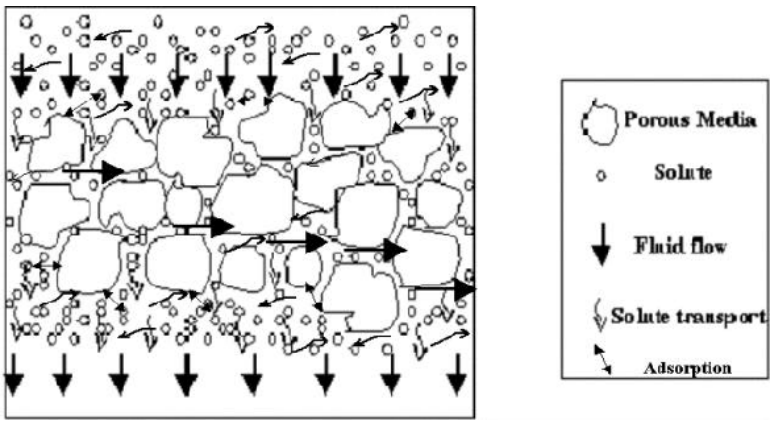

Figure 1.1: Different processes for contaminant transport and retention in soils.

understanding and estimating how chemicals are transported or retained in soils is becoming more and more important.

Different processes, including advection, dispersion, diffusion, and adsorption, work together or separately to determine the transport and retention of contaminants in the groundwater (Figure 1.1). The movement of contaminants through the soil matrix to groundwater is primarily a liquid phase process, but the partitioning of the chemicals between sorbed and dissolved phases is a critical factor in determining how rapidly the contaminants leach [2]. The adsorption of contaminants by soil constituents is an important mechanism of retention. The ability for contaminants to attach to the soil is determined by the properties of both the soil and the contaminants. Non-retained contaminants tend to leach downwards through advection, which is the movement of the contaminants with the groundwater flow. The groundwater flow rate by the hydraulic conductivity of the soil is determined by Darcy's law [2]. Lateral spread of non-retained contaminants results from processes such as dispersion and diffusion, through channels and pores in the soil matrix [3]. Diffusion and dispersion produce the similar effects and are described by Fick's law [2]. Dispersion results from the irregularities of the flow path in porous media like soils, while diffusion smoothes out the concentration gradients of contaminants in soils.

The term “adsorption” often includes all the retention and release reactions in soils, such as precipitation, dissolution, ion exchange, and adsorption-desorption mechanisms [4]. The usual conceptualization of retention mechanisms in soils often includes equilibrium models, in which it is assumed that the reaction of an individual solute species is sufficiently fast or instantaneous, and kinetic models, in which the amount of solute retained or released from the soil solution is time-dependent. Multi-reaction models, investigated in this research, deal with multiple interactions of one species in the soil environment [4], accounting for reversible as well as irreversible processes of solutes in soils [5].

1.3 Literature Review

In order to analyze the transport and retention of chemical contaminants in groundwater flowing through soils, experimental and theoretical studies generated several reliable models. Different numerical methods were applied to solve the governing equations efficiently, while computer models were developed to simulate the physical and chemical processes. This section briefly reviews the previous research in the following aspects: analytical models, numerical methods, and current computer applications. The shortcomings of the previous work are analyzed, and some of them are overcome in this dissertation research.

Among several analytical methods for the prediction of movement of dissolved substances in soils, one model was developed by Leij et al. [6] for three-dimensional non-equilibrium transport with one-dimensional steady flow in a semi-infinite soil system. In this model, the solute movement is treated as one-dimensional downward flow with three-dimensional dispersion to simplify the analytical solution. One other model, proposed by Rudakov and Rudakov [7], analyzed the risk of ground water pollution caused by leaks from surface depositories containing water-soluble toxic substances. In this analytical model, the pollutant migration was also simplified into two stages: predominantly vertical (one-dimensional) advection and three-dimensional dispersion of the pollutants in the groundwater. Usually, analytical methods have many restrictions when dealing with three-dimensional models and do not include complicated boundary conditions.

Due to the difficulties of getting general solutions in the analytical models, many numerical models were developed to simulate the solute transport and retention processes in soils. Deane et al. [8] analyzed the transport and fate of hydrophobic organic chemicals (HOCs) in consolidated sediments and saturated soils. Walter et al. [9] developed a model for simulating transport of multiple thermodynamically reacting chemical substances in groundwater systems. Islam et al. [10] presented a modeling approach to simulate the complex biogeochemical interactions in the landfill leachate contaminated soils. However, none of these three implementations deal with three-dimensional domains. In numerical models, higher dimensionality increases the computational complexity greatly. Therefore, previous research usually used simplified models to decrease the level of computational difficulty.

Manguerra and Garcia [11] introduced a modified linked-approach to solve the governing partial differential equations (PDE) for subsurface flow and salt transport by finite difference method. A variant of strongly implicit procedure (SIP), one of the most popular methods for solving matrix equations by iteration, was adopted. However, when the desired application required a full three-dimensional implementation of the model, innovative modifications must be applied to reduce the computations involved. In order to circumvent the difficulties in fully three-dimensional saturated-unsaturated flow and transport models, Yakirevich et al. [12] reduced the governing equations to quasi-three-dimensional formulations. This model coupled one-dimensional Richards equation for vertical flow in the unsaturated zone and two-dimensional equation for horizontal flow in the saturated zone. Simulations for the quasi-two-dimensional case, using finite difference numerical scheme, proved to be computationally efficient. However, this method was unstable for large time steps, and could not be implemented for the cases in which horizontal water fluxes were significant. Some other numerical schemes, such as Hopscotch algorithm [13] and Galerkin finite element technique [14], were applied for three-dimensional transport and retention problem. They all suffered the efficiency problem, which came from the computational complexity in three-dimensional applications.

The three-dimensional method-of-characteristics (MOC3D) [15, 16, 17] is a transport model that calculates transient changes in the concentration of a single solute in a three-dimensional ground water flow field. The groundwater flow equation describes the head distribution in the aquifer. The solute transport equation describes the solute concentration within the flow system. The MOC3D coupled the flow equation with the solute-transport equation, so that this model can be applied to both steady state and transient groundwater flow problems. Instead of solving the advection-dispersion governing PDEs directly, this method of characteristics solves an equivalent system of ordinary differential equations to increase computational efficiency. This approximation inevitably decreases the accuracy and precision of the numerical results. MOC3D programs, developed in FORTRAN 77, are restricted to mathematically simple retention reactions, such as first-order adsorption.

In order to avoid the restrictions to complicated adsorptive reactions in the MOC3D, Selim et al. [4, 5] developed an application based on the multi-reaction model (MRM) and multi-reaction transport model (MRTM). The MRM model includes concurrent and concurrent-consecutive retention processes of the non-linear kinetic type. It accounts for equilibrium (Freundlich) sorption and irreversible reactions. The processes considered are based on linear (first order) and non-linear kinetic reactions. The MRM model assumes that the solute in the soil environment is present in the soil solution and in several phases representing chemicals retained by the soil. This model is capable of describing chemicals under batch (kinetic) conditions without considering the water flow. The MRTM model represents an extension of the MRM model because it includes transport processes in addition to the adsorption behavior of chemicals in the soil environment [5]. The kinetic retention reaction equations and advection-dispersion equations are solved in explicit-implicit finite-difference methods in both the MRM and MRTM models. However, only one-dimensional problems were considered due to the complexity of the higher dimensional problems.

1.4 Dissertation Contributions

This review shows that due to the constraints of the analytical three-dimensional models in dealing with realistic boundary, flow and transport conditions, and the computational complexity involved in numerical schemes, previous research simplified or avoided three-dimensional models. Additionally, they do not fully utilize the current power of remote computing services provided by the Internet. Due to the lack of web-based remote computing capability, the users usually need to install the computational software on their local computers before running them. This research overcomes these difficulties with the following features:

- Extending the one-dimensional MRTM transport and retention model [5, 18] to three-dimensional applications. Chemical experiments for the one-dimensional transport and retention are widely used and proved for long time [4], while three-dimensional experiments are hard to implement and verify. Therefore, the MRTM model extension to the three-dimensional domain provides scientists an alternative to trace the contaminant movement in soils outside laboratories.

- Using the alternating direction implicit (ADI) numerical algorithm to reduce the computational complexity [19]. Although the ADI method is very efficient to solve the governing advection-dispersion-adsorption equations in the three-dimensional MRTM model, achieving higher order accuracy with different boundary conditions remains a difficult research topic. This research develops a new numerical scheme to achieve second-order accuracy with the Neumann-type boundary conditions.

- Using parallel computing to achieve high performance on multiprocessor computers. The ADI method is suitable for parallel computing, especially in the shared memory environment. Using the OpenMP, parallel implementation is realized “incrementally” without altering the data structure or program logic of the sequential code [20].

- Integrating the web-based system with user-friendly interfaces, which provide client-side and server-side computing services and web-based visualization functionality. In

the client-side computing one-dimensional MRTM simulation system, the legacy code written in FORTRAN and C are wrapped and reused with Java code, which provides the web-based graphic user interface (GUI). The server-side computing three-dimensional MRTM simulation system integrates the remote high performance computing resources, database management systems, online visualization functionality, and web-based user-friendly GUIs. Given access to the Internet, users can execute and manage remote high performance computing jobs anywhere anytime, even through a web browser from a laptop personal computer.

The rest of this dissertation is organized as follows. Chapter II discusses the theoretical background of chemical transport and retention in soils, and the numerical mathematics used to solve the MRTM governing advection-dispersion-adsorption equations. Chapter III addresses the web technologies to integrate the remote web-based simulation systems, and Chapter IV introduces the parallel computing concepts and technologies. Chapter V discusses the results and comparisons of the MRTM simulations. Finally, the conclusions are given in Chapter VI.

CHAPTER II

THEORETICAL MODEL AND MATHEMATICAL FORMULATION

This chapter first introduces the theoretical background of chemical transport and retention in soils. The multi-reaction transport model (MRTM) addresses all the processes of transport and retention, including advection, dispersion, diffusion, and adsorption. After reviewing the one-dimensional MRTM model, this chapter discusses in detail the three-dimensional MRTM extension, which is one of the contributions of this research.

Achieving high efficiency in numerical solution to the governing equations of the MRTM model is another major topic of this research. This chapter reviews the numerical methods used in the one-dimensional model, and addresses the need to improve in the three-dimensional model. The alternating direction implicit (ADI) method, which greatly reduces the computations in higher dimensional problems, is applied to the numerical solution of the three-dimensional MRTM model.

2.1 Theoretical Background

The fate of chemical transport and retention is determined by the cooperation of three different models: the fluid movement model, the solute transport model, and the solute adsorption model. The fluid movement model describes the groundwater flow in soils. The solute transport model describes the solute concentration within the groundwater flow. The solute adsorption model describes the chemical reactions between the solute and the soils. The calculation of solutions of governing partial differential equations of these models, often called advection-dispersion-adsorption equations, are very computational intensive in three-dimensional case.

2.1.1 Fluid Movement Model

The fluid movement model in soils describes the discharge rates for the solute transport model. The soil column is considered as a regular parallelepiped of dimensions X, Y, Z (X oriented downwards). The fluid enters through the upper side (YZ) and exits at the bottom. Defining $dV = dx * dy * dz$ as the finite control volume of dimensions, q_x as the discharge rate that enters side $dy * dz$, and ρ_w as the density of the fluid (aqueous solution), the amount of water entering the control volume will be:

$$mass_{in} = \rho_w q_x dy dz \tag{2.1}$$

Applying Taylor Series and truncating to the first order term, the mass flux leaving the control volume is:

$$mass_{out} = \rho_w q_x dy dz + \rho_w dy dz \frac{\partial q_x}{\partial x} dx \tag{2.2}$$

Then, the net amount of fluid mass accumulated over time is:

$$\frac{\partial M}{\partial t} = mass_{out} - mass_{in} = \rho_w dy dz \frac{\partial q_x}{\partial x} dx \tag{2.3}$$

The mass of fluid in the control volume dV can also be expressed as $M = \rho_w \eta dV$, where η is the soil porosity. Then, $M = \rho_w \eta dx dy dz$. Substituting this expression into Equation 2.3:

$$\frac{\partial(\rho_w \eta)}{\partial t} = \rho_w \frac{\partial q_x}{\partial x} \tag{2.4}$$

The term $\rho_w \eta$ is considered to be a function of the variation of the hydraulic head h with time $\partial h / \partial t$ and some structural properties of the medium. For an incompressible fluid (ρ_w is constant) and a non-deformable porous medium (η is constant), the equation is simplified to:

$$0 = \frac{\partial q_x}{\partial x} \tag{2.5}$$

Darcy's law provides: $q_x = K_x \, \partial h / \partial x$. For a homogeneous porous medium, the hydraulic conductivity K_x is constant. Substituting Darcy's law into Equation 2.5:

$$0 = K_x \frac{\partial^2 h}{\partial x^2} \tag{2.6}$$

Generalizing, for x, y, z directions:

$$K_x \frac{\partial^2 h}{\partial x^2} + K_y \frac{\partial^2 h}{\partial y^2} + K_z \frac{\partial^2 h}{\partial z^2} = 0 \tag{2.7}$$

If the medium is isotropic:

$$K \frac{\partial^2 h}{\partial x^2} + K \frac{\partial^2 h}{\partial y^2} + K \frac{\partial^2 h}{\partial z^2} = 0 \tag{2.8}$$

Therefore:

$$K \nabla^2 h = 0 \tag{2.9}$$

Solving Equation 2.7 or 2.9, the hydraulic head h can be calculated. Therefore, the velocities at any point in the soil column can also be calculated with:

$$q_i = K_i \frac{\partial h}{\partial i} \ , \tag{2.10}$$

where $i = x, y, z$. These calculated q_i values are required by the solute transport model later.

2.1.2 Solute Transport Model

The solute transport model in this section does not consider the adsorption when the solute travels through soils. Working again with a control volume dV of dimensions $dV = dx * dy * dz$, the flux (or rate of movement) of solute entering face $dydz$ in the X direction, J_x produces a solute inflow rate: $J_x dydz$. Then, the solute outflow rate leaving the control volume along this same direction is: $J_x dydz + \partial J_x / \partial x \ dxdydz$. Doing the same analysis for the Y and Z

directions, the rate of accumulation results:

$$\frac{\partial M_s}{\partial t} = -(\frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y} + \frac{\partial J_z}{\partial z})dxdydz \tag{2.11}$$

Part of the control volume $dV = dxdydz$ is occupied by the medium. If the volume of water per unit volume of bulk soil (volumetric soil water content Θ) is known, the total volume of water in the control volume will be $\Theta dV = \Theta\, dxdydz$. Then the total solute mass in the volume element (at any time t) is $\Theta\, C\, dV = \Theta\, C\, dxdydz$, where C is the solute concentration in the solution. Replacing this expression into Equation 2.11:

$$\frac{\partial \Theta C}{\partial t}dxdydz = -(\frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y} + \frac{\partial J_z}{\partial z})dxdydz \tag{2.12}$$

Simplifying:

$$\frac{\partial (\Theta\, C)}{\partial t} = -(\frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y} + \frac{\partial J_z}{\partial z}) \tag{2.13}$$

Taking into account the dispersion and diffusion (Fick's law) and the advection (solute movement with water) processes, the solute flux in the X direction is:

$$J_x = -\Theta\, D\, \frac{\partial C}{\partial x} + q_x\, C, \tag{2.14}$$

where D is the dispersion coefficient. Applying Equation 2.14 to Equation 2.13, the transport in the X direction (assuming that the transport in Y or Z is zero) is:

$$\frac{\partial \Theta C}{\partial t} = -\frac{\partial J_x}{\partial x} = -\frac{\partial}{\partial x}(-\Theta\, D\, \frac{\partial C}{\partial x} + q_x\, C) \tag{2.15}$$

Then, the so-called advection-dispersion equation results:

$$\frac{\partial \Theta C}{\partial t} = \Theta\, D\, \frac{\partial^2 C}{\partial x^2} - q_x \frac{\partial C}{\partial x} \tag{2.16}$$

In the three-dimensional case, Equation 2.16 becomes (in case Θ and D are constant):

$$\frac{\partial \Theta C}{\partial t} = \Theta\, D_x \frac{\partial^2 C}{\partial x^2} - q_x \frac{\partial C}{\partial x} + \Theta\, D_y \frac{\partial^2 C}{\partial y^2} - q_y \frac{\partial C}{\partial y} + \Theta\, D_z \frac{\partial^2 C}{\partial z^2} - q_z \frac{\partial C}{\partial z} \tag{2.17}$$

or

$$\frac{\partial \Theta C}{\partial t} = \Theta\, D\, \nabla^2 C - \nabla(q\, C) \tag{2.18}$$

To account for rates of removal or production of the solute (volatilization, root uptake, irreversible reactions, etc.), it is customary to add a Source/Sink term Q:

$$\frac{\partial \Theta C}{\partial t} = \Theta\, D\, \nabla^2 C - \nabla(q\, C) - Q \tag{2.19}$$

2.1.3 Solute Adsorption Model

The movement of chemicals in soils is also influenced by the so-called ”adsorption” mechanisms, including precipitation, dissolution, ion exchange, and adsorption-desorption. The solute adsorption model combines these retention effects with the fluid movement and solute transport models.

If an amount of solute is retained by the soil, the total amount of solute will be: $\chi = \Theta\, C + \rho\, S$, where ρ is the soil bulk density and S is the amount of solute retained by the soil. Then, Equation 2.13 changes to:

$$\frac{\partial(\Theta\, C + \rho\, S)}{\partial t} = -(\frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y} + \frac{\partial J_z}{\partial z}) \tag{2.20}$$

If Θ and ρ are constant:

$$\Theta \frac{\partial C}{\partial t} + \rho \frac{\partial S}{\partial t} = -(\frac{\partial J_x}{\partial x} + \frac{\partial J_y}{\partial y} + \frac{\partial J_z}{\partial z}) \tag{2.21}$$

Combining the advection-dispersion Equation 2.17, all the processes including advection, dispersion, diffusion, and adsorption, can be described in the following three-dimensional

advection-dispersion-adsorption equation:

$$\Theta \frac{\partial C}{\partial t} + \rho \frac{\partial S}{\partial t} = \Theta D_x \frac{\partial^2 C}{\partial x^2} - q_x \frac{\partial C}{\partial x} + \Theta D_y \frac{\partial^2 C}{\partial y^2} - q_y \frac{\partial C}{\partial y} + \Theta D_z \frac{\partial^2 C}{\partial z^2} - q_z \frac{\partial C}{\partial z} - Q \quad (2.22)$$

This advection-dispersion-adsorption equation, taking the q_x, q_y, and q_z values from the fluid movement model described above, governs the fate of chemical transport and retention in soils.

2.2 One-dimensional MRTM Model

The multi-reaction model (MRM), presented by Selim et al. [4, 5], includes both concurrent and concurrent-consecutive retention processes of the nonlinear kinetic type. The multi-reaction transport model (MRTM), an extension to MRM model, incorporates the retention processes into the convection-dispersion equation for solute transport in soils under steady water flow [4]. The computer programs for the one-dimensional MRM and MRTM models, using the explicit-implicit finite-difference numerical method, were developed for DOS-based personal computers [5].

2.2.1 Adsorption Model

The MRM model accounts for both equilibrium (Freundlich) sorption and irreversible reactions. The processes considered are based on linear (first order) and non-linear kinetic reactions. The multi-reaction model is capable of describing chemicals under batch kinetic conditions where water flow is not considered. The details on the hypothesized conceptual model and the corresponding mathematical formulation, as presented in Selim et al. [5], are summarized below.

The presence of the solute is postulated to occur in six phases (Figure 2.1), where interactions are represented by Equations 2.23 to 2.27. In those equations, k_d is a distribution coefficient, b is a Freundlich parameter, Θ is the soil volumetric water content, k_i are kinetic reaction rate coefficients ($i = 1, 2, 3, 4, 5, 6$), U and W are reaction order parameters, and k_s is a kinetic rate coefficient. The six phases are:

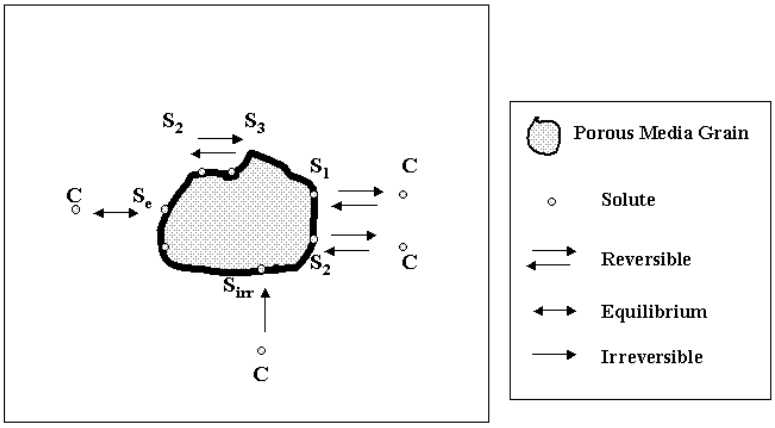

Figure 2.1: Solute phases and adsorptive reactions in soils

- C: solute dissolved in the water (concentration in the soil solution phase).
- S_e: solute sorbed reversibly in the soil and always in local equilibrium with the soil solution phase C:

$$S_e = k_d C^b \tag{2.23}$$

- S_1: solute sorbed reversibly in the soil and reacting kinetically with the soil solution phase C:

$$\rho \frac{\partial S_1}{\partial t} = \Theta k_1 C^U - \rho k_2 S_1 \tag{2.24}$$

- S_2: solute sorbed reversibly in the soil and slowly reacting kinetically with the soil solution phase C:

$$\rho \frac{\partial S_2}{\partial t} = \Theta k_3 C^W - \rho k_4 S_2 - \rho k_5 S_2 + \rho k_6 S_3 \tag{2.25}$$

- S_3: solute strongly and reversibly sorbed in the soil, slowly reacting kinetically with the S_2 phase:

$$\frac{\partial S_3}{\partial t} = k_5 S_2 - k_6 S_3 \tag{2.26}$$

- S_{irr}: solute sorbed irreversibly in the soil (irreversible sink term), reacting kinetically with the soil solution phase C

$$\rho \frac{\partial S_{irr}}{\partial t} = \Theta k_s C \tag{2.27}$$

In the numerical model, the kinetic reaction pertaining to S_2 is only considered concurrent (not consecutive), therefore the last two terms in the right hand side of Equation 2.25 ($-\rho k_5 S_2 + \rho k_6 S_3$) are not included when implementing the numerical algorithm.

2.2.2 Transport Model

The MRM model describes the retention or adsorption processes in soils where the water flow or transport process is not considered. The MRTM model, as the extension to the MRM model, incorporates the retention processes into the advection-dispersion equation for solute transport in soils under steady water flow. Derived from the solute adsorption model mentioned in Section 2.1.3, the description of chemical movement through the soil matrix is given by the advection-dispersion-adsorption equation for one-dimensional flow:

$$\rho \frac{\partial S}{\partial t} + R\Theta \frac{\partial C}{\partial t} = -q \frac{\partial C}{\partial x} + \Theta D \frac{\partial^2 C}{\partial x^2} - Q \tag{2.28}$$

Here, S is the solute concentration associated with the solid phase of the soil, ρ is the soil bulk density, D is the hydrodynamic dispersion coefficient, q is the Darcy's water flux density, Q is a sink term, x is the soil depth, and t is the time. R is a retardation term that accounts for equilibrium-reversible solute retention in the soil. It is explicitly introduced as:

$$R = 1 + \frac{b \rho K_d}{\Theta} C^{b-1} \tag{2.29}$$

The MRM model is connected to the transport model through S, such that:

$$S = S_e + S_1 + S_2 + S_3, \;\rightarrow\; \frac{\partial S}{\partial t} = \frac{\partial S_e}{\partial t} + \frac{\partial S_1}{\partial t} + \frac{\partial S_2}{\partial t} + \frac{\partial S_3}{\partial t}, \tag{2.30}$$

and

$$Q = \rho \frac{\partial S_{irr}}{\partial t} = \Theta k_s C. \tag{2.31}$$

In order to simplify discussions, steady-state water flow conditions are used and it is assumed that the soil matrix is homogeneous and isotropic (ρ and Θ are constant).

2.2.3 Numerical Method

The existence, uniqueness and other properties of the solutions to the partial differential equations can be studied analytically [21, 22]. However, the analytical solutions are often approximated by numerical methods in practice due to their complexity [23, 24]. This research uses the finite difference method [19, 25], one of the most widely used numerical methods, to solve the governing advection-dispersion-adsorption equations in the MRTM models. The finite difference method discretizes the space and time domains into grid points, and approximates the derivatives by difference formula at those points. The advection-dispersion-adsorption equation for the one-dimensional MRTM model is assumed to be well-posed with the following proper initial and boundary conditions.

For a soil profile of depth L, initial conditions are imposed by Equations 2.32 and 2.33, which assume that the soil contains a uniform initial concentration C_0 in the solution and the soil matrix is devoid of sorbed phases at time zero:

$$C = C_0, \quad t = 0, \quad 0 < x < L \tag{2.32}$$

$$S_e = S_1 = S_2 = S_3 = 0, \quad t = 0, \quad 0 < x < L \tag{2.33}$$

The Dirichlet-type boundary conditions represented by Equations 2.34 and 2.35 assume that a solute solution of known concentration C_i is applied at the soil surface for a given duration t_p. This solute pulse-type input is assumed to be followed by a solute-free solution application at the soil surface:

$$q\, C_i = -\Theta\, D \frac{\partial C}{\partial x} + q\, C, \quad x = 0, \quad t < t_p \tag{2.34}$$

$$0 = -\Theta\, D \frac{\partial C}{\partial x} + q\, C, \quad x = 0, \quad t > t_p \tag{2.35}$$

At the bottom of the soil profile, a Neumann-type boundary condition is specified as:

$$\frac{\partial C}{\partial x} = 0, \quad x = L, \quad t > 0 \tag{2.36}$$

The MRTM model is based on the MRM model, and describes the chemical transport and retention in soils. In the one-dimensional MRTM model, the kinetic retention-reaction equations and the advection-dispersion equation are expressed in finite-difference forms and solved using explicit-implicit finite-difference methods, subject to the initial and boundary conditions described above. The numerical solution provides distributions of C and S at incremental distances Δx and time steps Δt. The solute concentration in solution and solid phases are expressed as:

$$C(x,t) = C(i\Delta x,\, j\Delta t) = C_i^j\,, \tag{2.37}$$

$$S(x,t) = S(i\Delta x,\, j\Delta t) = S_i^j\,, \tag{2.38}$$

$$i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots,$$

where $N = L/\Delta x$ is the number of incremental depths in the soil.

There are two types of finite difference methods for time-dependent problems: explicit and implicit [19, 25]. The explicit scheme is usually easy to implement, but less efficient due to the strict stability requirement. The implicit scheme has better stability, but often needs to solve a computational intensive system of algebraic equations. The Crank-Nicolson algorithm [19, 26], mixing the explicit and implicit features, is used to solve the governing advection-dispersion-adsorption equation in the one-dimensional MRTM model with both accuracy and stability advantages.

The solute concentrations at time step $j + 1$ in solid phases are calculated using the explicit scheme. It is assumed that the concentration distribution at time j and $j - 1$ is known throughout

the soil. Thus, the adsorption terms are discretized by:

$$\rho\frac{\partial S_1}{\partial t} = \Theta k_1 C^U - \rho k_2 S_1 = \Theta k_1[\frac{C_i^j + C_i^{j-1}}{2}]^U - \rho k_2 (S_1)_i^j\,, \tag{2.39}$$

$$\rho\frac{\partial S_2}{\partial t} = \Theta k_3 C^W - \rho k_4 S_2 = \Theta k_3[\frac{C_i^j + C_i^{j-1}}{2}]^W - \rho k_4 (S_2)_i^j\,, \tag{2.40}$$

$$\rho\frac{\partial S_3}{\partial t} = \rho k_5 S_2 - \rho k_6 S_3 = \rho k_5 (S_2)_i^j + \rho k_6 (S_3)_i^j\,, \tag{2.41}$$

$$\rho\frac{\partial S_{irr}}{\partial t} = \Theta k_s C = \Theta k_s \frac{(C_i^j + C_i^{j-1})}{2}. \tag{2.42}$$

The Crank-Nicholson algorithm is applied to translate the one-dimensional MRTM governing equation 2.28 into the finite-difference form. The dispersion term is discretized by:

$$\Theta D\frac{\partial^2 C}{\partial x^2} = \Theta D\frac{C_{i+1}^{j+1} - 2C_i^{j+1} + C_{i-1}^{j+1}}{2(\Delta x)^2} + \Theta D\frac{C_{i+1}^{j} - 2C_i^{j} + C_{i-1}^{j}}{2(\Delta x)^2} + O(\Delta x)^2, \tag{2.43}$$

where $O(\Delta x)^2$ is the truncation error.

The advection term is expressed as:

$$q\frac{\partial C}{\partial x} = q\frac{C_{i+1}^{j+1} - C_{i-1}^{j+1}}{4\Delta x} + q\frac{C_{i+1}^{j} - C_{i-1}^{j}}{4\Delta x} + O(\Delta x)^2\,, \tag{2.44}$$

with the truncation error of order $O(\Delta x)^2$ too.

The finite-difference version of the accumulation rate term in Equation 2.28 is:

$$R\,\Theta\frac{\partial C}{\partial t} = R_i^j\,\Theta\frac{C_i^{j+1} - C_i^j}{\Delta t} + O(\Delta t), \tag{2.45}$$

where the term R_i^j is calculated with:

$$R_i^j = 1 + \frac{b\,\rho K_d}{\Theta}\left(\frac{C_i^j + C_i^{j-1}}{2}\right)^{b-1}. \tag{2.46}$$

After replacing all finite-difference equations into Equation 2.28, the terms containing the unknowns C_{i-1}^{j+1}, C_i^{j+1} and C_{i+1}^{j+1} are grouped. This grouping process is summarized as follows.

Replacement of the finite-difference Equations 2.43, 2.44 and 2.45 into Equation 2.28 leads to:

$$\begin{aligned}
&\rho\frac{\partial S}{\partial t} + R_i^j\Theta\frac{C_i^{j+1}-C_i^j}{\Delta t} \\
&\quad = \Theta D\frac{C_{i+1}^{j+1}-2C_i^{j+1}+C_{i-1}^{j+1}}{2(\Delta x)^2} + \Theta D\frac{C_{i+1}^{j}-2C_i^{j}+C_{i-1}^{j}}{2(\Delta x)^2} \\
&\quad -q\frac{C_{i+1}^{j+1}-C_{i-1}^{j+1}}{4\Delta x} - q\frac{C_{i+1}^{j}-C_{i-1}^{j}}{4\Delta x} - \Theta k_s\frac{(C_i^{j+1}+C_i^j)}{2}.
\end{aligned} \tag{2.47}$$

Grouping unknown terms to the left and known terms to the right:

$$\begin{aligned}
&-(\gamma D + \frac{\alpha\beta}{4})C_{i-1}^{j+1} + (2\gamma D + \frac{\Delta t k_s}{2} + R_i^j)C_i^{j+1} + (\frac{\alpha\beta}{4} - \gamma D)C_{i+1}^{j+1} \\
&= \gamma D(C_{i+1}^j - 2C_i^j + C_{i-1}^j) - \frac{\alpha\beta}{4}(C_{i+1}^j - C_{i-1}^j) \\
&-\frac{\Delta t\, k_s}{2}C_i^j - \frac{\Delta t\,\rho}{\Theta}\frac{\partial S}{\partial t} + R_i^j C_i^j\,,
\end{aligned} \tag{2.48}$$

where $\gamma = \frac{\Delta t}{2(\Delta x)^2}$, $\beta = \frac{\Delta t}{\Delta x}$ and $\alpha = \frac{q}{\Theta}$.

Equation 2.48 generates a tri-diagonal $N \times N$ system of equations of the form:

$$a_i^j C_{i-1}^{j+1} + b_i^j C_i^{j+1} + u_i^j C_{i+1}^{j+1} = e_i^j\,, \tag{2.49}$$

where N is the number of incremental depths in the soil ($N = L/\Delta x$), and

$a_i^j = -(\gamma\, D + \frac{\alpha\beta}{4})$,

$b_i^j = 2\gamma\, D + \frac{\Delta t\, k_s}{2} + R_i^j$,

$u_i^j = \frac{\alpha\,\beta}{4} - \gamma\, D$,

$e_i^j = \gamma\, D(C_{i+1}^j - 2Ci^j + C_{i-1}^j) - \frac{\alpha\beta}{4}(C_{i+1}^j - C_{i-1}^j) - \frac{\Delta t\, k_s}{2}C_i^j - \frac{\Delta t\,\rho}{\Theta}\frac{\partial S}{\partial t} + R_i^j C_i^j$.

Expanding Equation 2.49 for a generic time step $j+1$, the tridiagonal nature of the system is more clear:

$$\begin{aligned}
a_1^j C_0^{j+1} + b_1^j C_1^{j+1} + u_1^j C_2^{j+1} + 0\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots &= e_1^j \\
0 + a_2^j C_1^{j+1} + b_2^j C_2^{j+1} + u_2^j C_3^{j+1} + 0\cdots\cdots\cdots\cdots\cdots\cdots &= e_2^j
\end{aligned}$$

$$\begin{aligned} \cdots + 0 + a_3^j C_2^{j+1} + b_3^j C_3^{j+1} + u_3^j C_4^{j+1} + 0 \cdots &= e_3^j \\ \ddots \qquad\qquad \ddots \qquad\qquad \ddots \qquad &\vdots \\ \cdots + 0 + a_N^j C_{N-1}^{j+1} + b_N^j C_N^{j+1} + u_N^j C_{N+1}^{j+1} &= e_N^j \end{aligned} \tag{2.50}$$

The Crank-Nicolson algorithm is unconditional stable with the accuracy of order $O(\Delta t^2, \Delta x^2)$ [19, 26]. The system of equations generated by the Crank-Nicolson algorithm can be solved using the Thomas algorithm [27] with tri-diagonal Jacobian coefficient matrix. The solute concentrations in the soil solution (C_i^{j+1}) at all the nodal points are solved with the amount of solute retained by the soil (S_i^j) calculated explicitly. These C_i^{j+1} values are then fed into Equations 2.24 to 2.27, to calculate the S_i^{j+1}. The solution to those equations provides the amount of sorbed phases due to the irreversible and reversible reactions at time $j+1$. In this way, the adsorption-reaction equations are coupled with the advection-dispersion equations. The computational intensive part of the one-dimensional MRTM model is to solve the system with tri-diagonal Jocobian coefficient matrix. Since the complexity to solve the system of equations ($O(N)$) is proportional to the product of the square of the bandwidth (which is 3) with the dimension (which is N) of the system [28], the numerical scheme applied is efficient enough for the one-dimensional MRTM model.

2.3 Three-dimensional MRTM Model and ADI Method

Most previous research simplified the three-dimensional transport and retention models with different approximations [11, 12], or even avoided the three-dimensional problem [8, 9, 10]. There are many theoretical or numerical difficulties to overcome. General solutions are hard to get through analytical methods due to the complexity of the three-dimensional problems for transport and retention in soils [6, 7]. Numerical methods also have the difficulty of computational complexity [13, 14]. However, the development of efficient numerical algorithms (such as alternating direction implicit (ADI) method [29]) and the server-side parallel computing technologies make the solution to the real three-dimensional MRTM model not only possible but also efficient.

2.3.1 Three-dimensional MRTM Model

The three-dimensional MRTM model, as the extension of the one-dimensional MRTM model, considers all the advection, dispersion, diffusion, and adsorption processes in three dimensions. Based on the previous discussion of transport and retention in soils, the governing equation of the three-dimensional MRTM model can be expressed in the following form:

$$R\Theta \frac{\partial C}{\partial t} + \rho \frac{\partial S}{\partial t} = \Theta D_x \frac{\partial^2 C}{\partial x^2} - q_x \frac{\partial C}{\partial x} + \Theta D_y \frac{\partial^2 C}{\partial y^2} - q_y \frac{\partial C}{\partial y} + \Theta D_z \frac{\partial^2 C}{\partial z^2} - q_z \frac{\partial C}{\partial z} - Q \quad (2.51)$$

Here, S is the solute concentration associated with the solid phase of the soil, ρ is the soil bulk density, D_x, D_y, D_z are the hydrodynamic dispersion coefficients in different directions, q_x, q_y, q_z are the Darcy's water flux densities in different directions, Q is a sink term, x, y, z are the soil dimensions, and t is time. R is a retardation term that accounts for equilibrium-reversible solute retention in the soil. It is explicitly introduced as:

$$R = 1 + \frac{b\rho K_d}{\Theta} C^{b-1} \quad (2.52)$$

The MRM model is connected to the transport model through S, the same as in the one-dimensional MRTM model:

$$S = S_e + S_1 + S_2 + S_3, \ \rightarrow \ \frac{\partial S}{\partial t} = \frac{\partial S_e}{\partial t} + \frac{\partial S_1}{\partial t} + \frac{\partial S_2}{\partial t} + \frac{\partial S_3}{\partial t}, \quad (2.53)$$

and

$$Q = \rho \frac{\partial S_{irr}}{\partial t} = \Theta k_s C \, . \quad (2.54)$$

Steady-state water flow conditions are used and it is assumed that the soil matrix is homogeneous and isotropic. Therefore, all the coefficients, including Θ, ρ, D_x, D_y, D_z, q_x, q_y, and q_z, are irrelative to the solution of C and S. The sink term Q might also be given in some nonlinear reaction forms, so as to adapt to different kinds of retentive reactions.

In order to simplify discussions, the advection-dispersion-adsorption Equation 2.51 for the three-dimensional MRTM model is transformed to:

$$\frac{\partial C}{\partial t} + d\frac{\partial C}{\partial x} + e\frac{\partial C}{\partial y} + f\frac{\partial C}{\partial z} = a\frac{\partial^2 C}{\partial x^2} + b\frac{\partial^2 C}{\partial y^2} + c\frac{\partial^2 C}{\partial z^2} + F(x, y, z, t), \tag{2.55}$$

or

$$C_t + dC_x + eC_y + fC_z = aC_{xx} + bC_{yy} + cC_{zz} + F(x, y, z, t), \tag{2.56}$$

where $d = \frac{q_x}{R\Theta}$, $e = \frac{q_y}{R\Theta}$, $f = \frac{q_z}{R\Theta}$, $a = \frac{D_x}{R}$, $b = \frac{D_y}{R}$, $c = \frac{D_z}{R}$, and $F = -\frac{Q}{R\Theta} - \frac{\rho}{R\Theta}\frac{\partial S}{\partial t}$.

The computational domain is normalized into a cube with $(x, y, z) \in [0, 1] \times [0, 1] \times [0, 1]$. This governing equation is assumed to be well-posed with the following proper initial and boundary conditions.

The soil is supposed to contain a uniform initial concentration C_0 in the solution and the soil matrix is devoid of sorbed phases at time zero:

$$C(x, y, z, t) = C_0, \quad t = 0, \quad (x, y, z) \in [0, 1] \times [0, 1] \times [0, 1]. \tag{2.57}$$

and

$$S_e(x, y, z, t) = S_1(x, y, z, t) = S_2(x, y, z, t) = S_3(x, y, z, t) = 0,$$
$$t = 0, \quad (x, y, z) \in [0, 1] \times [0, 1] \times [0, 1]. \tag{2.58}$$

The Dirichlet-type boundary conditions assume that a solution of known concentration (C_i) is applied at the soil top surface for a given duration t_p. This solute pulse-type input is assumed to be followed by a solute-free solution application at the soil top surface. Three different types of boundary conditions are derived according to the distribution of the pulse-type input: central-point source (Equations 2.59 and 2.60), line source (Equations 2.61 and 2.62), and multi-point

source (Equations 2.63 and 2.64).

$$q_z\, C_i = -\Theta\, D_z \frac{\partial C}{\partial z} + q_z\, C, \quad z = 0, \quad x = 1/2, \quad y = 1/2, \quad t < t_p, \tag{2.59}$$

$$0 = -\Theta\, D_z \frac{\partial C}{\partial z} + q_z\, C, \quad z = 0, \quad t > t_p. \tag{2.60}$$

or

$$q_z\, C_i = -\Theta\, D_z \frac{\partial C}{\partial z} + q_z\, C, \quad z = 0, \quad x = 1/2, \quad t < t_p, \tag{2.61}$$

$$0 = -\Theta\, D_z \frac{\partial C}{\partial z} + q_z\, C, \quad z = 0, \quad t > t_p. \tag{2.62}$$

or

$$q_z\, C_i = -\Theta\, D_z \frac{\partial C}{\partial z} + q_z\, C, \quad z = 0, \quad (x, y)\, somewhere \quad t < t_p, \tag{2.63}$$

$$0 = -\Theta\, D_z \frac{\partial C}{\partial z} + q_z\, C, \quad z = 0, \quad t > t_p. \tag{2.64}$$

At the bottom of the soil profile, a Neumann-type boundary condition is specified as:

$$\frac{\partial C}{\partial z} = 0, \quad z = 1, \quad t > 0. \tag{2.65}$$

On the other sides of the soil profile, Neumann-type boundary conditions are also applied as:

$$\frac{\partial C}{\partial x} = 0, \quad x = 0, \quad t > 0, \tag{2.66}$$

$$\frac{\partial C}{\partial x} = 0, \quad x = 1, \quad t > 0, \tag{2.67}$$

$$\frac{\partial C}{\partial y} = 0, \quad y = 0, \quad t > 0, \tag{2.68}$$

$$\frac{\partial C}{\partial y} = 0, \quad y = 1, \quad t > 0. \tag{2.69}$$

2.3.2 Three-dimensional ADI Method

Due to the increased computational complexity, the three-dimensional MRTM model requires improvements in the finite difference method to solve the governing advection-dispersion-adsorption equations. The adsorptive reaction part ($\frac{\partial S}{\partial t}$) in the three-dimensional MRTM model is still treated explicitly, with the same finite-difference forms as described in the one-dimensional model. Thus $\frac{\partial S}{\partial t}$ can be merged into $F(x, y, z, t)$ together with the sink term Q, to calculate the solute concentrations in the soil solution C in the next time step. In the same way as the one-dimensional model, the adsorptive reaction equations are coupled with the advection-dispersion equations, which are solved with completely different numerical schemes.

The three-dimensional domain is discretized on a rectangular grid (x_i, y_j, z_k), where $i = 0, 1, 2, ..., NI$, $j = 0, 1, 2, ..., NJ$, and $k = 0, 1, 2, ..., NK$. The solute concentration is expressed as:

$$C(x, y, z, t) = C(i\Delta x,\ j\Delta y,\ k\Delta z,\ n\Delta t) = C^n_{i,j,k},$$
$$i = 0, 1, ..., NI, \quad j = 0, 1, ..., NJ, \quad k = 0, 1, ..., NK, \quad n = 0, 1, 2, ... \tag{2.70}$$

where $\Delta x = \frac{1}{NI}$, $\Delta y = \frac{1}{NJ}$, and $\Delta z = \frac{1}{NK}$.

2.3.2.1 Crank-Nicolson Method

The three-dimensional MRTM governing equation can be discretized using the Crank-Nicolson algorithm as:

$$\begin{aligned}\frac{C^{n+1}_{i,j,k} - C^n_{i,j,k}}{\Delta t} &+ \frac{d}{2}((C_x)^{n+1}_{i,j,k} + (C_x)^n_{i,j,k}) + \frac{e}{2}((C_y)^{n+1}_{i,j,k} + (C_y)^n_{i,j,k}) + \frac{f}{2}((C_z)^{n+1}_{i,j,k} + (C_z)^n_{i,j,k}) \\ &= \frac{a}{2}((C_{xx})^{n+1}_{i,j,k} + (C_{xx})^n_{i,j,k}) + \frac{b}{2}((C_{yy})^{n+1}_{i,j,k} + (C_{yy})^n_{i,j,k}) + \frac{c}{2}((C_{zz})^{n+1}_{i,j,k} + (C_{zz})^n_{i,j,k}) \\ &+ \frac{F^{n+1}_{i,j,k} + F^n_{i,j,k}}{2}\end{aligned} \tag{2.71}$$

or

$$\frac{C_{i,j,k}^{n+1}-C_{i,j,k}^{n}}{\Delta t}+d\delta_x(\frac{C_{i,j,k}^{n+1}+C_{i,j,k}^{n}}{2})+e\delta_y(\frac{C_{i,j,k}^{n+1}+C_{i,j,k}^{n}}{2})+f\delta_z(\frac{C_{i,j,k}^{n+1}+C_{i,j,k}^{n}}{2})$$
$$=a\delta_x^2(\frac{C_{i,j,k}^{n+1}+C_{i,j,k}^{n}}{2})+b\delta_y^2(\frac{C_{i,j,k}^{n+1}+C_{i,j,k}^{n}}{2})+c\delta_z^2(\frac{C_{i,j,k}^{n+1}+C_{i,j,k}^{n}}{2})+\frac{F_{i,j,k}^{n+1}+F_{i,j,k}^{n}}{2} \quad (2.72)$$

where

$$\begin{aligned}
\delta_x C_{i,j,k}^{n} &= \frac{C_{i+1,j,k}^{n}-C_{i-1,j,k}^{n}}{2\Delta x},\\
\delta_y C_{i,j,k}^{n} &= \frac{C_{i,j+1,k}^{n}-C_{i,j-1,k}^{n}}{2\Delta y},\\
\delta_z C_{i,j,k}^{n} &= \frac{C_{i,j,k+1}^{n}-C_{i,j,k-1}^{n}}{2\Delta z},\\
\delta_x^2 C_{i,j,k}^{n} &= \frac{C_{i+1,j,k}^{n}-2C_{i,j,k}^{n}+C_{i-1,j,k}^{n}}{\Delta x^2},\\
\delta_y^2 C_{i,j,k}^{n} &= \frac{C_{i,j+1,k}^{n}-2C_{i,j,k}^{n}+C_{i,j-1,k}^{n}}{\Delta y^2},\\
\delta_z^2 C_{i,j,k}^{n} &= \frac{C_{i,j,k+1}^{n}-2C_{i,j,k}^{n}+C_{i,j,k-1}^{n}}{\Delta z^2}.
\end{aligned}$$

Grouping the unknown terms (at time step $n+1$) to the left and known terms (at time step n) to the right:

$$\begin{aligned}
&(I+\frac{\Delta td}{2}\delta_x-\frac{\Delta ta}{2}\delta_x^2+\frac{\Delta te}{2}\delta_y-\frac{\Delta tb}{2}\delta_y^2+\frac{\Delta tf}{2}\delta_z-\frac{\Delta tc}{2}\delta_z^2)C_{i,j,k}^{n+1}\\
&\quad=(I-\frac{\Delta td}{2}\delta_x+\frac{\Delta ta}{2}\delta_x^2-\frac{\Delta te}{2}\delta_y+\frac{\Delta tb}{2}\delta_y^2-\frac{\Delta tf}{2}\delta_z+\frac{\Delta tc}{2}\delta_z^2)C_{i,j,k}^{n}\\
&\quad+\frac{F_{i,j,k}^{n+1}+F_{i,j,k}^{n}}{2}\Delta t
\end{aligned} \quad (2.73)$$

This discretization from Crank-Nicolson algorithm is unconditionally stable with the accuracy of order $O(\Delta t^2, \Delta x^2)$ [19, 26]. Instead of the tri-diagonal Jacobian matrix in the one-dimensional MRTM model, the system of algebraic equations generated by this discretization, i.e.

$$\mathbf{A}\mathbf{c}^{n+1}=\mathbf{H}\mathbf{c}^{n}+\frac{\Delta t}{2}(\mathbf{f}^{n+1}+\mathbf{f}^{n}) \quad (2.74)$$

has the following coefficient matrix in a nested block form:

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} & \mathbf{C} & & & \\ \mathbf{C} & \mathbf{B} & \mathbf{C} & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \mathbf{C} \\ & & & \mathbf{C} & \mathbf{B} \end{bmatrix},$$

where

$$\mathbf{B} = \begin{bmatrix} \mathbf{D} & \mathbf{E} & & & \\ \mathbf{E} & \mathbf{D} & \mathbf{E} & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \mathbf{E} \\ & & & \mathbf{E} & \mathbf{D} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \times & & & & \\ & \times & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \times \end{bmatrix},$$

and

$$\mathbf{D} = \begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \times \\ & & & \times & \times \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \times & & & & \\ & \times & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \times \end{bmatrix}.$$

The three-dimensional MRTM model updates the solute concentrations $C^n_{i,j,k}$ in each time step by solving the system of equations with the Jacobian coefficient matrix $\mathbf{A}$ above. Assuming that $NI = NJ = NK = N$ to simplify the complexity analysis, the dimensions of matrix $\mathbf{D}$ and $\mathbf{E}$ are $N \times N$. The dimensions of matrix $\mathbf{B}$ and $\mathbf{C}$ are $N^2 \times N^2$. Therefore, the Jocobian coefficient matrix $\mathbf{A}$ takes a multi-diagonal form with dimensions $N^3 \times N^3$. The half bandwidth of $\mathbf{A}$ is N^2. Since the complexity to solve the system of equations using direct methods is

proportional to the product of the square of the bandwidth with the dimension of the system [28], this numerical scheme has the total complexity of $(N^2)^2 \times N^3$ or $O(N^7)$. Therefore, the traditional Crank-Nicolson numerical scheme with direct solution algorithms is too expensive and unacceptable for the three-dimensional MRTM model.

2.3.2.2 Fractional Time Step Method

In order to reduce the computational complexity, this research compares two different schemes to improve the three-dimensional numerical method. One straightforward way to reduce the computational complexity is to use the fractional time step algorithm, in which only the spatial operators in one direction are treated implicitly at each fractional time step:

$$\begin{aligned}\frac{C_{i,j,k}^{n+\frac{1}{3}} - C_{i,j,k}^{n}}{\frac{1}{3}\Delta t} &= a\frac{C_{i-1,j,k}^{n+\frac{1}{3}} - 2C_{i,j,k}^{n+\frac{1}{3}} + C_{i+1,j,k}^{n+\frac{1}{3}}}{\Delta x^2} - d\frac{C_{i+1,j,k}^{n+\frac{1}{3}} - C_{i-1,j,k}^{n+\frac{1}{3}}}{2\Delta x} \\ &+ b\frac{C_{i,j-1,k}^{n} - 2C_{i,j,k}^{n} + C_{i,j+1,k}^{n}}{\Delta y^2} - e\frac{C_{i,j+1,k}^{n} - C_{i,j-1,k}^{n}}{2\Delta y} \\ &+ c\frac{C_{i,j,k-1}^{n} - 2C_{i,j,k}^{n} + C_{i,j,k+1}^{n}}{\Delta z^2} - f\frac{C_{i,j,k+1}^{n} - C_{i,j,k-1}^{n}}{2\Delta z} \\ &+ \frac{1}{2}(Q_{i,j,k}^{n+\frac{1}{3}} + Q_{i,j,k}^{n}) \end{aligned} \tag{2.75}$$

$$\begin{aligned}\frac{C_{i,j,k}^{n+\frac{2}{3}} - C_{i,j,k}^{n+\frac{1}{3}}}{\frac{1}{3}\Delta t} &= a\frac{C_{i-1,j,k}^{n+\frac{1}{3}} - 2C_{i,j,k}^{n+\frac{1}{3}} + C_{i+1,j,k}^{n+\frac{1}{3}}}{\Delta x^2} - d\frac{C_{i+1,j,k}^{n+\frac{1}{3}} - C_{i-1,j,k}^{n+\frac{1}{3}}}{2\Delta x} \\ &+ b\frac{C_{i,j-1,k}^{n+\frac{2}{3}} - 2C_{i,j,k}^{n+\frac{2}{3}} + C_{i,j+1,k}^{n+\frac{2}{3}}}{\Delta y^2} - e\frac{C_{i,j+1,k}^{n+\frac{2}{3}} - C_{i,j-1,k}^{n+\frac{2}{3}}}{2\Delta y} \\ &+ c\frac{C_{i,j,k-1}^{n+\frac{1}{3}} - 2C_{i,j,k}^{n+\frac{1}{3}} + C_{i,j,k+1}^{n+\frac{1}{3}}}{\Delta z^2} - f\frac{C_{i,j,k+1}^{n+\frac{1}{3}} - C_{i,j,k-1}^{n+\frac{1}{3}}}{2\Delta z} \\ &+ \frac{1}{2}(Q_{i,j,k}^{n+\frac{2}{3}} + Q_{i,j,k}^{n+\frac{1}{3}}) \end{aligned} \tag{2.76}$$

$$\begin{aligned}\frac{C_{i,j,k}^{n+1} - C_{i,j,k}^{n+\frac{2}{3}}}{\frac{1}{3}\Delta t} &= a\frac{C_{i-1,j,k}^{n+\frac{2}{3}} - 2C_{i,j,k}^{n+\frac{2}{3}} + C_{i+1,j,k}^{n+\frac{2}{3}}}{\Delta x^2} - d\frac{C_{i+1,j,k}^{n+\frac{2}{3}} - C_{i-1,j,k}^{n+\frac{2}{3}}}{2\Delta x}\end{aligned}$$

$$+b\frac{C_{i,j-1,k}^{n+\frac{2}{3}}-2C_{i,j,k}^{n+\frac{2}{3}}+C_{i,j+1,k}^{n+\frac{2}{3}}}{\Delta y^2}-e\frac{C_{i,j+1,k}^{n+\frac{2}{3}}-C_{i,j-1,k}^{n+\frac{2}{3}}}{2\Delta y}$$
$$+c\frac{C_{i,j,k-1}^{n+1}-2C_{i,j,k}^{n+1}+C_{i,j,k+1}^{n+1}}{\Delta z^2}-f\frac{C_{i,j,k+1}^{n+1}-C_{i,j,k-1}^{n+1}}{2\Delta z}$$
$$+\frac{1}{2}(Q_{i,j,k}^{n+1}+Q_{i,j,k}^{n+\frac{2}{3}}) \tag{2.77}$$

For the three-dimensional advection-dispersion-adsorption governing equation, this straightforward approximation alternatively applies implicit finite difference scheme to the first and second order derivatives in different direction during each fractional time step ($\frac{\Delta t}{3}$). It reduces a three-dimensional problem into a sequence of one-dimensional problems that only require solving systems of equations with tri-diagonal coefficient matrices, thereby greatly decreasing the computational complexity. Although the boundary conditions at all fractional time steps are clearly defined at $n+\frac{1}{3}$, $n+\frac{2}{3}$, and $n+1$, this method is only conditionally stable and first-order accurate in time [19, 30]. Therefore, the straightforward fractional time step method can not provide efficient and accurate numerical solution for the three-dimensional MRMT model.

2.3.2.3 Approximate Factorization ADI Method

Another scheme to improve the numerical solution to the three-dimensional advection-dispersion-adsorption governing equation is the approximate factorization method. Derived from the Crank-Nicolson algorithm, Equation 2.73 can be approximately factorized as:

$$(I+\frac{\Delta td}{2}\delta_x-\frac{\Delta ta}{2}\delta_x^2)(I+\frac{\Delta te}{2}\delta_y-\frac{\Delta tb}{2}\delta_y^2)(I+\frac{\Delta tf}{2}\delta_z-\frac{\Delta tc}{2}\delta_z^2)C_{i,j,k}^{n+1}$$
$$=(I-\frac{\Delta td}{2}\delta_x+\frac{\Delta ta}{2}\delta_x^2)(I-\frac{\Delta te}{2}\delta_y+\frac{\Delta tb}{2}\delta_y^2)(I-\frac{\Delta tf}{2}\delta_z+\frac{\Delta tc}{2}\delta_z^2)C_{i,j,k}^{n}$$
$$+\frac{F_{i,j,k}^{n+1}+F_{i,j,k}^{n}}{2}\Delta t \tag{2.78}$$

The difference between Equations 2.78 and 2.73, or the additional error of the factorization, is:

$$
\begin{aligned}
&(\tfrac{\Delta t^2 de}{4}\delta_x\delta_y - \tfrac{\Delta t^2 db}{4}\delta_x\delta_y^2 - \tfrac{\Delta t^2 ae}{4}\delta_x^2\delta_y + \tfrac{\Delta t^2 ab}{4}\delta_x^2\delta_y^2 + \tfrac{\Delta t^2 ef}{4}\delta_y\delta_z - \tfrac{\Delta t^2 bf}{4}\delta_y^2\delta_z \\
&-\tfrac{\Delta t^2 ec}{4}\delta_y\delta_z^2 + \tfrac{\Delta t^2 bc}{4}\delta_y^2\delta_z^2 + \tfrac{\Delta t^2 df}{4}\delta_x\delta_z - \tfrac{\Delta t^2 af}{4}\delta_x^2\delta_z - \tfrac{\Delta t^2 dc}{4}\delta_x\delta_z^2 + \tfrac{\Delta t^2 ac}{4}\delta_x^2\delta_z^2) \\
&(C_{i,j,k}^{n+1} - C_{i,j,k}^{n}) \\
&+(\tfrac{\Delta t^3 def}{8}\delta_x\delta_y\delta_z - \tfrac{\Delta t^3 dbf}{8}\delta_x\delta_y^2\delta_z - \tfrac{\Delta t^3 aef}{8}\delta_x^2\delta_y\delta_z + \tfrac{\Delta t^3 abf}{8}\delta_x^2\delta_y^2\delta_z \\
&-\tfrac{\Delta t^3 dec}{8}\delta_x\delta_y\delta_z^2 + \tfrac{\Delta t^3 dbc}{8}\delta_x\delta_y^2\delta_z^2 + \tfrac{\Delta t^3 aec}{8}\delta_x^2\delta_y\delta_z^2 - \tfrac{\Delta t^3 abc}{8}\delta_x^2\delta_y^2\delta_z^2) \\
&(C_{i,j,k}^{n+1} + C_{i,j,k}^{n}) \\
&= \\
&(\tfrac{\Delta t^2 de}{4}\delta_x\delta_y - \tfrac{\Delta t^2 db}{4}\delta_x\delta_y^2 - \tfrac{\Delta t^2 ae}{4}\delta_x^2\delta_y + \tfrac{\Delta t^2 ab}{4}\delta_x^2\delta_y^2 + \tfrac{\Delta t^2 ef}{4}\delta_y\delta_z - \tfrac{\Delta t^2 bf}{4}\delta_y^2\delta_z \\
&-\tfrac{\Delta t^2 ec}{4}\delta_y\delta_z^2 + \tfrac{\Delta t^2 bc}{4}\delta_y^2\delta_z^2 + \tfrac{\Delta t^2 df}{4}\delta_x\delta_z - \tfrac{\Delta t^2 af}{4}\delta_x^2\delta_z - \tfrac{\Delta t^2 dc}{4}\delta_x\delta_z^2 + \tfrac{\Delta t^2 ac}{4}\delta_x^2\delta_z^2) \\
&((C_t)_{i,j,k}^{n+\frac{1}{2}} + O(\Delta t^2))\Delta t \\
&+2(\tfrac{\Delta t^3 def}{8}\delta_x\delta_y\delta_z - \tfrac{\Delta t^3 dbf}{8}\delta_x\delta_y^2\delta_z - \tfrac{\Delta t^3 aef}{8}\delta_x^2\delta_y\delta_z + \tfrac{\Delta t^3 abf}{8}\delta_x^2\delta_y^2\delta_z \\
&-\tfrac{\Delta t^3 dec}{8}\delta_x\delta_y\delta_z^2 + \tfrac{\Delta t^3 dbc}{8}\delta_x\delta_y^2\delta_z^2 + \tfrac{\Delta t^3 aec}{8}\delta_x^2\delta_y\delta_z^2 - \tfrac{\Delta t^3 abc}{8}\delta_x^2\delta_y^2\delta_z^2) \\
&(C_{i,j,k}^{n+\frac{1}{2}} + O(\Delta t^2)) \\
&= \\
&(\tfrac{de}{4}\delta_x\delta_y - \tfrac{db}{4}\delta_x\delta_y^2 - \tfrac{ae}{4}\delta_x^2\delta_y + \tfrac{ab}{4}\delta_x^2\delta_y^2 + \tfrac{ef}{4}\delta_y\delta_z - \tfrac{bf}{4}\delta_y^2\delta_z \\
&-\tfrac{ec}{4}\delta_y\delta_z^2 + \tfrac{bc}{4}\delta_y^2\delta_z^2 + \tfrac{df}{4}\delta_x\delta_z - \tfrac{af}{4}\delta_x^2\delta_z - \tfrac{dc}{4}\delta_x\delta_z^2 + \tfrac{ac}{4}\delta_x^2\delta_z^2) \\
&((C_t)_{i,j,k}^{n+\frac{1}{2}} + O(\Delta t^2))\Delta t^3 \\
&+2(\tfrac{def}{8}\delta_x\delta_y\delta_z - \tfrac{dbf}{8}\delta_x\delta_y^2\delta_z - \tfrac{aef}{8}\delta_x^2\delta_y\delta_z + \tfrac{abf}{8}\delta_x^2\delta_y^2\delta_z \\
&-\tfrac{dec}{8}\delta_x\delta_y\delta_z^2 + \tfrac{dbc}{8}\delta_x\delta_y^2\delta_z^2 + \tfrac{aec}{8}\delta_x^2\delta_y\delta_z^2 - \tfrac{abc}{8}\delta_x^2\delta_y^2\delta_z^2) \\
&(C_{i,j,k}^{n+\frac{1}{2}} + O(\Delta t^2))\Delta t^3
\end{aligned}
$$

$$
\begin{aligned}
&= \\
&\tfrac{de\Delta t^3}{4}(C_{txy}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2) + O(\Delta t^2)) - \tfrac{db\Delta t^3}{4}(C_{txyy}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2) + O(\Delta t^2)) \\
&- \tfrac{ae\Delta t^3}{4}(C_{txxy}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2) + O(\Delta t^2)) + \tfrac{ab\Delta t^3}{4}(C_{txxyy}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2) + O(\Delta t^2)) \\
&+ \tfrac{ef\Delta t^3}{4}(C_{tyz}^{n+\frac{1}{2}} + O(\Delta y^2\Delta z^2) + O(\Delta t^2)) - \tfrac{bf\Delta t^3}{4}(C_{tyyz}^{n+\frac{1}{2}} + O(\Delta y^2\Delta z^2) + O(\Delta t^2)) \\
&- \tfrac{ec\Delta t^3}{4}(C_{tyzz}^{n+\frac{1}{2}} + O(\Delta y^2\Delta z^2) + O(\Delta t^2)) + \tfrac{bc\Delta t^3}{4}(C_{tyyzz}^{n+\frac{1}{2}} + O(\Delta y^2\Delta z^2) + O(\Delta t^2)) \\
&+ \tfrac{df\Delta t^3}{4}(C_{txz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta z^2) + O(\Delta t^2)) - \tfrac{af\Delta t^3}{4}(C_{txxz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta z^2) + O(\Delta t^2)) \\
&- \tfrac{dc\Delta t^3}{4}(C_{txzz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta z^2) + O(\Delta t^2)) + \tfrac{ac\Delta t^3}{4}(C_{txxzz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta z^2) + O(\Delta t^2)) \\
&+ \tfrac{def\Delta t^3}{4}(C_{xyz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2\Delta z^2) + O(\Delta t^2)) \\
&- \tfrac{dbf\Delta t^3}{4}(C_{xyyz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2\Delta z^2) + O(\Delta t^2)) \\
&- \tfrac{aef\Delta t^3}{4}(C_{xxyz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2\Delta z^2) + O(\Delta t^2)) \\
&+ \tfrac{abf\Delta t^3}{4}(C_{xxyyz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2\Delta z^2) + O(\Delta t^2)) \\
&- \tfrac{dec\Delta t^3}{4}(C_{xyzz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2\Delta z^2) + O(\Delta t^2)) \\
&+ \tfrac{dbc\Delta t^3}{4}(C_{xyyzz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2\Delta z^2) + O(\Delta t^2)) \\
&+ \tfrac{aec\Delta t^3}{4}(C_{xxyzz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2\Delta z^2) + O(\Delta t^2)) \\
&- \tfrac{abc\Delta t^3}{4}(C_{xxyyzz}^{n+\frac{1}{2}} + O(\Delta x^2\Delta y^2\Delta z^2) + O(\Delta t^2)) \\
&= O(\Delta t^3)\,,
\end{aligned}
$$

provided C_{txy}, C_{txyy}, C_{txxy}, C_{txxyy}, C_{tyz}, C_{tyyz}, C_{tyzz}, C_{tyyzz}, C_{txz}, C_{txxz}, C_{txzz}, C_{txxzz}, C_{xyz}, C_{xyyz}, C_{xxyz}, C_{xxyyz}, C_{xyzz}, C_{xyyzz}, C_{xxyzz}, and C_{xxyyzz} are bounded. This additional error is of the same order as the truncation error in the original Crank-Nicolson algorithm (Equation 2.73). Therefore, the following three-step approximate factorization ADI method still maintains second-order accuracy in both time and space.

$$
\begin{aligned}
&(I + \frac{\Delta td}{2}\delta_x - \frac{\Delta ta}{2}\delta_x^2)C_{i,j,k}^{*} \\
&\quad = (I - \frac{\Delta td}{2}\delta_x + \frac{\Delta ta}{2}\delta_x^2)(I - \frac{\Delta te}{2}\delta_y + \frac{\Delta tb}{2}\delta_y^2)(I - \frac{\Delta tf}{2}\delta_z + \frac{\Delta tc}{2}\delta_z^2)C_{i,j,k}^{n} \\
&\quad + \frac{F_{i,j,k}^{n+1} + F_{i,j,k}^{n}}{2}\Delta t\,,
\end{aligned}
\tag{2.79}
$$

$$(I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)C_{i,j,k}^{**} = C_{i,j,k}^{*}\,, \tag{2.80}$$

$$(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)C_{i,j,k}^{n+1} = C_{i,j,k}^{**}\,. \tag{2.81}$$

The solutions to these three sets of equations in 2.79 2.80 and 2.81 can be computed by solving systems of algebraic equations with tri-diagonal coefficient matrices, since the left hand sides of the equations involve only three-point central difference operators δ_x^2, δ_x, δ_y^2, δ_y, δ_z^2, and δ_z. Although the right hand side of Equation 2.79 involves the product of these operators, it does not complicate the solution process since it is applied to the known solution values from the previous time step.

Assuming that $NI = NJ = NK = N$ to simplify the complexity analysis, the dimension of the tri-diagonal Jocobian coefficient matrix is N^3. Since the complexity to solve the system of algebraic equations is proportional to the product of the square of the bandwidth with the dimension of the system [27, 28, 31], the three-dimensional approximate factorization ADI scheme has the total complexity of $O(N^3)$. Compared to the traditional Crank-Nicolson numerical scheme ($O(N^7)$), this approximate factorization scheme provides much more efficient solution to the three-dimensional convection-dispersion-adsorption governing equation.

In general, the three-step approximate factorization only incurs the same order of additional error as the truncation error caused by the Crank-Nicolson method, and reversely calculates the boundary conditions for the intermediate time steps (n^{**} and n^{*}) without further accuracy loss (discussed in Section 2.3.3). This three-step ADI method greatly decreases the computational complexity by applying implicit finite difference method alternatively in different directions, and solving the system of equations with tri-diagonal Jacobian coefficient matrix three times. The approximate factorization method also maintains unconditional stability with second order accuracy in both time and space [19, 29, 32, 31].

Furthermore, the computations to solve the algebraic equation systems are independent, thus, providing enough parallelism for parallel computing [32, 33, 34, 35]. The straightforward parallel version of the three-step ADI algorithm just assigns different equation systems to

parallel processors, so that different processors work on all the independent tri-diagonal systems simultaneously and efficiently (discussed in Chapter IV). Therefore, the three-step approximate factorization ADI method, with unconditional stability and second-order accuracy in both time and space, provides an efficient method for solving the three-dimensional MRTM model.

2.3.3 Boundary Conditions for the ADI Method

Although the ADI method has been widely used for solving partial differential equations (PDE) [29, 36, 37, 38], the accuracy improvement in both time and space for different types of boundary conditions is still an important research topic. Two types of boundary conditions are used in most modeling and simulation problems using convection diffusion equations: 1) the Dirichlet boundary condition that specifies function values of the solution to be calculated, and 2) the Neumann boundary condition that specifies the derivatives of the solution to be calculated. For example, when studying the transport and retention of contaminants in soils with groundwater flow, the Dirichlet boundary conditions are used for the boundary where a solution of known concentration is applied (for example corresponding to a leak in a tank holding the solution), while the Neumann boundary conditions are used for the boundary where the flow rate is known (for example corresponding to a no-flow or impermeable boundary). Sometimes the boundary conditions may also be given in terms of a combination of both function values and derivatives at the same spatial point.

For the fractional time step algorithm given by Equations 2.75 to 2.77, since the solution values calculated at all three fractional time steps are clearly defined at $n+\frac{1}{3}$, $n+\frac{2}{3}$, and $n+1$, it is straightforward to deal with boundary conditions of either type as long as they are given for all fractional time steps. However, this method only provides second-order accuracy in space and first-order accuracy in time with conditional stability.

For the ADI algorithm based on approximate factorization given by Equations 2.79 to 2.81, it is more complicated to deal with boundary conditions. Since the solutions $C^{*}_{i,j,k}$ and $C^{**}_{i,j,k}$, to be calculated in (2.79) and (2.80), respectively, are intermediate variables without clear connection to the physical time level, the given boundary conditions can not be used in a straightforward

way. One way is to apply the known boundary conditions at the time levels $n+\frac{1}{3}$ and $n+\frac{2}{3}$ to $C^{*}_{i,j,k}$ in Equation 2.79 and $C^{**}_{i,j,k}$ in Equation 2.80, respectively. However, this will degrade the accuracy of the algorithm to only first order accurate in time, as demonstrated later in Section 2.3.4.

In order to achieve higher order accuracy, the approximate factorization three-step ADI method requires reverse calculations for the boundary conditions at the intermediate time steps (n^{*} and n^{**}). This research applies two different schemes to achieve second-order accuracy in both time and space with Dirichlet and Neumann boundary conditions.

2.3.3.1 Dirichlet Boundary Condition

The numerical scheme for approximate factorization ADI method to maintain second order accuracy in both time and step with Dirichlet boundary conditions is straightforward [19, 29]. The computations are carried out on a rectangular grid (x_i, y_j, z_k), where $i = 0, 1, ..., NI$, $j = 0, 1, ..., NJ$, and $k = 0, 1, ..., NK$.

$$C(x, y, z, t) = C(i\Delta x,\, j\Delta y,\, k\Delta z,\, n\Delta t) = C^{n}_{i,j,k}\,, \tag{2.82}$$

where $\Delta x = \frac{1}{NI}$, $\Delta y = \frac{1}{NJ}$, and $\Delta z = \frac{1}{NK}$, and $n = 0, 1, 2, \ldots$.

While solving the first step Equation 2.79, boundary conditions for $C^{*}_{0,j,k}$ and $C^{*}_{NI,j,k}$ ($j = 1, 2, ..., NJ-1$ and $k = 1, 2, ..., NK-1$) are calculated reversely by combining the second and third step Equations 2.80 and 2.81 when $i = 0$ and $i = NI$ respectively:

$$C^{*}_{0,j,k} = (I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)C^{n+1}_{0,j,k} \tag{2.83}$$

$$C^{*}_{NI,j,k} = (I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)C^{n+1}_{NI,j,k} \tag{2.84}$$

Extending the left hand side of Equation 2.79:

$$(I + \frac{\Delta td}{2}\delta_x - \frac{\Delta ta}{2}\delta_x^2)C^{*}_{i,j,k}$$

$$
\begin{aligned}
&= C^*_{i,j,k} + \frac{\Delta td}{4\Delta x}(C^*_{i+1,j,k} - C^*_{i-1,j,k}) - \frac{\Delta ta}{2\Delta x^2}(C^*_{i+1,j,k} - 2C^*_{i,j,k} + C^*_{i-1,j,k}) \\
&= (-\frac{\Delta td}{4\Delta x} - \frac{\Delta ta}{2\Delta x^2})C^*_{i-1,j,k} + (1 + \frac{\Delta ta}{\Delta x^2})C^*_{i,j,k} + (\frac{\Delta td}{4\Delta x} - \frac{\Delta ta}{2\Delta x^2})C^*_{i+1,j,k} \\
&= -(r_x + s_x)C^*_{i-1,j,k} + (1 + 2r_x)C^*_{i,j,k} - (r_x - s_x)C^*_{i+1,j,k}\,, \qquad (2.85)
\end{aligned}
$$

where $r_x = \frac{\Delta td}{4\Delta x}$, and $s_x = \frac{\Delta ta}{2\Delta x^2}$. Therefore, the equation systems for the first time step have the following form:

$$
\mathbf{L}\mathbf{C}^* = \mathbf{R}\mathbf{C}^n + \mathbf{B}^* + \frac{\Delta t}{2}(\mathbf{F}^n + \mathbf{F}^{n+1})\,, \qquad (2.86)
$$

where

$$
\mathbf{L} = \begin{bmatrix}
1 + 2r_x & -(r_x - s_x) & & & & \\
-(r_x + s_x) & 1 + 2r_x & -(r_x - s_x) & & & \\
& \ddots & \ddots & \ddots & & \\
& & -(r_x + s_x) & 1 + 2r_x & -(r_x - s_x) & \\
& & & -(r_x + s_x) & 1 + 2r_x &
\end{bmatrix},
$$

$$
\mathbf{C} = \begin{bmatrix} C_{1,j,k} \\ C_{2,j,k} \\ \vdots \\ \vdots \\ C_{NI-1,j,k} \end{bmatrix}, \quad
\mathbf{B}^* = \begin{bmatrix} (r_x + s_x)C^*_{0,j,k} \\ 0 \\ \vdots \\ 0 \\ (r_x - s_x)C^*_{NI,j,k} \end{bmatrix}, \quad
\mathbf{F} = \begin{bmatrix} F_{1,j,k} \\ F_{2,j,k} \\ \vdots \\ \vdots \\ F_{NI-1,j,k} \end{bmatrix},
$$

and **R** comes from the extension of the right hand side of Equation 2.79:

$$
(I - \frac{\Delta td}{2}\delta_x + \frac{\Delta ta}{2}\delta_x^2)(I - \frac{\Delta te}{2}\delta_y + \frac{\Delta tb}{2}\delta_y^2)(I - \frac{\Delta tf}{2}\delta_z + \frac{\Delta tc}{2}\delta_z^2)C^n_{i,j,k}\,.
$$

While solving the second step Equation 2.80, boundary conditions for $C^{**}_{i,0,k}$ and $C^{**}_{i,NJ,k}$ ($i = 1, 2, ..., NI - 1$ and $k = 1, 2, ..., NK - 1$) are calculated reversely from the third step

Equation 2.81 when $j = 0$ and $j = NJ$ respectively:

$$C^{**}_{i,0,k} = (I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)C^{n+1}_{i,0,k} \tag{2.87}$$

$$C^{**}_{i,NJ,k} = (I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)C^{n+1}_{i,NJ,k} \tag{2.88}$$

The equation systems for the second time step have the following form:

$$\mathbf{L}\mathbf{C}^{**} = \mathbf{C}^{*} + \mathbf{B}^{**}\,, \tag{2.89}$$

where

$$\mathbf{L} = \begin{bmatrix} 1+2r_y & -(r_y - s_y) & & & & \\ -(r_y + s_y) & 1+2r_y & -(r_y - s_y) & & & \\ & \ddots & \ddots & \ddots & & \\ & & -(r_y + s_y) & 1+2r_y & -(r_y - s_y) & \\ & & & -(r_y + s_y) & 1+2r_y & \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} C_{i,1,k} \\ C_{i,2,k} \\ \vdots \\ \vdots \\ C_{i,NJ-1,k} \end{bmatrix}, \quad \mathbf{B}^{**} = \begin{bmatrix} (r_y + s_y)C^{**}_{i,0,k} \\ 0 \\ \vdots \\ 0 \\ (r_y - s_y)C^{**}_{i,NJ,k} \end{bmatrix},$$

and $r_y = \frac{\Delta te}{4\Delta y}$, and $s_y = \frac{\Delta tb}{2\Delta y^2}$.

While solving the third step Equation 2.81, boundary conditions for $C^{n+1}_{i,j,0}$ and $C^{n+1}_{i,j,NK}$ ($i = 1, 2, ..., NI - 1$ and $k = 1, 2, ..., NK - 1$) are given directly without the need of reverse

calculations. The equation systems for the third time step have the following form:

$$\mathbf{L}\mathbf{C}^{n+1} = \mathbf{C}^{**} + \mathbf{B}^{n+1}\,, \tag{2.90}$$

where

$$\mathbf{L} = \begin{bmatrix} 1+2r_z & -(r_z - s_z) & & & & \\ -(r_z + s_z) & 1+2r_z & -(r_z - s_z) & & & \\ & \ddots & \ddots & \ddots & & \\ & & -(r_z + s_z) & 1+2r_z & -(r_z - s_z) & \\ & & & -(r_z + s_z) & 1+2r_z & \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} C_{i,j,1} \\ C_{i,j,2} \\ \vdots \\ \vdots \\ C_{i,j,NK-1} \end{bmatrix}, \quad \mathbf{B^{n+1}} = \begin{bmatrix} (r_z + s_z)C^{n+1}_{i,j,0} \\ 0 \\ \vdots \\ 0 \\ (r_z - s_z)C^{n+1}_{i,j,NK} \end{bmatrix},$$

$r_z = \frac{\Delta t f}{4\Delta z}$, and $s_z = \frac{\Delta t c}{2\Delta z^2}$.

2.3.3.2 Neumann Boundary Condition

Since the above approach does not apply to the Neumann boundary conditions, a new method is developed to utilize the given derivative boundary conditions to solve Equations 2.79 to 2.81. For Neumann boundary conditions, the computations are carried out on a rectangular grid (x_i, y_j, z_k), where $i = 0, 1, ..., NI + 2$, $j = 0, 1, ..., NJ + 2$, and $k = 0, 1, ..., NK + 2$.

$$C(x, y, z, t) = C(i\Delta x,\, j\Delta y,\, k\Delta z,\, n\Delta t) = C^n_{i,j,k}, \tag{2.91}$$

where $\Delta x = \frac{1}{NI}$, $\Delta y = \frac{1}{NJ}$, and $\Delta z = \frac{1}{NK}$, and $n = 0, 1, 2, ...$ Here, first-order derivatives are given on the surfaces of the computational domain (where $i = 1$, $i = NI + 1$, $j = 1$, $j = NJ+1$, $k = 1$, and $k = NK+1$). In order to construct second order approximation for the Neumann boundary conditions, an additional layer of "ghost points" is created on each side of the boundary. Therefore, the computational domain now includes all points (x_i, y_j, z_k), where $i = 0, 1, ..., NI + 2$, $j = 0, 1, ..., NJ + 2$, and $k = 0, 1, ..., NK + 2$.

While solving the first step Equation 2.79, boundary conditions for $C^*_{0,j,k}$ and $C^*_{NI+2,j,k}$ ($j = 1, 2, ..., NJ + 1$ and $k = 1, 2, ..., NK + 1$) are calculated reversely by combining the second and third step Equations 2.80 and 2.81 and applying a second order central difference operator δ_x to both sides where $i = 1$ and $i = NI + 1$ respectively:

$$\delta_x C^*_{1,j,k} = \delta_x (I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2) C^{n+1}_{1,j,k} \tag{2.92}$$

$$\delta_x C^*_{NI+1,j,k} = \delta_x (I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2) C^{n+1}_{NI+1,j,k} \tag{2.93}$$

Due to the commutativity of the right hand side operators δ_x, δ_y, δ_z, and the relation $\delta_x C \approx C_x$ (C_x is known as boundary conditions), these two equations are expressed as:

$$\frac{C^*_{2,j,k} - C^*_{0,j,k}}{2\Delta x} = (I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)(C_x)^{n+1}_{1,j,k}$$

$$\frac{C^*_{NI+2,j,k} - C^*_{NI,j,k}}{2\Delta x} = (I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)(C_x)^{n+1}_{NI+1,j,k}$$

or

$$C^*_{0,j,k} = C^*_{2,j,k} - 2\Delta x (I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)(C_x)^{n+1}_{1,j,k} \tag{2.94}$$

$$C^*_{NI+2,j,k} = C^*_{NI,j,k} + 2\Delta x (I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)(C_x)^{n+1}_{NI+1,j,k} \tag{2.95}$$

Therefore, the left hand side of Equation 2.79 are extended as:

$$
\begin{aligned}
&(I + \frac{\Delta td}{2}\delta_x - \frac{\Delta ta}{2}\delta_x^2)C^*_{i,j,k} \\
&= -(r_x + s_x)C^*_{i-1,j,k} + (1 + 2r_x)C^*_{i,j,k} - (r_x - s_x)C^*_{i+1,j,k}\,,
\end{aligned} \tag{2.96}
$$

when $i = 2, 3, ..., NI$, $r_x = \frac{\Delta td}{4\Delta x}$, and $s_x = \frac{\Delta ta}{2\Delta x^2}$, with

$$
\begin{aligned}
&-(r_x + s_x)C^*_{0,j,k} + (1 + 2r_x)C^*_{1,j,k} - (r_x - s_x)C^*_{2,j,k} \\
&= -(r_x + s_x)(C^*_{2,j,k} - 2\Delta x B^*_{1,j,k}) + (1 + 2r_x)C^*_{1,j,k} - (r_x - s_x)C^*_{2,j,k} \\
&= 2\Delta x(r_x + s_x)B^*_{1,j,k} + (1 + 2r_x)C^*_{1,j,k} - 2r_x C^*_{2,j,k}\,,
\end{aligned} \tag{2.97}
$$

and

$$
\begin{aligned}
&-(r_x + s_x)C^*_{NI,j,k} + (1 + 2r_x)C^*_{NI+1,j,k} - (r_x - s_x)C^*_{NI+2,j,k} \\
&= -(r_x + s_x)C^*_{NI,j,k} + (1 + 2r_x)C^*_{NI+1,j,k} - (r_x - s_x)(C^*_{NI,j,k} + 2\Delta x B^*_{NI+1,j,k}) \\
&= -2r_x C^*_{NI,j,k} + (1 + 2r_x)C^*_{NI+1,j,k} - 2\Delta x(r_x - s_x)B^*_{NI+1,j,k}\,,
\end{aligned} \tag{2.98}
$$

where $B^*_{1,j,k} = (I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)(C_x)^{n+1}_{1,j,k}$,
and $B^*_{NI+1,j,k} = (I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)(C_x)^{n+1}_{NI+1,j,k}$.

The equation systems for the first time step have the following form:

$$
\mathbf{L}\mathbf{C}^* = \mathbf{R}\mathbf{C}^n + \mathbf{B}^* + \frac{\Delta t}{2}(\mathbf{F}^n + \mathbf{F}^{n+1})\,, \tag{2.99}
$$

where

$$\mathbf{L} = \begin{bmatrix} 1+2r_x & -2r_x & & & \\ -(r_x+s_x) & 1+2r_x & -(r_x-s_x) & & \\ & \ddots & \ddots & \ddots & \\ & & -(r_x+s_x) & 1+2r_x & -(r_x-s_x) \\ & & & -2r_x & 1+2r_x \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} C_{1,j,k} \\ C_{2,j,k} \\ \vdots \\ \vdots \\ C_{NI+1,j,k} \end{bmatrix}, \quad \mathbf{B}^* = \begin{bmatrix} -2\Delta x(r_x+s_x)B^*_{1,j,k} \\ 0 \\ \vdots \\ 0 \\ 2\Delta x(r_x-s_x)B^*_{NI+1,j,k} \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} F_{1,j,k} \\ F_{2,j,k} \\ \vdots \\ \vdots \\ F_{NI+1,j,k} \end{bmatrix},$$

and **R** comes from the extension of the right hand side of Equation 2.79:

$$(I - \frac{\Delta td}{2}\delta_x + \frac{\Delta ta}{2}\delta_x^2)(I - \frac{\Delta te}{2}\delta_y + \frac{\Delta tb}{2}\delta_y^2)(I - \frac{\Delta tf}{2}\delta_z + \frac{\Delta tc}{2}\delta_z^2)C^n_{i,j,k}\,.$$

When solving the second step Equation 2.80, boundary conditions for $C^{**}_{i,0,k}$ and $C^{**}_{i,NJ+2,k}$ ($i = 1, 2, ..., NI+1$ and $k = 1, 2, ..., NK+1$) are calculated reversely from the third step equation by applying a second order central difference operator δ_y to both sides where $j = 1$ and $j = NJ+1$ respectively:

$$\delta_y C^{**}_{i,1,k} = \delta_y(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)C^{n+1}_{i,1,k} \tag{2.100}$$

$$\delta_y C^{**}_{i,NJ+1,k} = \delta_y(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)C^{n+1}_{i,NJ+1,k} \tag{2.101}$$

so

$$C^{**}_{i,0,k} = C^{**}_{i,2,k} - 2\Delta y(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)(C_y)^{n+1}_{i,1,k} \tag{2.102}$$

$$C^{**}_{i,NJ+2,k} = C^{**}_{i,NJ,k} + 2\Delta y(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)(C_y)^{n+1}_{i,NJ+1,k} \tag{2.103}$$

The equation systems for the second time step have the following form:

$$\mathbf{L}\mathbf{C}^{**} = \mathbf{C}^{*} + \mathbf{B}^{**}\,, \tag{2.104}$$

where

$$\mathbf{L} = \begin{bmatrix} 1+2r_y & -2r_y & & & & \\ -(r_y+s_y) & 1+2r_y & -(r_y-s_y) & & & \\ & \ddots & \ddots & \ddots & & \\ & & -(r_y+s_y) & 1+2r_y & -(r_y-s_y) & \\ & & & -2r_y & 1+2r_y & \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} C_{i,1,k} \\ C_{i,2,k} \\ \vdots \\ \vdots \\ C_{i,NJ+1,k} \end{bmatrix}, \quad \mathbf{B}^{**} = \begin{bmatrix} -2\Delta y(r_y+s_y)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)(C_y)^{n+1}_{i,1,k} \\ 0 \\ \vdots \\ 0 \\ 2\Delta y(r_y-s_y)(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)(C_y)^{n+1}_{i,NJ+1,k} \end{bmatrix},$$

$r_y = \frac{\Delta te}{4\Delta y}$, and $s_y = \frac{\Delta tb}{2\Delta y^2}$.

When solving the third step Equation 2.81, boundary conditions for $C^{n+1}_{i,j,0}$ and $C^{n+1}_{i,j,NK+2}$ ($i = 1, 2, ..., NI+1$ and $k = 1, 2, ..., NK+1$) are given directly without the need of reverse calculations. The equation systems for the third time step have the following form:

$$\mathbf{L}\mathbf{C}^{n+1} = \mathbf{C}^{**} + \mathbf{B}^{n+1}\,, \tag{2.105}$$

where

$$\mathbf{L} = \begin{bmatrix} 1+2r_z & -2r_z & & & \\ -(r_z+s_z) & 1+2r_z & -(r_z-s_z) & & \\ & \ddots & \ddots & \ddots & \\ & & -(r_z+s_z) & 1+2r_z & -(r_z-s_z) \\ & & & -2r_z & 1+2r_z \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} C_{i,j,1} \\ C_{i,j,2} \\ \vdots \\ \vdots \\ C_{i,j,NK+1} \end{bmatrix}, \quad \mathbf{B^{n+1}} = \begin{bmatrix} -2\Delta z(r_z+s_z)(C_z)^{n+1}_{i,j,1} \\ 0 \\ \vdots \\ 0 \\ 2\Delta z(r_z-s_z)(C_z)^{n+1}_{i,j,NK+1} \end{bmatrix},$$

$r_z = \frac{\Delta t f}{4\Delta z}$, and $s_z = \frac{\Delta t c}{2\Delta z^2}$.

2.3.3.3 Ghost Points

Since the reverse calculations to approximate Neumann boundary conditions apply some additional operators on the first-order derivatives, additional boundary conditions at the "ghost points" are needed as shown in Figure 2.2.

When solving the first step Equation 2.79, the combination of operators δ_y and δ_z is applied to $(C_x)_{1,j,k}$ and $(C_x)_{NI+1,j,k}$, where $1 \leq j \leq NJ+1$ and $1 \leq k \leq NK+1$ (Equations 2.94 and 2.95). The Neumann boundary conditions just provide first-order derivatives at points (x_i, y_j, z_k), where $1 \leq i \leq NI+1$, $1 \leq j \leq NJ+1$, and $1 \leq k \leq NK+1$. Therefore, additional first-order derivatives are needed along the following eight lines:

$$\begin{aligned} (x_1, y_0, z_0) &\rightarrow (x_1, y_{NJ+2}, z_0) \\ (x_{NI+1}, y_0, z_0) &\rightarrow (x_{NI+1}, y_{NJ+2}, z_0) \\ (x_1, y_0, z_{NK+2}) &\rightarrow (x_1, y_{NJ+2}, z_{NK+2}) \end{aligned}$$

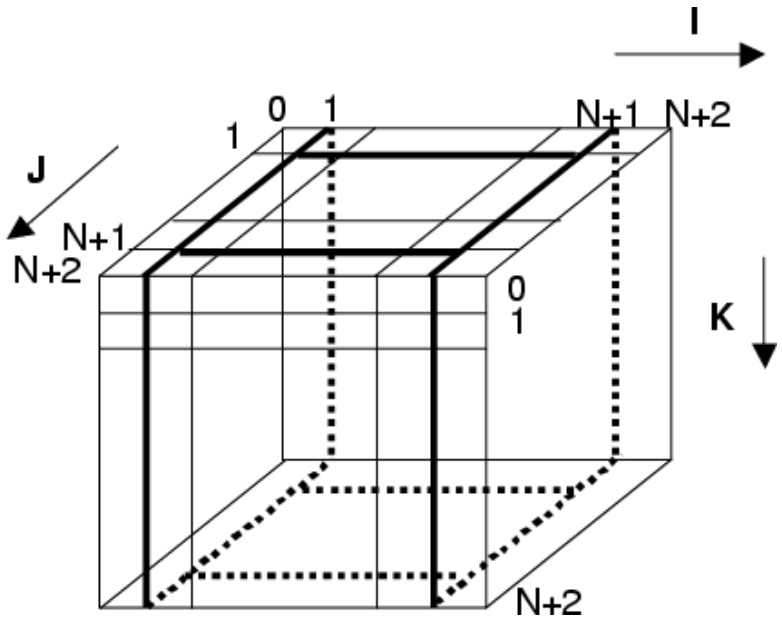

Figure 2.2: Ghost points for Neumann boundary conditions

$$\begin{aligned}
(x_{NI+1}, y_0, z_{NK+2}) &\rightarrow (x_{NI+1}, y_{NJ+2}, z_{NK+2}) \\
(x_1, y_0, z_0) &\rightarrow (x_1, y_0, z_{NK+2}) \\
(x_1, y_{NJ+2}, z_0) &\rightarrow (x_1, y_{NJ+2}, z_{NK+2}) \\
(x_{NI+1}, y_0, z_0) &\rightarrow (x_{NI+1}, y_0, z_{NK+2}) \\
(x_{NI+1}, y_{NJ+2}, z_0) &\rightarrow (x_{NI+1}, y_{NJ+2}, z_{NK+2})
\end{aligned} \tag{2.106}$$

Similarly, when solving the second step Equation 2.80, the operator δ_z is applied to $(C_y)_{i,1,k}$ and $(C_y)_{i,NJ+1,k}$, where $1 \leq i \leq NI+1$ and $1 \leq k \leq NK+1$ (Equations 2.102 and 2.103). Therefore, additional first-order derivatives are needed along the following four lines:

$$\begin{aligned}
(x_1, y_1, z_0) &\rightarrow (x_{NI+1}, y_1, z_0) \\
(x_1, y_{NJ+1}, z_0) &\rightarrow (x_{NI+1}, y_{NJ+1}, z_0) \\
(x_1, y_1, z_{NK+2}) &\rightarrow (x_{NI+1}, y_1, z_{NK+2}) \\
(x_1, y_{NJ+1}, z_{NK+2}) &\rightarrow (x_{NI+1}, y_{NJ+1}, z_{NK+2})
\end{aligned} \tag{2.107}$$

The additional boundary conditions at the "ghost points" are also needed to update the boundary solution values after each iteration described as the following three steps. First, $C_{i,j,0}$ and $C_{i,j,NK+2}$ are updated where $1 \leq i \leq NI+1$, and $1 \leq j \leq NJ+1$ without the help of additional boundary conditions. Second, $C_{i,0,k}$ and $C_{i,NJ+2,k}$ are updated where $1 \leq i \leq NI+1$, and $0 \leq k \leq NK+2$ with the help of additional first-order derivatives along the four lines in Equation 2.107. Finally, $C_{0,j,k}$ and $C_{NI+2,j,k}$ are updated where $0 \leq j \leq NJ+2$, and $0 \leq k \leq NK+2$ with the help of additional first-order derivatives along the eight lines in Equation 2.106. These updated boundary solution values on the "ghost points" are required by the right hand side of Equation 2.79, because of the similar effects from operators δ_x, δ_y, and δ_z.

The additional first-order derivatives at the "ghost points" can be calculated by the second order extrapolations from the known derivatives at the neighbor points. In the one-dimensional case, the solution value at any point can expressed as the combination of solution values at three neighbor points:

$$f(0) = af(h) + bf(2h) + cf(3h) + O(h^2) \tag{2.108}$$

Using Taylor Series,

$$\begin{aligned} f(h) &= f(0) + f'(0)h + \frac{f''(0)}{2}h^2 + O(h^2) \\ f(2h) &= f(0) + f'(0)2h + \frac{f''(0)}{2}4h^2 + O(h^2) \\ f(3h) &= f(0) + f'(0)3h + \frac{f''(0)}{2}9h^2 + O(h^2) \end{aligned}$$

We have following relations:

$$\begin{aligned} a + b + c &= 1 \\ a + 2b + 3c &= 0 \\ a + 4b + 9c &= 0 \end{aligned}$$

or

$$a = 3$$

$$b = -3$$

$$c = 1$$

Therefore

$$f(0) = 3f(h) - 3f(2h) + f(3h) + O(h^2) \tag{2.109}$$

Equation 2.109 also applies to the three-dimensional domain, so as to calculate the first-order derivatives at the "ghost points". For example, the additional first-order derivatives along the four lines (Equation 2.107) for the second step Equation 2.80 are solved using the given Neumann boundary conditions:

$$(C_y)_{i,1,0} = 3(C_y)_{i,1,1} - 3(C_y)_{i,1,2} + (C_y)_{i,1,3}$$
$$(C_y)_{i,NJ+1,0} = 3(C_y)_{i,NJ+1,1} - 3(C_y)_{i,NJ+1,2} + (C_y)_{i,NJ+1,3}$$
$$(C_y)_{i,1,NK+2} = 3(C_y)_{i,1,NK+1} - 3(C_y)_{i,1,NK} + (C_y)_{i,1,NK-1}$$
$$(C_y)_{i,NJ+1,NK+2} = 3(C_y)_{i,NJ+1,NK+1} - 3(C_y)_{i,NJ+1,NK} + (C_y)_{i,NJ+1,NK-1}\,,$$

where $1 \leq i \leq NI + 1$.

Similarly, the first-order derivatives along the eight lines (Equation 2.106) for the first step Equation 2.79 are calculated using the given Neumann boundary conditions. The total truncation error caused by Taylor Series is $O(\Delta x^2, \Delta y^2, \Delta z^2)$, thus the complete approximate factorization ADI method maintains second order accuracy in both time and space.

2.3.4 Numerical Experiments

Numerical experiments are presented to demonstrate the accuracy and stability of three numerical methods: the fractional time step method, the approximate factorization ADI method

without reverse boundary calculations, and the approximate factorization ADI method with reverse boundary calculations. The following three-dimensional convection-dispersion equation is used in the numerical experiments:

$$C_t + C_x + C_y + C_z = C_{xx} + C_{yy} + C_{zz} + F(x, y, z, t)\,, \tag{2.110}$$

$$0 < t < 1\,, \quad (x, y, z) \in [0, 1] \times [0, 1] \times [0, 1]\,, \tag{2.111}$$

2.3.4.1 Case 1

To accommodate the exact solution

$$C(x, y, z, t) = e^{-\frac{t}{3}} \sin\frac{x}{3} \sin\frac{y}{3} \sin\frac{z}{3} \tag{2.112}$$

Initial condition is known as:

$$C(x, y, z, 0) = \sin\frac{x}{3} \sin\frac{y}{3} \sin\frac{z}{3}\,,$$

with the source term:

$$F(x, y, z, t) = \frac{1}{3} e^{-\frac{t}{3}} (\cos\frac{x}{3} \sin\frac{y}{3} \sin\frac{z}{3} + \sin\frac{x}{3} \cos\frac{y}{3} \sin\frac{z}{3} + \sin\frac{x}{3} \sin\frac{y}{3} \cos\frac{z}{3})\,.$$

Example 1: Dirichlet boundary conditions

$$\begin{aligned}
C(0, y, z, t) &= 0\,, \\
C(1, y, z, t) &= e^{-\frac{t}{3}} \sin\frac{1}{3} \sin\frac{y}{3} \sin\frac{z}{3}\,, \\
C(x, 0, z, t) &= 0\,, \\
C(x, 1, z, t) &= e^{-\frac{t}{3}} \sin\frac{x}{3} \sin\frac{1}{3} \sin\frac{z}{3}\,, \\
C(x, y, 0, t) &= 0\,, \\
C(x, y, 1, t) &= e^{-\frac{t}{3}} \sin\frac{x}{3} \sin\frac{y}{3} \sin\frac{1}{3}
\end{aligned}$$

Table 2.1: Maximum error between the calculated solution and the exact solution at $T = 1.0$ with Dirichlet boundary conditions. e_1 is the maximum error from the fractional time step method (1st order); e_2 is the maximum error from the approximate factorization ADI method without reverse boundary condition calculations; e_3 is the maximum error from the approximate factorization ADI method (2nd order) with reverse boundary condition calculations, where $\Delta t = \Delta x = \Delta y = \Delta z = h$. e_4 is the maximum error from the fractional time step method (1st order) with $\Delta t = h^2$.

h	0.5	0.25	0.125	0.0625	0.03125	0.015625
e_1	3.3115e-06	1.7632e-05	8.8417e-01	3.8925e+13	∞	∞
e_2	4.6012e-04	8.7192e-04	8.8201e-04	7.3443e-04	6.1301e-04	5.4187e-04
e_3	1.6865e-05	8.5828e-06	2.8530e-06	7.7917e-07	2.0128e-07	5.0645e-08
e_4	9.6212e-08	6.0264e-07	6.7389e-08	5.7439e-09	7.3950e-10	

The data in Table 4.1 shows the maximum error between the numerical solutions obtained using different methods and the exact solution at $t = 1.0$ with Dirichlet boundary conditions. The computation grid is $\Delta t = \Delta x = \Delta y = \Delta z = h$. Three methods are considered: the notation e_1 represents the error $||\cdot||_\infty$, between the exact solution and the numerical solution calculated using the fractional time step method (Equations 2.75 to 2.77); e_2 represents the error between the exact solution and the numerical solution calculated using the approximate factorization ADI method (Equations 2.79 to 2.81) defining the Dirichlet boundary conditions at the time levels $n + \frac{1}{3}$ and $n + \frac{2}{3}$; and e_3 represents the error between the exact solution and the numerical solution calculated using the approximate factorization ADI method (Equations 2.79 to 2.81) and the reverse calculations (Equations 2.83, 2.84, 2.87, and 2.88) to handle Dirichlet boundary conditions. e_4 represents the error between the exact solution and the numerical solution calculated using the fractional time step method with the computation grid $\Delta t = h^2$. It is clear from the table that e_1 shows that the fractional time step method is only conditionally stable, e_2 demonstrates no accuracy improvement when the grid is refined, e_3 demonstrates second order accuracy in both time and space, and e_4 shows that the fractional time step method is first order accurate in time and second order in space. Since the relation $\Delta t = h^2$ to maintain the stability for the fractional time step method increases computational time greatly, the test case e_4 for $h = 0.015625$ is omitted.

Example 2: Neumann boundary conditions

$$
\begin{aligned}
(C_x)(0,y,z,t) &= \frac{1}{3}e^{-\frac{t}{3}}\sin\frac{y}{3}\sin\frac{z}{3}, \\
(C_x)(1,y,z,t) &= \frac{1}{3}e^{-\frac{t}{3}}\cos\frac{1}{3}\sin\frac{y}{3}\sin\frac{z}{3}, \\
(C_y)(x,0,z,t) &= \frac{1}{3}e^{-\frac{t}{3}}\sin\frac{x}{3}\sin\frac{z}{3}, \\
(C_y)(x,1,z,t) &= \frac{1}{3}e^{-\frac{t}{3}}\sin\frac{x}{3}\cos\frac{1}{3}\sin\frac{z}{3}, \\
(C_z)(x,y,0,t) &= \frac{1}{3}e^{-\frac{t}{3}}\sin\frac{x}{3}\sin\frac{y}{3}, \\
(C_z)(x,y,1,t) &= \frac{1}{3}e^{-\frac{t}{3}}\sin\frac{x}{3}\sin\frac{y}{3}\cos\frac{1}{3}
\end{aligned}
$$

Table 2.2: Maximum error between the calculated solution and the exact solution at $T = 1.0$ with Neumann boundary conditions. e_1 is the maximum error from the fractional time step method (1st order); e_2 is the maximum error from the approximate factorization ADI method (1st order) without reverse boundary condition calculations; e_3 is the maximum error from the approximate factorization ADI method (2nd order) with reverse boundary condition calculations, where $\Delta t = \Delta x = \Delta y = \Delta z = h$. e_4 is the maximum error from the fractional time step method (1st order) where $\Delta t = h^2$.

h	0.5	0.25	0.125	0.0625	0.03125	0.015625
e_1	2.6981e-04	2.6970e-04	6.8894e+00	5.2292e+14	8.7986e+48	∞
e_2	6.5340e-03	3.2795e-03	1.6240e-03	8.0441e-04	3.9933e-04	1.9869e-04
e_3	2.3049e-03	3.7786e-04	8.4547e-05	2.1123e-05	5.2850e-06	1.3221e-06
e_4	1.2207e-04	2.5438e-05	6.0374e-06	1.4892e-06	3.7104e-07	

The data in Table 2.2 shows the maximum error between the numerical solutions obtained using different methods and the exact solution at $t = 1.0$ with Neumann boundary conditions. The computation grid is $\Delta t = \Delta x = \Delta y = \Delta z = h$. Three methods are considered: the notation e_1 represents the error $||\cdot||_\infty$, between the exact solution and the numerical solution calculated using the fractional time step method (Equations 2.75 to 2.77); e_2 represents the error between the exact solution and the numerical solution calculated using the approximate factorization ADI method (Equations 2.79 to 2.81) defining the Neumann boundary conditions at the time levels $n+\frac{1}{3}$ and $n+\frac{2}{3}$; and e_3 represents the error between the exact solution and the

numerical solution calculated using the approximate factorization ADI method (Equations 2.79 to 2.81) and the reverse calculations (Equations 2.94, 2.95, 2.102, and 2.103) to handle Neumann boundary conditions. e_4 represents the error between the exact solution and the numerical solution calculated using the fractional time step method with the computation grid $\Delta t = h^2$. It is clear from the table that e_1 shows that the fractional time step method is only conditionally stable, e_2 demonstrates first order accuracy when the grid is refined, e_3 demonstrates second order accuracy in both time and space, and e_4 shows that the fractional time step method is first order accurate in time and second order in space. Since the relation $\Delta t = h^2$ to maintain the stability for the fractional time step method increases computational time greatly, the test case e_4 for $h = 0.015625$ is omitted.

2.3.4.2 Case 2

In comparison, another exact solution is provided:

$$C(x, y, z, t) = e^{-t}(x^6 + y^6 + z^6) \tag{2.113}$$

Initial condition is known as:

$$C(x, y, z, 0) = x^6 + y^6 + z^6 \,,$$

with the source term:

$$F(x, y, z, t) = e^{-t}(6x^5 + 6y^5 + 6z^5 - x^6 - y^6 - z^6 - 30x^4 - 30y^4 - 30z^4) \,.$$

Example 3: Dirichlet boundary conditions

$$\begin{aligned} C(0, y, z, t) &= e^{-t}(y^6 + z^6) \,, \\ C(1, y, z, t) &= e^{-t}(1 + y^6 + z^6) \,, \\ C(x, 0, z, t) &= e^{-t}(x^6 + z^6) \,, \end{aligned}$$

$$
\begin{aligned}
C(x,1,z,t) &= e^{-t}(x^6+1+z^6)\,, \\
C(x,y,0,t) &= e^{-t}(x^6+y^6)\,, \\
C(x,y,1,t) &= e^{-t}(x^6+y^6+1)
\end{aligned}
$$

Table 2.3: Maximum error between the calculated solution and the exact solution at $T = 1.0$ with Dirichlet boundary conditions. e_1 is the maximum error from the fractional time step method (1st order); e_2 is the maximum error from the approximate factorization ADI method without reverse boundary condition calculations; e_3 is the maximum error from the approximate factorization ADI method (2nd order) with reverse boundary condition calculations, where $\Delta t = \Delta x = \Delta y = \Delta z = h$. e_4 is the maximum error from the fractional time step method (1st order) where $\Delta t = h^2$.

h	0.5	0.25	0.125	0.0625	0.03125	0.015625
e_1	3.7756e-02	1.0754e-01	3.2553e+03	1.6143e+17	∞	∞
e_2	1.3189e-01	1.4060e-01	1.1896e-01	9.7283e-02	7.6985e-02	6.2900e-02
e_3	3.0171e-02	1.3570e-02	4.8177e-03	1.2998e-03	3.3199e-04	8.3617e-05
e_4	5.7090e-02	1.7492e-02	5.3140e-03	1.3724e-03	3.4728e-04	

Table 2.3 shows the maximum error between the numerical solutions obtained using different methods and the exact solution at $t = 1.0$ with Dirichlet boundary conditions. Although the experiments with polynomial functions have larger errors than those with trigonometrical functions, they reflect the same trend of accuracy and stability. It is clear from the table that e_1 shows that the fractional time step method is only conditionally stable, e_2 demonstrates no significant accuracy improvement when the grid is refined, e_3 demonstrates second order accuracy in both time and space, and e_4 shows that the fractional time step method is first order accurate in time and second order in space.

Example 4: Neumann boundary conditions

$$
\begin{aligned}
(C_x)(0,y,z,t) &= 0\,, \\
(C_x)(1,y,z,t) &= 6e^{-t}\,, \\
(C_y)(x,0,z,t) &= 0\,,
\end{aligned}
$$

$$
\begin{aligned}
(C_y)(x,1,z,t) &= 6e^{-t}\,, \\
(C_z)(x,y,0,t) &= 0\,, \\
(C_z)(x,y,1,t) &= 6e^{-t}
\end{aligned}
$$

Table 2.4: Maximum error between the calculated solution and the exact solution at $T = 1.0$ with Neumann boundary conditions. e_1 is the maximum error from the fractional time step method (1st order); e_2 is the maximum error from the approximate factorization ADI method without reverse boundary condition calculations; e_3 is the maximum error from the approximate factorization ADI method (2nd order) with reverse boundary condition calculations, where $\Delta t = \Delta x = \Delta y = \Delta z = h$. e_4 is the maximum error from the fractional time step method (1st order) where $\Delta t = h^2$.

h	0.5	0.25	0.125	0.0625	0.03125	0.015625
e_1	4.7534e+00	4.9861e+00	1.2231e+05	1.3721e+17	∞	∞
e_2	4.2207e+00	9.2392e-01	1.4718e-01	2.0545e-02	2.7906e-02	1.8562e-02
e_3	7.2060e+00	1.4783e+00	3.4704e-01	8.4809e-02	2.1011e-02	5.2321e-03
e_4	5.1894e+00	1.3038e+00	3.2638e-01	8.1626e-02	2.0408e-02	

Table 2.4 shows the maximum error between the numerical solutions obtained using different methods and the exact solution at $t = 1.0$ with Neumann boundary conditions. Although the experiments with polynomial functions have larger errors than those with trigonometrical functions, they reflect the similar trend of accuracy and stability. It is clear from the table that e_1 shows that the fractional time step method is only conditionally stable, e_2 demonstrates no consistent first order accuracy when the grid is refined, e_3 demonstrates second order accuracy in both time and space, and e_4 shows that the fractional time step method is first order accurate in time and second order in space.

In conclusion with all the numerical experiments, the fractional time step method is conditionally stable with first order accuracy in time and second order accuracy in space; the approximate factorization ADI method without reverse boundary condition calculations is unconditionally stable without a consistent or predictable accuracy; and the approximate

factorization ADI method with the reverse boundary condition calculations is unconditionally stable with second order accuracy in both time and space.

CHAPTER III

WEB-BASED SYSTEM INTEGRATION

With the development of Internet, on-line access to scientific computing becomes a new trend. The web-based system integration of computational simulations can provide not only user-friendly graphical user interface (GUI), but also the flexibility to access remote computing resources, manage job information, and even distribute the software. In this research, different web technologies are applied to the one-dimensional and three-dimensional MRTM simulation systems respectively. This chapter covers these web technologies, especially the Java technologies, used in the simulation system integration in detail. The major goals and system requirements of these two different simulations are also compared.

3.1 General Comparison

The one-dimensional MRTM simulation system utilizes the client-side computing resources, because the computational loads in the one-dimensional problem are manageable by the client-side computers. The major goal is to reuse (instead of rewrite) the legacy code written in C or FORTRAN, and to relieve users from tedious command-line operations with Java Swing GUI components [39]. The legacy computational modules are wrapped within the Java code using the Java Native Interface (JNI) technology, and executed in the Java Virtual Machine (JVM) of the web browser [40, 41].

Figure 3.1 shows the simple client-server computing architecture in the one-dimensional web-based simulation system. The legacy FORTRAN and C code is packaged into dynamic libraries with the JNI code, and uploaded to a web server which supports basic HTTP protocols. Java Swing components build powerful graphical user interfaces, which allow users to access the web server (or website) through web browsers. The user-transparency hides from users the mechanism that the JNI libraries are downloaded (through HTTP) to the client side and run

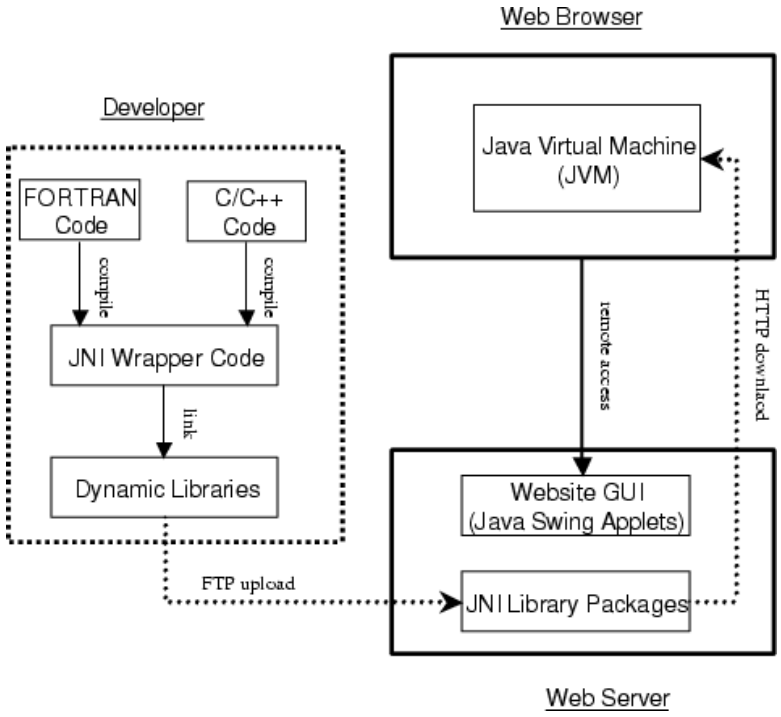

Figure 3.1: System architecture for the one-dimensional MRTM simulation.

inside the JVM of web browsers. However, the JNI dynamic libraries are inevitably platform-dependent, which means developers should compile and build different libraries for different platforms (i.e. Windows or Unix). The client-side computational simulation system also require users to configure their system security, so as to grant the downloaded JNI code permissions to execute computations on their local computers [18, 42].

The computations in the three-dimensional MRTM simulation system are much more time-consuming than those in the one-dimensional MRTM simulation system. Such kind of work load is too much for most client-side computing resources. Therefore, the client-side computing model used in the one-dimensional MRTM simulation system does not fit the three-dimensional cases. High performance (parallel) computing technologies are needed to achieve better performance. This research aims to transfer the traditional high performance computing model to a web-based one, so that users are relieved from tedious text-based terminal operations and manual maintenance of result data files. Java 2 Enterprise Edition (J2EE) technologies [43]

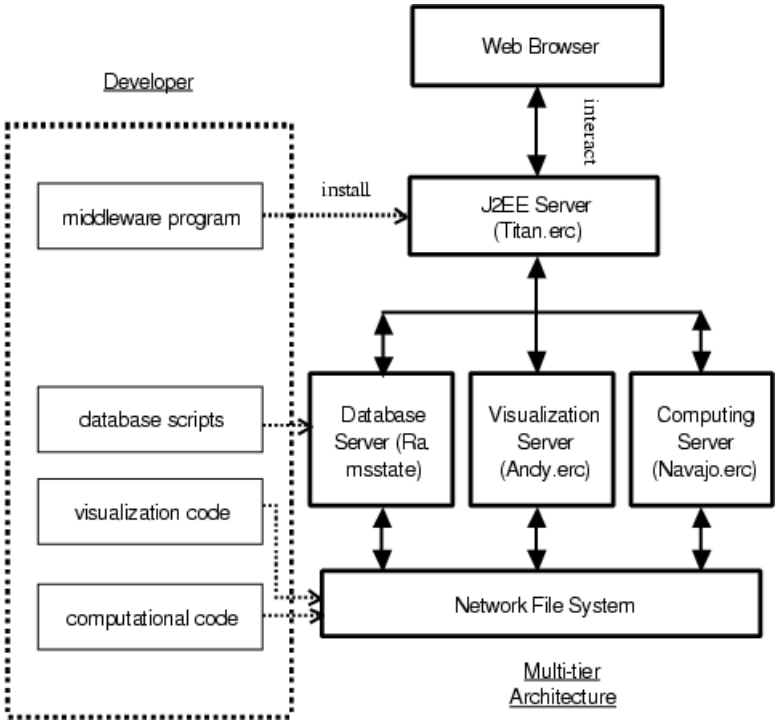

Figure 3.2: System architecture for the three-dimensional MRTM simulation.

enable the server-side computing features of the three-dimensional MRTM simulation system. Inside the multi-tiered computing architecture (Figure 3.2), the middleware program runs inside a J2EE server all the time, and provides web services for the requests to execute the three-dimensional simulation computations. The J2EE server connects the client and server sides smoothly. Computing requests are submitted by the client-side web browser, arranged by the J2EE middleware, and finally sent to the server-side computing resources. The high performance computing is executed on the server-side parallel computers. These parallel computers might be the same machines where the J2EE server runs, but also could be different machines. The computing job information and result data are maintained inside a database, which resides on the additional database server. Although the system architecture becomes much complicated than the simple client-server model in the one-dimensional system, the three-dimensional MRTM simulation system is more flexible, portable, and secure.

3.2 One-dimensional Simulation System and Related Technologies

The one-dimensional web-based simulation environment for retention and transport of organic and inorganic compounds in soil is based on the one-dimensional MRTM transport model by Selim et al. [4, 5]. The core computing components were written in C or FORTRAN for their computational efficiency. The simulation system is accessible from the Internet. Some emerging Java techniques, like Java Native Interface [40, 41] better known as JNI, and Swing [39, 44], are used in the development of the web-based simulation environment. The computations in this simulation system executes on the client side, and the incurred Java client side security problem has been considered and solved [18, 42].

The JNI allows the Java code that runs inside the Java virtual machine (JVM) of the web browser to interoperate with applications and libraries written in other programming languages, such as C, C++, and FORTRAN (Figure 3.1). This makes it possible to reuse legacy code (i.e., existing code) written in other programming languages and enable researchers from any other discipline, in collaboration with computer scientists and mathematicians, to develop large code and add new functions. The code writing process can be done independently according to the researchers' preference of programming language. Those code can be linked afterwards and become, as a whole, the desired application.

3.2.1 Java Technologies in One-dimensional MRTM Simulation

The Java programming language became popular very quickly. During the first year of its existence, Java was already the Internet's adopted programming language. Java, designed to be type-safe and easy to use, is platform-independent. It enables programmers to "Compile once, run everywhere". Java provides features such as automatic memory management, garbage collection, and range checking on strings and arrays. Java applets run on web pages, and can be used to create dynamic and interactive web sites [45, 46].

The Java Foundation Classes (JFC), which encompass a group of features to help people build graphical user interfaces (GUIs), enables developers to easily incorporate high-quality 2D graphics, text, and images in Java applications and applets [39, 47]. The Swing components,

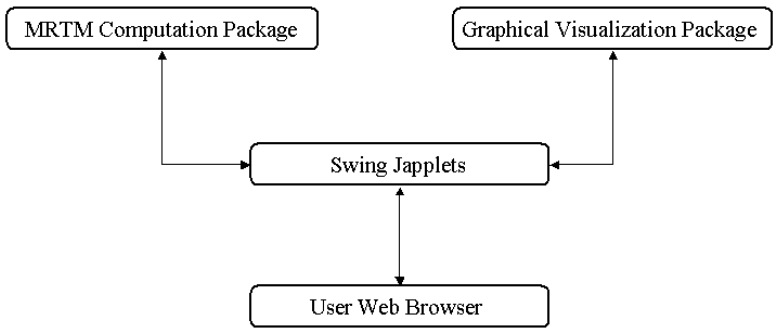

Figure 3.3: Components in the one-dimensional simulation system.

which are the major features of JFC, include everything from buttons to sliders to split panels. Swing, supported by JDK 1.2, has completely replaced Abstract Window Toolkit (AWT), which is the standard Application Programming Interface (API) to provide GUIs in former JDK versions. The biggest difference between Swing and AWT is that Swing components are implemented with pure Java code. Swing provides complete portability, since the pure Java design has less platform specific limitations [45, 47].

However powerful Java is, a pure Java world could not happen over night. Other native languages, such as C/C++, and FORTRAN, have existed for long time, and are still the primary languages for scientific computing. There are enormous invaluable native applications, which do not contain (as Java does) standardized and platform independent supports for creating web-based user interfaces. In order to reuse legacy code and provide web-based features, it is necessary for a programmer to interface Java code with native code written in other languages. In this one-dimensional MRTM simulation, legacy FORTRAN and C code are slightly modified and packaged into dynamical library, so that applets written in Java providing web-based user interface, can call the core computational functions written in C/C++ or FORTRAN smoothly.

The integrated simulation system contains three parts (Figure 3.3). The MRTM computation package, which is written in FORTRAN, deals with all the transport-retention calculations. The graphical visualization package plots two-dimensional curves in Java applets. The Swing-featured applet manages the whole system. It receives input parameters from user web browsers,

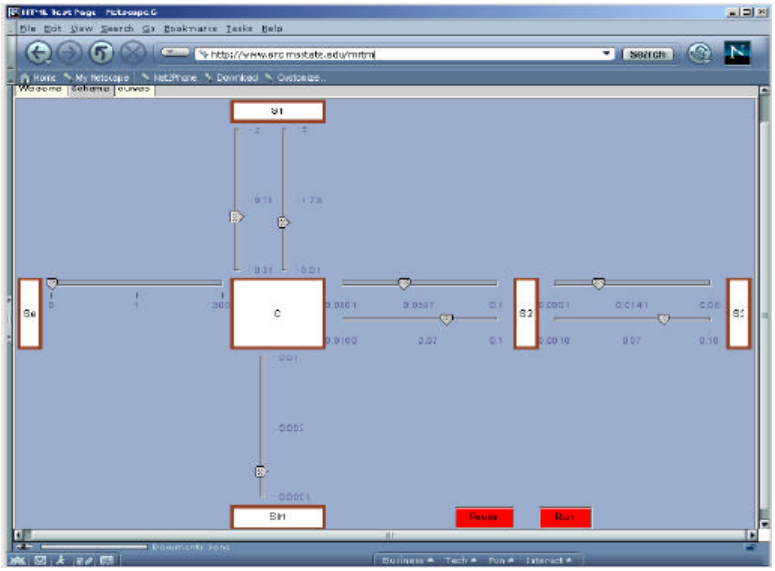

Figure 3.4: GUI for the one-dimensional simulation system.

calls the computational package to work, and lets the graphical package animate the reactions and plot result curves (Figures 3.4 and 3.5).

The user can specify the soil characteristics and transport and kinetic rate parameters using the sliders and input boxes (Figure 3.4). The computational simulation is initiated by clicking the control button. Each box in Figure 3.4 represents a particular solute phase in the MRTM model.

The simulation system generates soil concentration-depth curves, and allow users to choose specific time steps of the reactions in MRTM through the control buttons (Figure 3.5). All these features enable users of the simulation system to directly observe how the concentrations of different species respond to changes in various parameters, and identify trends and patterns that would otherwise be too time-consuming to discover. Text output files containing the numerical results are automatically generated in the current working directory of the local machine.

3.2.2 JNI and Applet Security

Legacy FORTRAN codes are embedded in this one-dimensional simulation system. They perform matrix computations, integration, and other numerical calculations. These code are

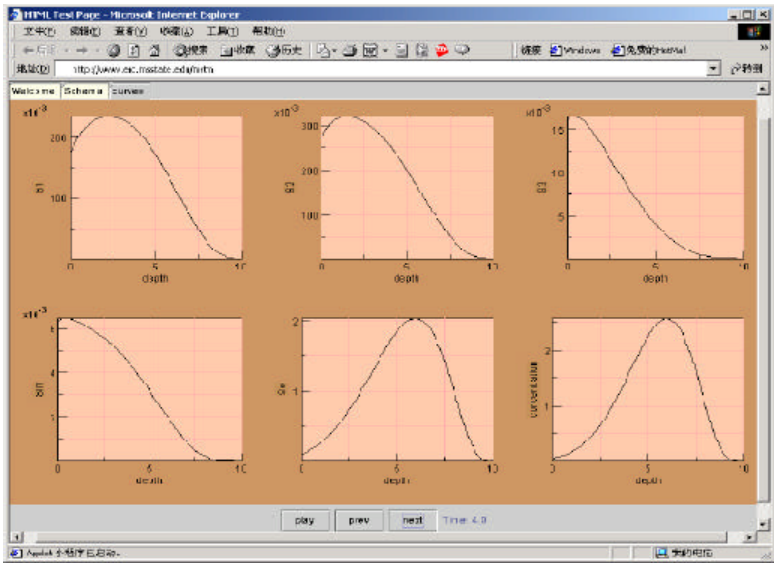

Figure 3.5: GUI for the one-dimensional simulation system.

kept intact instead of translated into Java, because FORTRAN language is still more efficient than Java and code transaction implies considerable labor cost. There are millions of lines of useful and tested FORTRAN codes today. The Java Native Interface (JNI) provides an efficient way for code reusability [40, 41].

The JNI is part of the Java development Kit (JDK). It allows Java code to call C/C++ code and provides all kinds of functionality for communications from both sides [40, 41]. The JNI is especially useful when the standard Java class library does not provide some needed platform-dependent features, when performance becomes the major concern or when existing legacy libraries written in other languages are too costly to rewrite.

Figure 3.6 shows the steps involved in the development of JNI programs. Once Java classes are defined with native methods they are compiled with the “Javac” command. Then, the utility program “Javah” generates header files for the native methods. Those native methods are implemented according to the declaration in the header files. The native code is then compiled to build a shared library (suffixed “.so” on Solaris platform) or a dynamic link library (suffixed “.dll” on Windows platform). The Java program loads the created library to be executed.

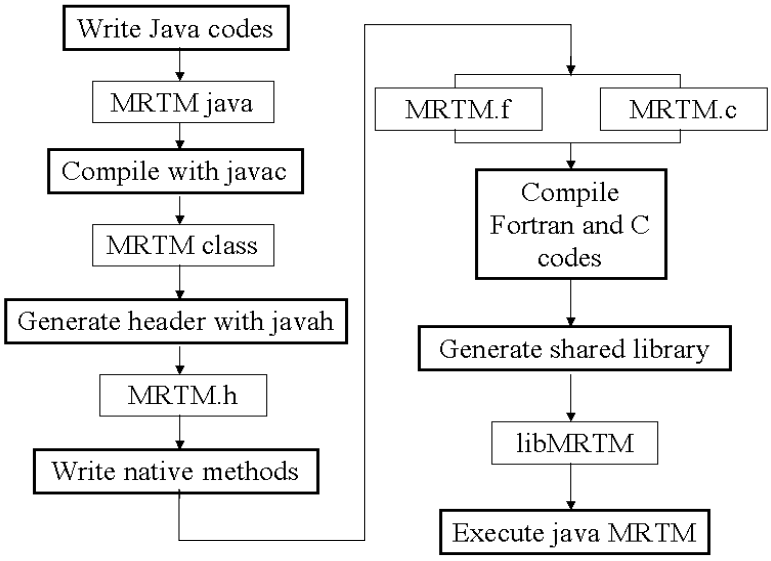

Figure 3.6: Flowchart of the combination of native methods with Java code.

The current version of JNI only permits the Java code to call C/C++ modules. In order to call FORTRAN or even assembly modules, building the shared (or dynamic) library must be extended using the C-FORTRAN interface [48]. When FORTRAN code needs to be called by Java using JNI, C and FORTRAN source code should be compiled separately to generate object module files suffixed with ".o". The linker utility then combines the C and FORTRAN object modules to build the desired library, which can be called by Java code [40].

There are two kinds of Java programs running in the client-side computers with different security restrictions. A Java application is a regular program running inside an abstract environment called Java Virtual Machine (JVM). A Java applet, on the other hand, runs inside the Internet browser that usually imposes strict network restrictions. Since applets need to be downloaded from the remote website to the client machine before execution, security problems arise. All standard Internet browsers implement security policies to ensure that applets do not compromise system security, because applets without security restrictions could make the client computers vulnerable to network attacks [49].

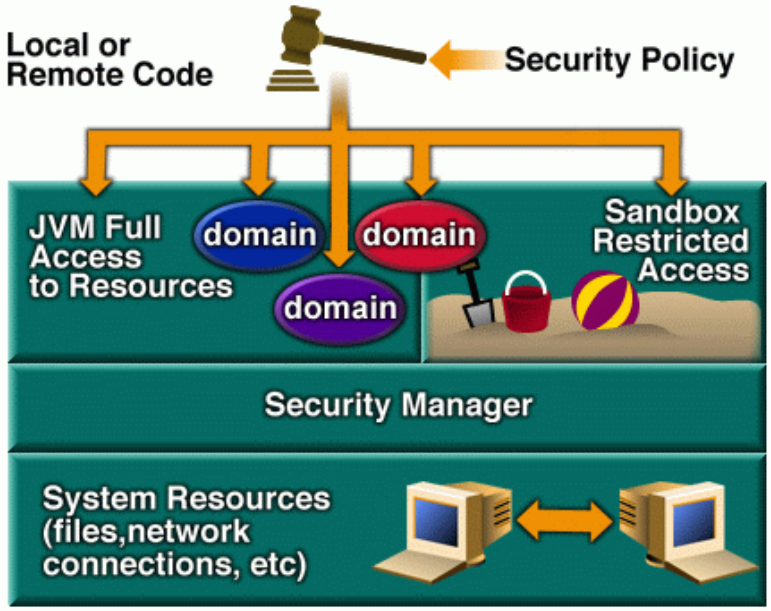

Figure 3.7: JDK1.2 security model [42].

The JDK 1.2 implemented the “sandbox” security model (Figure 3.7)[42, 44], in which default applications (local programs) have no limits within the system environment of the client machine. Default applets (remote programs) on the other hand, are treated as “untrusted” and restricted to the “sandbox” environment with limited privileges. Therefore, the applets cannot, e.g., load libraries, define native methods, read/write local files, read certain system properties, start any program on the client machine, etc. This research introduces a solution to the problem of applet’s restrictions to define native methods, by allowing those applets (with JNI native methods) to be downloaded to the client machine and executed. The Appletviewer utility provided by JDK 1.2 serves as the default browser due to its pure Java compatibility.

In order to grant further access to the applets using JNI, permissions are granted through a policy file using available features of the JDK 1.2. The security policy is set by the program user using the policy tool in JDK.1.2, which can create and modify the policy configuration files [42, 49]. In order for the applets to appear inside the browsers, the applet-class-loader checks the codebase property in the HTML file (that contains the Swing interface) to find where the Java classes are located, and then download those classes to the client machine.

When the JNI needs to load native libraries, it requires those libraries to reside on the local client machine. The Java applets, with granted permissions, download the libraries through HTTP connections. Then, the JNI loads the libraries through the system-class-loader instead of applet-class-loader. Therefore, the security policy needs to grant to the applets using JNI the following permissions: read file (libraries) from specified URL, write file (libraries) to specified local disk, and load libraries to call native methods. With all these permission, JNI-based applets can execute legacy libraries, and store computational result files on the client-side computers.

3.3 Three-dimensional Simulation System and Related Technologies

As mentioned before, the three-dimensional MRTM model is very computational intensive. The JNI technology, wrapping Java code with the legacy code and executing the computations on the client side, can not meet the requirement of such high computing work loads. It is unacceptable for users to wait online for results during the time-consuming computations. High performance (parallel) computing is also difficult to implement inside the client side web browsers. Furthermore, the web-based programs with JNI blending technology require users either to know how to configure security on their local computers, or trust Java applet providers completely. The one-dimensional system libraries running inside Java applets have the constrains of platform dependence too. All these restrictions lead developers to the server-side computing model to fulfill the requirements of high performance computations.

3.3.1 Traditional High Performance Computing Model

Traditional programs in high performance computational field simulations are often platform-dependent, standalone, and lack of embedded visualization tools. The platform-dependence requires users to recompile programs before running them on a new platform. The source code even needs some changes if the software packages have platform-dependent features. Standalone applications require users to keep a copy of compiled programs and install all necessary supporting software on their local computers. Parallel standalone programs make things more unaffordable, because parallel computing systems are often much more

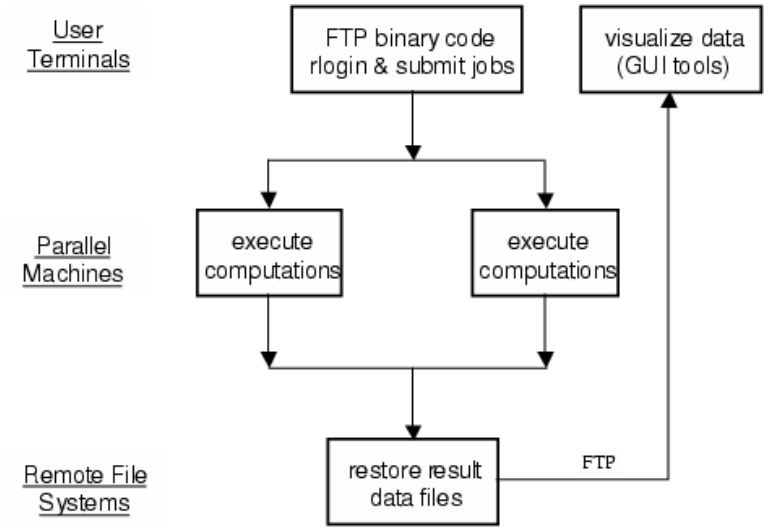

Figure 3.8: Traditional procedure for remote computation and data visualization.

expensive than personal computers. The traditional solution to utilize parallel resources is often to allocate accounts for different users, and permit them to logon the parallel system and then submit computational jobs. This solution adds more administrative loads to the parallel system, and apparently is not suitable for vast numbers of users. Furthermore, visualization of the computational results needs separate operations without the help of embedded visualization tools. The computational results are traditionally saved as data files after programs terminate. Although users can view the results with favorite visualization software, they must have the responsibility to install the visualization software locally and maintain their data files manually.

Figure 3.8 shows the traditional procedure to execute a high performance computational program and view the results afterwards. Authorized users transfer the computational-intensive binary codes to the remote supercomputers through FTP operations, submit the computational jobs through rlogin operations, and download the data results to their local computers through FTP operations later. Finally, users visualize the data results through GUI tools on their local computers. The whole procedure depends heavily on command-line-based operations.

Traditional simulation systems often interact directly with file systems, instead of database management systems, to search for the job information (i.e. result files). Since file systems

can't provide the convenience to efficiently manage and compare the information of different jobs, more additional command-line operations are needed. Furthermore, the traditional system developers should send the whole software package to users, even just for test purposes. The users then install and maintain the package along with necessary supporting software on their local machines. Such kind of software distribution is neither economical nor efficient for both developers and users.

3.3.2 Web-based High Performance Computing Model

With the development of the Internet, online access to remote high performance computing programs seems to become a natural extension to the traditional computational model. Through the popular web browsers, users could have the vision to take full advantages of the remote distributed resources, including supercomputers, visualization software, and storage systems (file systems and database systems). Users could get completely relieved from the tedious command-line-based operations with the back-end system complexity hidden from them. Web-based scientific computing systems, as applications of the so-called E-Science technologies, tend to provide the same convenient and powerful services as E-Commerce technologies do in business fields.

This research uses the Java 2 Enterprise Edition (J2EE) technologies [43, 50] by Sun Microsystems, and proposes a general architecture to build robust and scalable enterprise applications for web-based server-side high performance computing simulation systems (three-dimensional MRTM), as shown in Figure 3.2. The J2EE server, providing the running environment for the middle-ware programs, resides on a machine accessible through the Internet (i.e. Titan.erc.msstate.edu). GUI components controlling the simulation systems are embedded in the client-side web pages, which are displayed through the web browsers. The J2EE middle-ware programs transmit the job-managing requests (i.e. submitting computational jobs or visualizing data results) from the client-side web browsers to the server-side high performance computing resources, and feed back the responses by generating dynamical GUI web pages. The computational model developers build, validate, and test the computing modules separately, and

once for all install the binary code to the server-side supercomputers (Navajo.erc, Lakota.erc, and so on). The computational results are stored in the remote network file systems (NFS), and controlled by database management systems (DBMS) residing on a database server machine (i.e. Ra.msstate.edu). The visualization server machine (i.e. Andy.erc) performs remote visualization tasks utilizing the traditional graphics packages in batch mode.

The user-transparency provided by the web-based integration allows users to access the simulation system just through web browsers without caring about where the computations are performed. Since Java programs can "compile once, run everywhere", the J2EE middle-ware programs are platform-independent without portability problem in the traditional system integration. Since all codes (high performance computation, visualization, J2EE middle-ware and database SQL script) are developed, installed, and maintained on the server side by different developers, software development and distribution become more flexible. The integrated system administration can also be implemented as web services by the J2EE middle-ware. Therefore, not only the traditional rlogin and FTP operations can be relieved from users, but also the system maintenance operations (i.e. allocating user accounts, deleting obsolete data files) can be performed by administrators through the web interface. Furthermore, the J2EE server provides the low-level system security control on the server side, and simplifies the security logic during the code development. The integrated system architecture in Figure 3.2 takes full advantages of distributed resources with more efficiency than the traditional central management system architecture. In conclusion, the J2EE-based remote computing architecture guarantees the system portability, flexibility, security, and efficiency.

3.3.3 J2EE Technologies and Enterprise System Integration Architecture

The three-dimensional MRTM simulation system integration is based on the J2EE technologies, which provide the component-based approach to the design, development, assembly, and deployment of enterprise applications. The simulation system adopts the multi-tiered distributed application model, in which the whole system logic is divided into various components in different tiers running on different client-side or server-side computers. Thus,

people with different skills working on different components or tiers can collaborate during the whole development process. Therefore, the J2EE-based simulation system, as called the enterprise application, is very scalable and flexible.

Component-based system development is one of the major features of J2EE enterprise applications. This development paradigm is fundamentally built upon the object-oriented (OO) development paradigm. The principles of modularity, abstraction, and encapsulation inherent to the OO paradigm enable software reusability, scalability, maintainability, and rapid development. Compared to the OO software development paradigm, the component-based software development builds enterprise applications more rapidly and with a higher degree of reliability.

In the OO development paradigm, the fundamental software units are classes and objects. Classes encapsulate fine-grained concepts of a problem or solution, whereas objects represent instances of classes created during the runtime execution of the application program. A class only encapsulates a singular concept with a set of object attributes and operations. The OO class-level software reusability is not practical for the high-level development of enterprise applications, due to the big jump from logic analysis to program design.

On the contrary, the component-based development offers a coarser-grained means to analyze and encapsulate problems. A component represents a particular service as a function group of several concepts (or classes in the OO paradigm) in the component-based paradigm. The components expose service interfaces to developers and users. The interfaces encapsulate the component services and shield the implementation details. Representing a logical collection of one or more finer-grained classes at a higher and coarser-grained level, the component may directly encapsulate a partitioned problem and implement the application straightforward from logic analysis to program design. Different components are used in different tiers of the three-dimensional MRTM integrated system, including Applets in the client tier, JSPs and Servlets in the web tier, and EJBs and JavaBeans in the business tier. The component paradigm, as

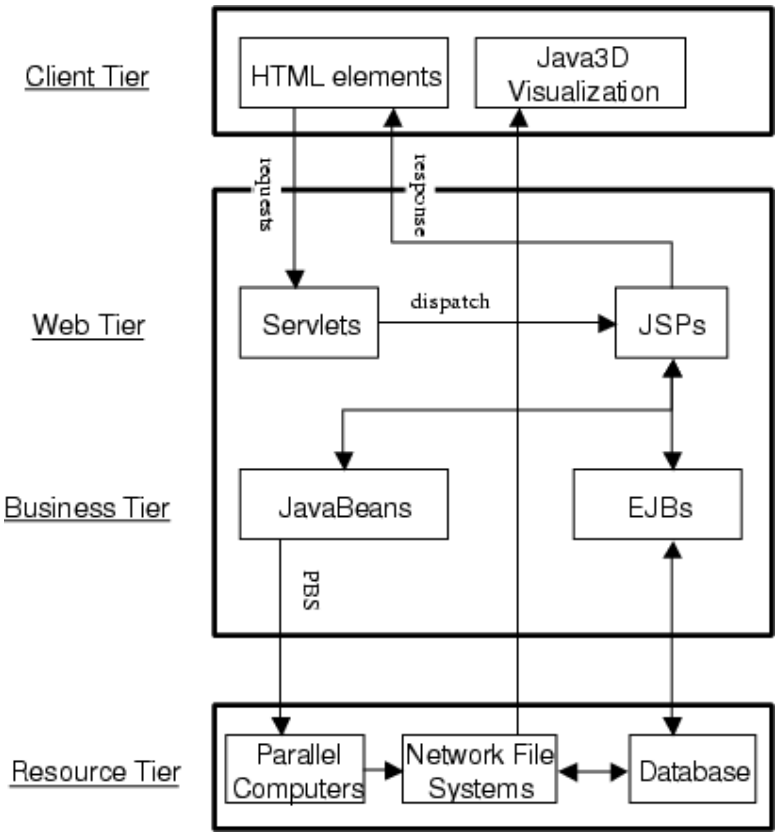

Figure 3.9: Enterprise system architecture for the three-dimensional MRTM simulation.

the foundation of J2EE technologies, greatly simplifies the enterprise application development [43, 50]. The multi-tiered system architecture is described in Figure 3.9.

3.3.3.1 Client Tier

As shown in the multi-tiered architecture (Figure 3.9), the client tier is the top layer of the three-dimensional MRTM simulation system. The components of the client tier run inside the web browsers on the client machines, or users' local computers. The client-tier users from web browsers access the enterprise application on the MRTM simulation system website, input parameters for the simulation (including transport and retention coefficients, even parallel computing configurations), and submit computing jobs through the web GUIs to parallel computers on the server side. The simulation system information, retrieved from the database through the underneath tiers, can also dynamically display on the web pages. Thus, users may

check their job information (view the computing results), and the simulation administrator may change the user information (add or delete users). All these presentations are implemented by components called the web clients, which are dynamic web pages written in various types of markup language (i.e. HTML and XML). The web browser transforms requests in the client tier into HTTP requests and sends them to the web tier, then receives HTTP responses from the web tier and transforms them into graphic user interface (GUI) contents to render in the web clients [51].

The web clients are often called thin-clients in the enterprise application, because they do not directly query databases, execute complex business logic, or connect to legacy applications. Such heavyweight operations are off-loaded to the components in the web tier or business tier underneath in order to utilize the security, speed, service, and reliability provided by J2EE server-side technologies. The thin-client design makes the enterprise application scalable, and development process simple and flexible.

Java applets, as J2EE client-side components, are embedded in the client tier serving as containers for visualization programs written in Java3D graphical packages [52]. The Java Swing technology combined with Java3D supports the visualization interface for the MRTM computational result data. As discussed in the one-dimensional MRTM simulation system, Java applets with security constrains execute in the Java Virtual Machine (JVM) of the web browser, hence requires users to configure permissions on their local computers. Such security restrictions from the client-side application model are relieved from users in this enterprise application, where the system permissions are configured through the J2EE server by the administrator.

3.3.3.2 Web Tier

As the second tier in the multi-tiered enterprise simulation system, the web tier communicates with both the upper client tier and lower business tier. Inside this tier, the information from the client-tier requests are processed and forwarded to the business tier. The dynamically generated responses by the web tier, together with the feedback from the business tier, are transmitted back to the client tier for displaying in the web pages. Web

components separate the web logic control from the web page design, enabling concise and modular development.

The logic of the web tier is implemented by the web-tier components running in the J2EE server. Java servlets and Java Server Pages (JSP) are the major components to generate dynamical responses for the client-tier requests. Servlets written as Java classes dynamically process HTTP requests and construct HTTP responses for web applications. Due to the object-oriented features of Java language, servlets have the advantages of maintainability and reusability, hence used as the central controlling units in the web tier of the three-dimensional MRTM enterprise simulation system. According to the client-tier requests, the controlling servlets forward the logic flow to different JSP pages, or the processing units.

JSP pages are the processing units in the web tier of the three-dimensional MRTM enterprise simulation system. Compared to the tedious pure-Java programming for the servlet development, the JSP technology allows a natural approach to creating web contents because of its simplicity, flexibility, and compatibility. JSP pages are text-based documents with two types of contents: the static scripts expressed in HTML (HyperText Markup Language) or XML (Extensible Markup Language) format, and the JSP elements, which construct dynamic responses. With the help of HTML or XML scripts, generating dynamical web responses according to client-side requests becomes very straightforward [53, 54]. The JSP elements contain the Java code which retrieves, processes and forwards information between the client and business tier. Custom Tags, an important JSP elements for example, realize the loop operations in dynamical web content generation in the MRTM simulation system [43, 54].

JavaBeans, another type of web-tier components, retrieve the input information from client-side HTTP requests, such as the user account and job management information. Since JavaBean components execute in the J2EE server, they can access the runtime Java Virtual Machine (JVM) of the server computers [50]. Compiled binary code written in native languages like C/C++ and FORTRAN can be called directly by the server-side runtime JVM through JavaBeans. Therefore, the client-side JNI technology used in the one-dimensional simulation system is replaced by

server-side JavaBean component technology. The binary code directly called by JavaBeans in the three-dimensional MRTM enterprise system is more efficient than the JNI wrapper code in the one-dimensional system.

3.3.3.3 Business Tier

In order to improve performance, business logic, such as processing user and job information, is relieved from the web tier and handled by the business tier. In the business tier, Enterprise JavaBean (EJB) components receive information from the web-tier components, process the format, and send to the database in the resource tier for storage. EJBs also retrieve data from the database in the resource tier, process them if necessary, and send back to the web tier for display.

In the MRTM enterprise system, two kinds of EJBs are used: session beans (session EJB) and entity beans (entity EJB). A session bean represents a transient conversation with the client side. When the client finishes executing (for example, the user logoff the simulation system), the session bean and the transient data are gone. In contrast, an entity bean represents persistent data stored in one row of a database table. If the client terminates or if the server shuts down, the underlying J2EE services ensure the entity bean data is saved in the database. The session EJBs retrieve information directly from the web tier, process the business logic, and then call the entity EJBs. Entity EJBs directly connect database in the resource tier to load or store the information processed by the session EJBs. Therefore, the system logic flows through servlets, JSPs, session EJBs, entity EJBs, and finally arrives the database in the resource tier.

The enterprise system can improve performance by dividing the business tier to session EJBs and entity EJBs. Since the J2EE server can only accommodate a limited number of concurrent database connections, excessive interactions between JSPs and entity EJBs (for example, when many users access the MRTM system simultaneously) can over-burden the server system resources. The session EJBs, acting as a buffer, process the business logic from the web tiers first, and call the entity EJBs to connect database only if necessary. Thus, the number of concurrent database connections are greatly reduced, hence system performance will improve.

This multi-tiered design also makes the J2EE application more scalable. The source code of servlets, JSPs, session EJBs, and entity EJBs in the enterprise system is clearly modulated and greatly simplified. Future functional enhancement in different tiers only cause minimum changes to the entire enterprise system.

3.3.3.4 Resource Tier

The resource tier in the J2EE-based MRTM simulation system contains the database server, parallel computing resources and network file systems. The database server, for example an Oracle server, can run on a separate machine. This enterprise system uses the Cloudscape database server embedded in the J2EE server to manage the database. Therefore, both the database server and J2EE server run on the same machine. The parallel computing resources, or the parallel computers, are eight SUN SuperMSPARC processors with the shared-memory environment. The network file systems store the parallel MRTM simulation code written in C language, and the computational result files.

Maintaining enterprise data, such as the system maintenance information (i.e. user account and authentication) and the computing result information (i.e. result file location), is one of the most important steps involved in building the three-dimensional MRTM enterprise simulation system. The collection of data, often referred to as a database, can be managed by a database management system (DBMS). DBMS is the information technology used by enterprise system to efficiently retrieve, update, and manage enterprise data. Java Database Connectivity architecture (JDBC) is the standard means for connecting database from Java applications [44, 55]. All the three-dimensional MRTM simulation information is stored and managed in a database through JDBC connections. The storage and management of scientific data through database management systems allows scientists to explore data resources across Internet efficiently.

In this enterprise system, entity EJBs in the business tier access the database through JDBC connections. JavaBeans in the web tier submit parallel computing jobs directly to the resource tier. The parallel computing job queues are managed by the Portable Batch System (PBS), which provides much flexibility and functionality. The PBS operates on multiple UNIX network

environments, including heterogeneous clusters of workstations, supercomputers, and other parallel systems. Users submit jobs to the PBS specifying the number and type of CPUs. The manual operations are automatized in the web-based enterprise MRTM simulation system. With specified email address in the PBS job submission, the job execution acknowledgment can be sent to users synchronously.

Although the parallel computers and J2EE server reside in different locations, they access the same network file systems in the resource tier. The result data files of the MRTM simulation system are saved by the parallel computers to the public_html directory of the J2EE server, and directly read and rendered by the Java3D applets in the client tier. URL connections help Java3D applets read data from the resource tier. Policy files for security configuration used in the one-dimensional simulation system are not needed in the three-dimensional enterprise simulation. The J2EE administrator, instead of the system users, configures the access authority of the public_html directory, so as to solve the security restrictions for the applets in the client tier. Since the system logic is performed transparently to users in the J2EE server, this server-side security configuration makes the enterprise system more transparent, flexible, and secure.

The one-dimensional MRTM system uses Java2D technology, whereas this enterprise simulation system uses more powerful Java3D technology. Java2D and Java3D are the graphics and multimedia components for J2EE user interfacing. Java2D enables sophisticated two-dimensional rendering of objects and images. Java3D provides interfaces for creating and manipulating three-dimensional shapes, as well as adding animation, texture, lighting, and shading to these objects [47, 52]. These graphics technologies, together with database management, automatize the web-based 3D visualization of computing results in the enterprise simulation system.

3.3.4 System Integration and Web Interface

In conclusion, the three-dimensional MRTM simulation system applies the J2EE technologies to build the multi-tiered distributed computing enterprise architecture. The client tier provides the web-based GUI components for users to transparently access the back-end

Figure 3.10: GUI for the enterprise MRTM simulation system.

resources (i.e. specifying the parallel computers), submit computational jobs, and visualize the result data with Java3D Applets. The J2EE middle-ware programs run inside the middle tier, which is divided into the web tier and the business tier. The web-tier components, including Java Servlets and Java Server Pages (JSP), accept user requests and generate dynamic GUI responses for the client tier. The Enterprise JavaBean (EJB) components in the business tier interact with the database system in the resource tier, and manage the remote computational and system maintenance information. The JavaBean components in the business tier interact with the portable batch system (PBS), and automatize the batch-mode job submission to remote supercomputers in the resource tier. Since the governing convection-dispersion-reaction equations for the three-dimensional MRTM model are computationally intensive, parallel alternating direction implicit (ADI) method is applied for efficient solutions. The OpenMP parallel implementations for different boundary conditions are installed on the resource-tier supercomputers, serving as remote high performance computing services. The computing result data are stored in the network file system (NFS), while the computational job information and system maintenance information is managed by the database system in the resource tier.

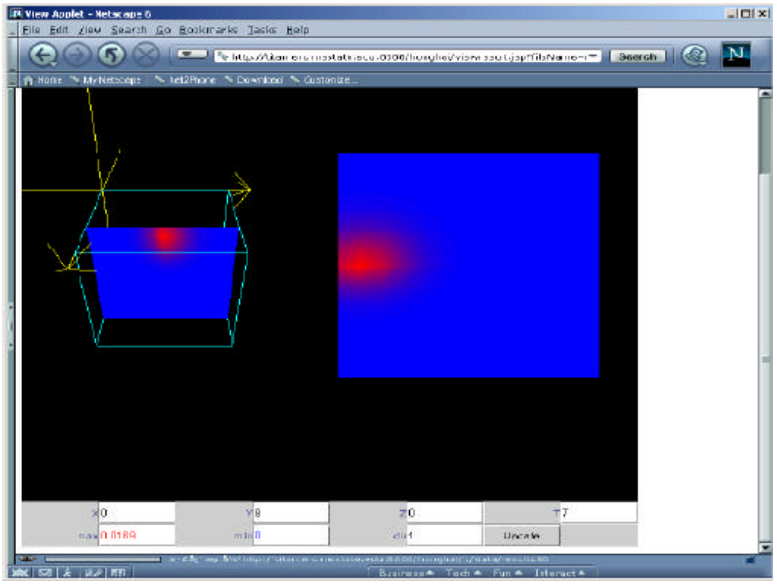

Figure 3.11: GUI for the enterprise MRTM simulation system.

As shown in Figure 3.10, the computational job information stored in database tables, analogous to Excel spreadsheets, is displayed in the web pages dynamically generated by J2EE middle-ware programs. Protected by password authentication, such web-based database management makes computational information easily accessible to scientists. The web GUIs contain links to the visualization page (Figure 3.11), where Java3D applets provide online controls to the 3D result data. Since the three-dimensional MRTM computing result files are huge in size, the Java3D applets visualize the solution of concentrates by cutting-planes in order to minimize the data transfer through the Internet. In addition, job submission page allows users to specify both the name of parallel computers and number of parallel threads. Users are automatically notified through emails when the remote computations start and finish. Other maintenance pages also provide online system administration functionality, such as allocating users and passwords, deleting obsolete job files, and so on.

In general, users of this J2EE-based simulation system feel just like shopping online when they submit computing jobs and view the visualization results. Such system integration can overcome the limitations of the previous JNI-based client-side computing architecture, and reuse

the platform-independent Swing GUI components in the one-dimensional MRTM simulation system.

As the E-Commerce technologies revolutionize the way people do business, E-Science is changing the way scientists do computing. Since the system logic is divided into various components running on different machines, developers with different skills can collaborate easily during the integration process. The web-based system integration provides the possibility to manage and utilize distributed computing resources more efficiently and economically. Scientists can also real-time trace remote high performance computations, easily manage and explore data resources across the Internet, and conveniently share and utilize remote software tools.

CHAPTER IV

PARALLEL IMPLEMENTATION

Parallel computing is widely used to efficiently solve scientific problems, which are often irregular, large and computationally intensive [56, 57, 58]. In this web-based three-dimensional MRTM simulation system, parallelization plays an important role to provide remote high performance computing services. The intensive computations are partitioned to parallel computers and solved simultaneously, with the results gathered and synchronized after all the processors finish their executions. Finding the best solutions represents an important contribution to the development and understanding of scientific phenomena. Based on the ADI method discussed in Chapter II, this chapter addresses the parallel computing concepts and focuses on the parallel technologies used in the MRTM simulation system. Parallelism is exploited at three different levels: the parallel algorithm, the system architecture, and the programming tool [56, 59].

4.1 Parallel Algorithm and Performance

Efficient parallel implementations are important to scientific problems. The major goal of the parallel implementations is to solve computing problems more quickly and efficiently with available parallel system resources. Many metrics are used to evaluate the performance of parallel algorithms, such as parallel/serial run time, speedup, efficiency, and cost [60, 61].

The serial run time (T_s) of a program is the time elapsed between the beginning and the end of its execution on a sequential computer. The parallel run time (T_p) is the time elapsed from the moment that a parallel computation starts to the moment that the last processor finishes execution. Speedup (S) is formally defined as the ratio of the serial run time of the best sequential algorithm for solving a problem, to the time taken by the parallel algorithm to solve the same problem on P processors. Efficiency (E), defined as the ratio of speedup to the number of

processors, reflects whether or not processors are usefully employed. The cost (C) of solving a problem on a parallel system, defined as the product of parallel run time and the number of processors used, reflects the sum of the time that each processor spends to solve the problem [32, 60, 61].

$$S = \frac{T_s}{T_p} \tag{4.1}$$

$$E = \frac{S}{P} \tag{4.2}$$

$$C = T_p \times P \tag{4.3}$$

A parallel system is cost-optimal if the cost of solving a problem on a parallel computer is proportional to the execution time of the fastest-known sequential algorithm on a single processor. The scalability of a parallel system, a measure of its capacity to increase speedup in proportion to the number of processors, reflects the parallel system's ability to effectively utilize increasing processing resources [56, 60].

Performance is the major concern in the parallel implementation. In an ideal system, speedup is equal to the number of processors P and efficiency is equal to one. However, in practice the speedup is less than P and the efficiency is between zero and one, depending on the degree of effectiveness with which processors are utilized. The speedup bound for a fixed-size problem can be expressed by the Amdahl's law [32, 62]. Assuming that a given problem needs N operations, each of which takes time τ to finish, the serial run time on one processor is

$$T_s = N\tau \tag{4.4}$$

Supposing a fraction $\alpha(0 \leq \alpha \leq 1)$ of the N operations are parallelizable, the parallel run time on P processors for that problem is

$$T_p = (1-\alpha)N\tau + \frac{\alpha N\tau}{P} \tag{4.5}$$

and the speedup is

$$S = \frac{T_s}{T_p} = \frac{N\tau}{(1-\alpha)N\tau + \frac{\alpha N\tau}{P}} = \frac{1}{(1-\alpha) + \frac{\alpha}{P}} = \frac{P}{(1-\alpha)P + \alpha} \tag{4.6}$$

Therefore, the speedup is bounded by $\frac{1}{1-\alpha}$, which is the highest possible value when $p \to \infty$. This disappointing prediction is based on the assumption that the problem size is fixed while the processor number increases. Actually, the goal of parallel computing is to solve large scale problems. An alternative model, based on fixed computation time instead of problem size, was proposed by Gustafson [32, 63]. Supposing that a given problem can be solved in one unit of time on a parallel machine with P processors and that the times in serial and parallel computations are $1 - f$ and f respectively, the time for a uni-processor computer to solve the same problem would be $1 - f + fP$. Therefore, the speedup is expressed as

$$S = \frac{1 - f + fP}{1} = 1 - f + fP \tag{4.7}$$

The Gustafson's model shows a linear relationship between the speedup and number of processors, and the possibility to achieve scalability of a parallel system [32]. In order to achieve good performance, the parallel system overhead, which is defined as the difference between its cost and the serial run time of the fastest known algorithm for solving the same problem, should be minimized. The definition of overhead encapsulates all the causes of the inefficiencies of a parallel system, whether due to the algorithm, the architecture, or the algorithm-architecture interaction. The major sources of overhead in a parallel system are inter-processor communication, load imbalance, and redundant computations [56, 57, 64]. The time to transfer data between processors is usually the most significant source of parallel processing overhead. Load imbalance occurs when different processors have different work loads, because some processors may be idle while others are busy working on the problem. The fastest known sequential algorithm for a problem may be difficult or impossible to parallelize, forcing people to use a parallel algorithm based on a poorer but easily parallelizable sequential algorithm. The

extra computations from the slower sequential algorithm may also contribute to the overhead [64].

At the parallel algorithm level, the detection for parallel resources and overhead of the approximate factorization ADI method, which is used to solve the governing equations in the MRTM model, is an important task in this research. Considering the following three sub-steps in the approximate factorization ADI method for each time step:

$$
\begin{aligned}
&(I + \frac{\Delta td}{2}\delta_x - \frac{\Delta ta}{2}\delta_x^2)u^*_{i,j,k} \\
&\quad = (I - \frac{\Delta td}{2}\delta_x + \frac{\Delta ta}{2}\delta_x^2)(I - \frac{\Delta te}{2}\delta_y + \frac{\Delta tb}{2}\delta_y^2)(I - \frac{\Delta tf}{2}\delta_z + \frac{\Delta tc}{2}\delta_z^2)u^n_{i,j,k} \\
&\quad + \frac{F^{n+1}_{i,j,k} + F^n_{i,j,k}}{2}\Delta t \qquad (4.8)
\end{aligned}
$$

$$
(I + \frac{\Delta te}{2}\delta_y - \frac{\Delta tb}{2}\delta_y^2)u^{**}_{i,j,k} = u^*_{i,j,k} \qquad (4.9)
$$

$$
(I + \frac{\Delta tf}{2}\delta_z - \frac{\Delta tc}{2}\delta_z^2)u^{n+1}_{i,j,k} = u^{**}_{i,j,k} \qquad (4.10)
$$

In each sub-step, the equations should be assembled along either x, y, or z directions depending on the nature of the spatial operators. For example, in the first step, the spatial operators δ_x and δ_x^2 act on the x direction, so all equations along that direction in Equation 4.8, corresponding to different i-index values, $i = 1, \cdots, NI + 1$, should be assembled to form a system of equations with a tri-diagonal coefficient matrix. Similarly, equations corresponding to different j-index values, $j = 1, \cdots, NJ + 1$, in Equation 4.9 and different k-index values, $k = 1, \cdots, NK + 1$, in Equation 4.10 should be assembled. This leads to three sets of algebraic equations:

$$
\mathbf{A}\mathbf{c}^*_{j,k} = \mathbf{H}\mathbf{c}^n_{j,k} + \frac{\Delta t}{2}(\mathbf{f}^{n+1}_{j,k} + \mathbf{f}^n_{j,k}),\ j = 1, \cdots, NJ + 1,\ k = 1, \cdots, NK + 1, \qquad (4.11)
$$

$$
\tilde{\mathbf{A}}\mathbf{c}^{**}_{i,k} = \mathbf{c}^*_{i,k}, \quad i = 1, \cdots, NI + 1, \quad k = 1, \cdots, NK + 1, \qquad (4.12)
$$

$$
\bar{\mathbf{A}}\mathbf{c}^{n+1}_{i,j} = \mathbf{c}^{**}_{j,k}, \quad i = 1, \cdots, NI + 1, \quad j = 1, \cdots, NJ + 1. \qquad (4.13)
$$

The computations to solve these systems of equations within each substep are independent to each other. For example, there are $(NJ+1) \times (NK+1)$ systems of equations in Equation 4.11 that can be solved concurrently by different processors. Similarly, the $(NI+1) \times (NK+1)$ equations in Equation 4.12 can be solved concurrently once all $\mathbf{c}^*_{j,k}$ has been calculated. This makes parallelization of the algorithm fairly straightforward, particularly in a shared memory programming memory environment [32, 65]. The following is the structure of the parallel code:

0. Data input and initialization

1. For time step $n = 1, \cdots, N$

  2. Solve $(NJ+1) \times (NK+1)$ systems of equations in Eq. (4.11) concurrently

  3. Solve $(NI+1) \times (NK+1)$ systems of equations in Eq. (4.12) concurrently

  4. Solve $(NI+1) \times (NJ+1)$ systems of equations in Eq. (4.13) concurrently

5. End time stepping loop

6. Output

The straightforward parallel implementation fully utilizes the parallel resources from the approximate factorization ADI method, by distributing systems of equations evenly for processors to solve concurrently. Since solving each equation in the systems 4.11 to 4.13 consumes nearly the same amount of computational time, this parallel implementation is load-balanced. The major concern to improve the performance lies in minimizing the communications among processors. Most communications happen when the computational results get synchronized after each time step. Sections 4.2 and 4.3 address how to minimize these communications.

4.2 Mapping Parallel Algorithms to Architectures

An successful parallel implementation should not only choose the optimal parallel algorithm, but also map the algorithm to proper parallel architecture [66, 67]. After exploiting the parallel

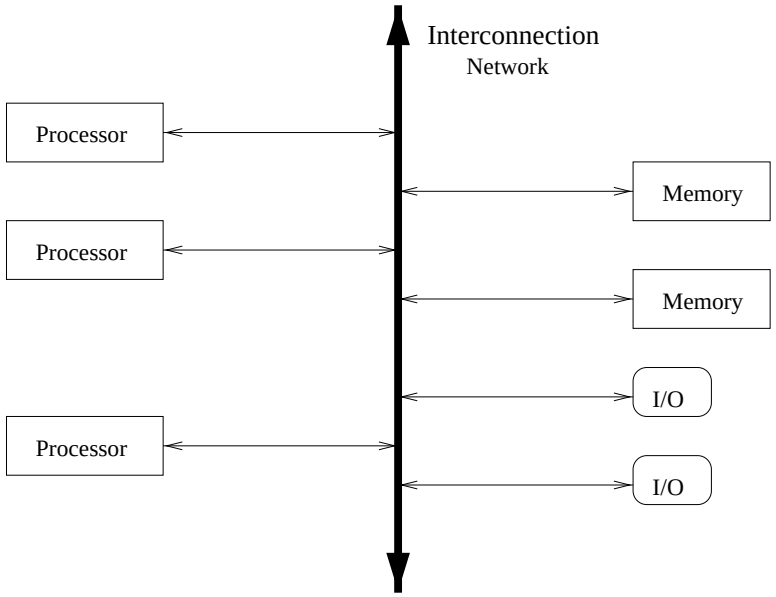

Figure 4.1: Shared memory environment.

resources from computational intensive ADI method at the parallel algorithm level, this section discusses the characteristics of parallel system and maps the parallel algorithm to parallel environment at the system architecture level.

There are two different kinds of parallel computing architectures according to the processor-memory architecture: the shared memory environment and the distributed memory environment. In the shared memory programming environment (Figure 4.1), programmers treat the parallel program as a collection of processes accessing a central pool of shared variables. There is no need to transfer data among processors. In the distributed memory or message-passing programming environment (Figure 4.2), programmers treat the parallel program as a collection of processes with private local variables, and manage the data transmission among processes by passing messages. Each processor stores its local data copy, and sends or receives data from other processors.

Different parallel algorithms adapt to different parallel computing environments. In order to obtain a high performance solution, the straightforward parallel ADI implementation should apply its sufficient exploitable parallelism to suitable system architecture. Although the parallel

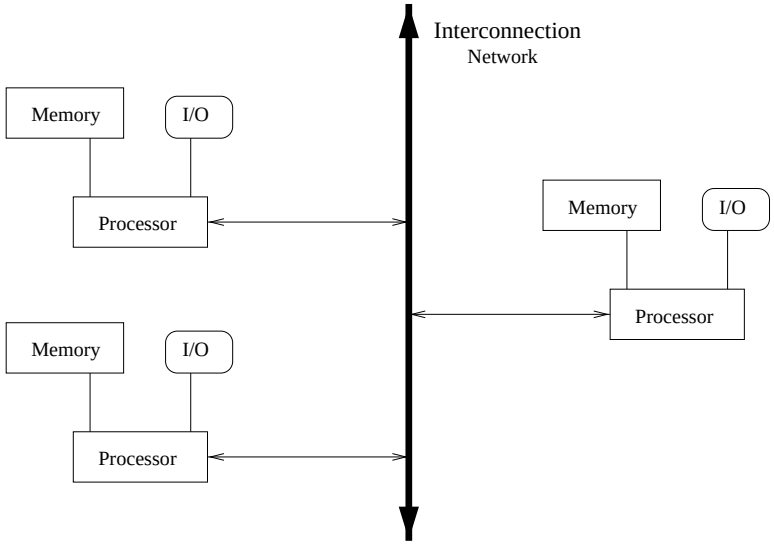

Figure 4.2: Distributed memory environment.

ADI method contains enough parallelism from the independent loops inside the algorithm, it requires each processor to access global data in the computational domain at each sub-step, due to the direction alternation during each iteration (time step).

For example in Figures 4.3 to 4.5, the computational domain is divided evenly to two processors perpendicular to the X direction. For the sub-steps while computing directions are parallel to the cutting plane (along X direction in Figure 4.3 and Y directions in Figure 4.4), processors only need few updated boundary data near the cutting plane from each other. However, when the computations switch to the X direction at the third sub-step, complete data exchanges among processors are necessary as shown in Figure 4.5.

Applied in distributed memory environments, the straightforward parallel ADI implementation should distribute and gather global data at each sub-step. The corresponding communication overhead could dominate the computation costs. Since all the processors in the shared memory environment access the global data from the central memory pool, no data exchange is needed among processors at each sub-step. Therefore, the communication overhead caused by distributing and gathering global information is minimized in the shared memory environment.

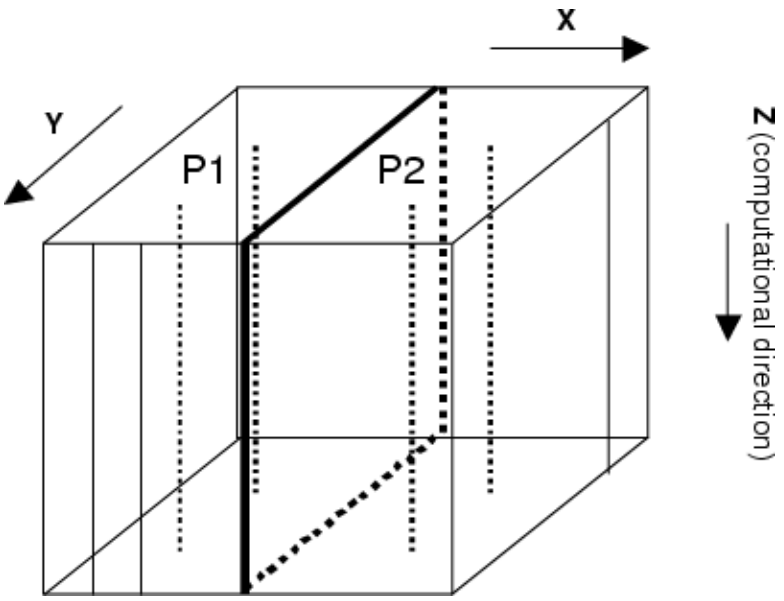

Figure 4.3: Data distribution for the straightforward parallel ADI implementation: computational direction Z is parallel to the domain decomposition.

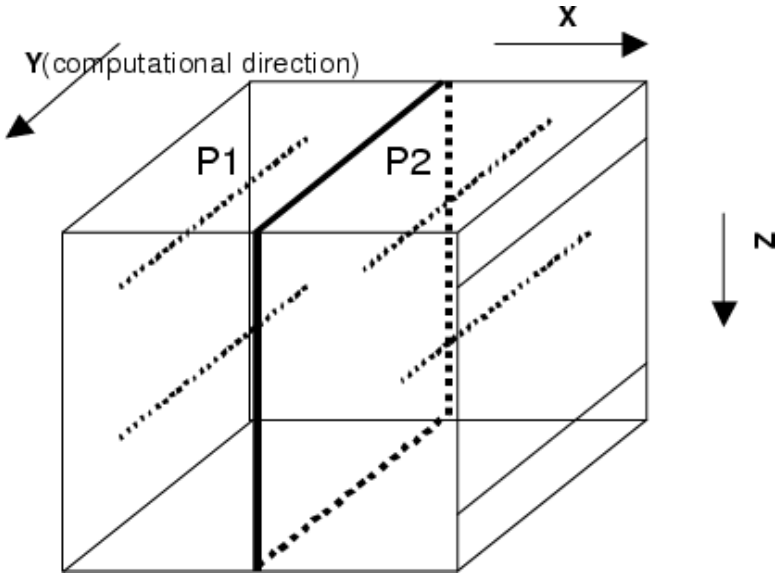

Figure 4.4: Data distribution for the straightforward parallel ADI implementation: computational direction Y is parallel to the domain decomposition.

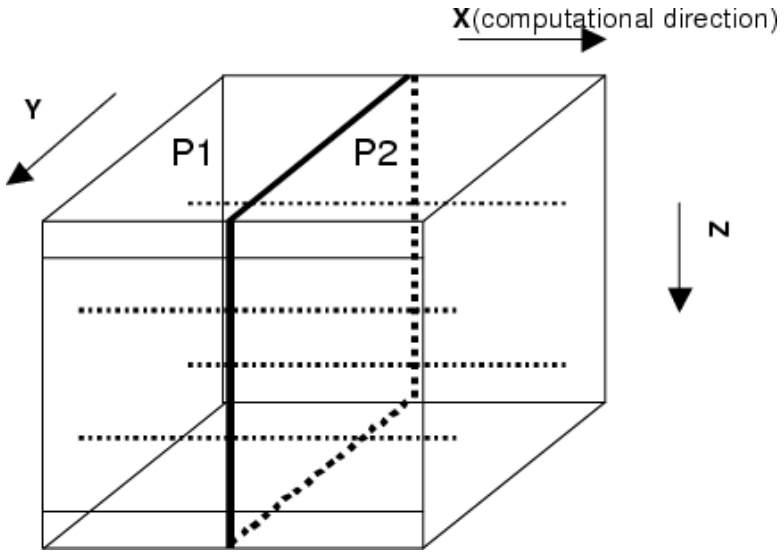

Figure 4.5: Data distribution for the straightforward parallel ADI implementation: computational direction X crosses the domain decomposition.

In conclusion, the shared memory environment is the optimal option for the straightforward ADI parallel implementation. In the shared memory environment, the same parallel implementation could achieve much more performance gain than in the distributed memory environment. In this research, the characteristics of the parallel algorithm determine the choice of parallel architecture. The straightforward parallel ADI algorithm is chosen first, hence the shared memory parallel environment.

4.3 OpenMP Parallelization on Shared Memory Environments

After exploiting the parallelism at the algorithm and architecture levels, developers need to choose proper parallel programming tools. OpenMP and Message Passing Interface (MPI) are two popular parallel tools with distinct characteristics.

OpenMP, an industry standard API for shared memory programming, is based on multiple threads, using the fork-join model (Figure 4.6) of parallel execution in the shared memory environments [20]. A shared memory process consists of multiple threads. An OpenMP program starts running as a single thread: the master thread. The master thread executes sequentially until the first parallel region construct is encountered. The parallel region contains the parallel

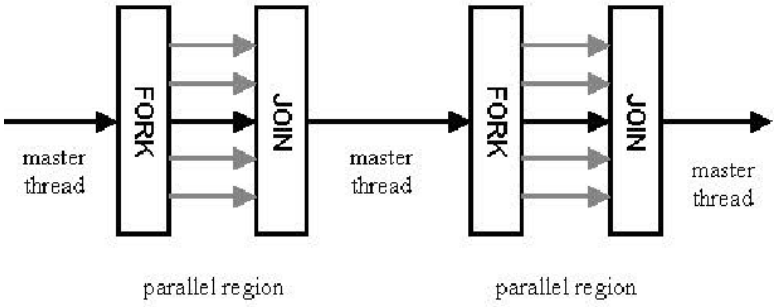

Figure 4.6: Fork-join model in OpenMP.

resources, such as independent loops in the program. The master thread then creates or "fork" a team of parallel threads, which run concurrently in the parallel region with the program code duplicated. After the whole team of parallel threads complete their execution, they synchronize and terminate, or "join" together, leaving only the master thread still running. This fork-join procedure continues until all the computations finish. OpenMP is favored by developers who need to quickly parallelize existing scientific code. The data structures in the parallel programs do not need to be partitioned, and all the processors can access the central pool of shared variables.

MPI is a portable, widely available, and accepted standard primarily for the distributed memory environment, in which programmers view their programs as a collection of processes with private local variables and the ability to send and receive data between processes by passing messages [68]. Processors use their local variables, and send or receive data among each other. MPI requires the program data structures to be explicitly partitioned, so that the entire application must be parallelized to work with the partitioned data structures. MPI is also supported on the shared memory environment too. The following comparison explains why OpenMP is preferred rather than MPI in this research.

The OpenMP parallelism is mainly specified through the use of compiler directives, which are imbedded in the C/C++ or FORTRAN source code. This feature, not provided by MPI, supports incremental parallelization of an existing sequential program, and makes parallelization quick and easy. Instead, developers have to rewrite the sequential code completely in order to

Table 4.1: OpenMP vs. MPI.

OpenMP	MPI
Compiler directives	Functions or subroutines
Incremental parallelization	Code rewrite completely
Simple	Complicated
Thread number changeable	Process number set
Flexible	Not flexible
SPMD and shared variables	only SPMD mode
Shared memory environment	Distributed memory environment

parallelize them using MPI subroutines. OpenMP can dynamically alter the number of threads which may be used to execute different parallel regions. This feature helps utilize system resources more efficiently. MPI could not support this feature, and requires the number of parallel processors set constant before running the program. In OpenMP, both single program multiple data (SPMD) and data sharing are supported. The parallel regions permit the thread team to execute the same program code with duplicated data, while controlling whether or not some special data is shared or private. However, current version of MPI primarily supports SPMD, in which data is explicitly communicated without sharing. Although MPI programs can also run in shared memory environments, they waste more memory storage than OpenMP programs as a general rule [68, 69].

The flexibility and simplicity of OpenMP compared to MPI can be shown from the simple programs to compute the sum of an array. The OpenMP parallel code just needs to add a compiler directive before the parallel region, without changing the data structure or flow control in the sequential code. Programmers only need to find the independent loops, or parallel regions, inside the sequential code. This “incremental” parallelization allows programmers to specify the shared and private variables in the parallel region together with global reduction operations. Therefore, the incremental OpenMP parallelization is desirable especially for the legacy code, written in C or FORTRAN, to be simply and quickly parallelized. On the contrary, the MPI parallel codes require data structures to be partitioned explicitly. Since the program flow is controlled

```
/* sequential program to compute array sum */
#include <stdio.h>

void main() {

  int i, n;
  float *array, sum;

  printf("please input the array size:");
  scanf("%d", &n);

  array = (float *)malloc(sizeof(float)*n);

  printf("please input the whole array:");
  for (i=0; i<n; i++)
    scanf("%f", &array[i]);

  sum = 0.0;
  for (i=0; i<n; i++)
    sum += array[i];

  printf("the summation is: %f", sum);
}
```

Figure 4.7: Sequential program to compute array sum

explicitly for different processors, the whole logic becomes much more complicated than the sequential code. Local variables are passed among processors so as to realize data sharing. The data distribution and gathering are implemented manually, rather than automatically in the OpenMP parallelization. Therefore, programmers often need to re-design the whole sequential code during the parallelization.

The properties of a problem, including the parallel algorithm to solve the problem, are the determining factors to choose the parallel tools. In this research, OpenMP is a better choice than MPI to parallelize the approximate factorization ADI method. Since the straightforward parallel ADI algorithm needs global data at each sub-step, the shared memory environment is chosen accordingly to realize the parallel implementation. Due to its simplicity and flexibility,

```
/* OpenMP paralle program to compute array sum */
#include <stdio.h>

void main() {

  int i, n;
  float *array, sum;

  printf("please input the array size:");
  scanf("%d", &n);

  array = (float *)malloc(sizeof(float)*n);

  printf("please input the whole array:");
  for (i=0; i<n; i++)
    scanf("%f", &array[i]);

  sum = 0.0;

#pragma omp parallel for private(i) \
 shared(n, array) reduction(+: sum)
  for (i=0; i<n; i++)
    sum += array[i];

  printf("the summation is: %f", sum);
}
```

Figure 4.8: OpenMP parallel program to compute array sum

OpenMP is the best choice to implement the straightforward parallel ADI algorithm in the shared memory environment. The incremental feature of OpenMP allows developers to parallelize the sequential programs available with the least amount of changes. The ADI method contains plenty of independent nested loops, which are the parallel regions for the programmers to apply OpenMP directives without changing data structures or flow controls in the sequential code.

Because of the popularity of distributed memory computing environment [68], parallelization using MPI with relatively complicated (not straightforward) but still efficient ADI parallel algorithm [32] remains an optional topic in the future research. Actually, OpenMP can even work together with MPI, because current parallel computing environments are often

```
/* MPI parallel program to compute array sum */
#include <stdio.h>
#include "mpi.h"

void main(int argc, char *argv[]){

  int i, n, rank, nprocs;
  float *array, *tmparray, sum, tmpsum;

  MPI_Init(&argc,&argv);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  MPI_Comm_size(MPI_COMM_WORLD,&nprocs);

  if(rank==0){
    printf("please input the array size:");
    scanf("%d", &n);

    array = (float *)malloc(sizeof(float)*n);

    printf("please input the whole array:");
    for (i=0; i<n; i++)
    scanf("%f", &array[i]);
  }

  tmparray = (float *)malloc(sizeof(float)*n/nprocs);

  MPI_Bcast(&n,1,MPI_INT,0,MPI_COMM_WORLD);
  MPI_Scatter(array,n/nprocs,MPI_FLOAT,tmparray,n/nprocs,MPI_FLOAT,
     0, MPI_COMM_WORLD);

  tmpsum = 0.0;
  for (i=0; i<n/nprocs; i++)
    tmpsum += tmparray[i];

  MPI_Reduce(&tmpsum,&sum,1,MPI_FLOAT,MPI_SUM,0,MPI_COMM_WORLD);

  if(rank==0){
    printf("the summation is: %f", sum);
  }

  MPI_Finalize();
}
```

Figure 4.9: MPI parallel program to compute array sum

composed of distributed computing nodes, each of which contains multiple processors with shared memory architecture. In conclusion, OpenMP is favored in this research due to its simplicity, flexibility, and the problem property of straightforward parallel approximate factorization ADI method.

4.4 Performance Experiments

Numerical experiments are taken to test the performance of the straightforward parallel ADI implementation in the shared memory environment. The key in the OpenMP parallelization is to identify the most computational intensive and independent loops, and determine shared or private variables in the sequential code. To evaluate the scalability of the parallel code, we calculate speedups by dividing the execution time for solving a given problem on one processor by the execution times on multiple processors.

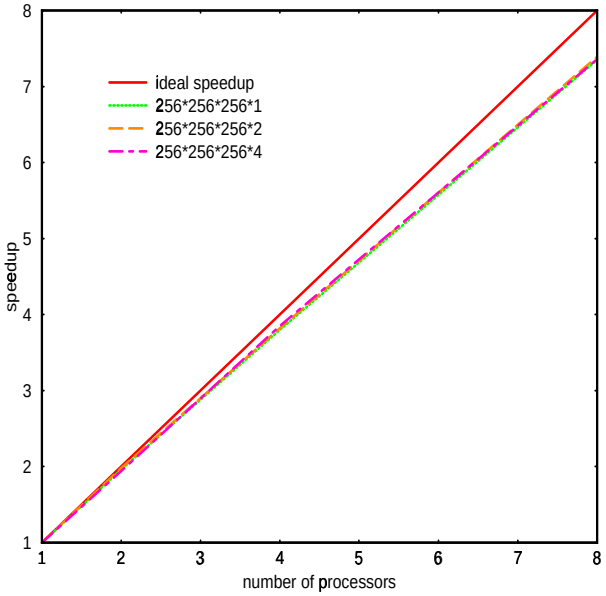

Figure 4.10: Speedup curves using a fixed grid of $256 \times 256 \times 256$ with different number of time steps 1, 2, and 4. It is clear that increasing the number of time steps does not affect the speedup.

Figure 4.10 shows the speedup curves obtained using an SGI Origin2000 parallel computer with 8 processors. The grid size is $256 \times 256 \times 256$. Three cases are reported using 1, 2, and 4

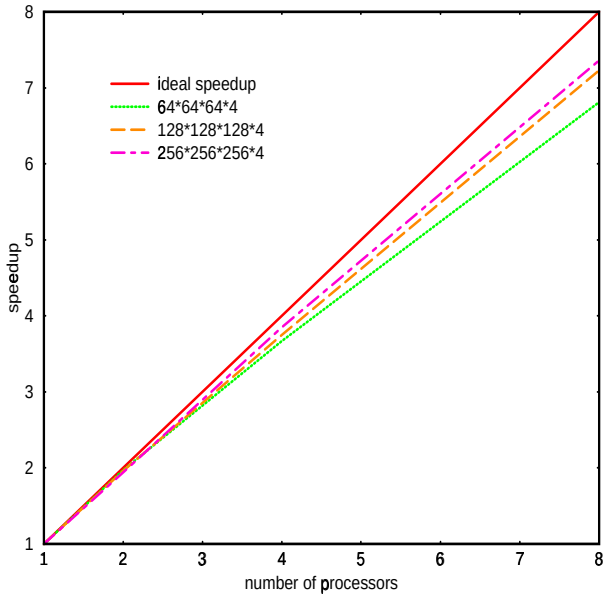

Figure 4.11: Speedup curves using different grids with the same number of time steps. It is clear that increasing the size of the grid also increase the speedup.

time steps respectively. It is clear from the figure that running more time steps does not change the speedup curve. This is not surprising since the parallelization was done within a time step. The measured speedup is very close to the ideal speedup, which indicates that the algorithm discussed here is very scalable.

Figure 4.11 shows the speedup curves obtained on the same computer using different grid sizes, i.e. $64 \times 64 \times 64$, $128 \times 128 \times 128$, and $256 \times 256 \times 256$. Four time steps are used in all cases. It is clear from the figure that the speedup improves as the grid size increases. This is due to the increased computation/communication ratio with increased grid size.

CHAPTER V

RESULTS AND ANALYSIS

The web-based MRTM simulations provide scientists an alternative way to trace contaminant movement in soils outside laboratories. The numerical solution experiments and parallel performance experiments were taken and analyzed in Chapter II and IV. This chapter analyzes several transport and retention scenarios using web interfaces, and addresses the advantages of web-based system integration. The JNI-based client-side one-dimensional MRTM simulation system is demonstrated first, and then the J2EE-based three-dimensional MRTM enterprise simulation.

5.1 One-dimensional MRTM Experiments

To demonstrate the JNI-based web interface on a hypothetical one-dimensional soil contamination problem, the following scenario was analyzed. A contaminant-free soil layer of 25 cm depth, having a volumetric moisture content of 0.4 cm^3/cm^3 and 1.25 g/cm^3 bulk density, received a load of a diluted contaminant X during 20 consecutive hours. The dispersion coefficient of water in the soil was estimated to be 1 cm^2/hr.

The compound-soil interaction was investigated in the laboratory [4, 5] revealing the following set of parameters: k_d = 1.0 cm^3/g (distribution coefficient), NEQ = 0.75 (Freundlich parameter), k_1 = 0.10 hr^{-1} (forward kinetic reaction rate), k_2 = 0.10 hr^{-1} (backward kinetic reaction rate), U = 0.5 (non-linear kinetic parameter), k_3 = 0.10 hr^{-1} (forward kinetic reaction rate), k_4 = 0.01 hr^{-1} (backward kinetic reaction rate), W = 0.5 (non-linear kinetic parameter), k_5 = 0.01 hr^{-1} (forward kinetic reaction rate), k_6 = 0.1 hr^{-1} (backward kinetic reaction rate), k_s = 0.001 (irreversible reaction rate).

The objective was to determine which combinations of solute concentration (input pulse) and water flux will provide the lowest concentrations after 10 hours of contaminant application

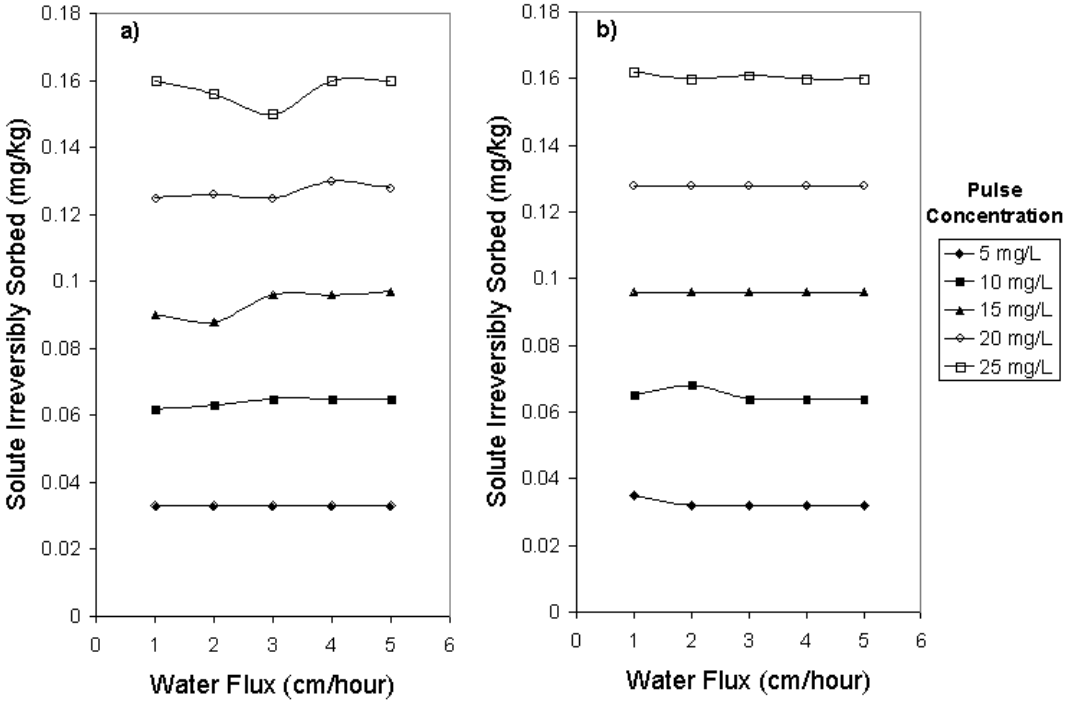

Figure 5.1: Solute irreversible sorbed in the soil. (a) Model estimations of the maximum amounts of solute irreversibly sorbed in the soil after 10 hours of a 20-hour input pulse of contaminant, under several combinations of input pulse concentrations and water fluxes. (b) After a total of 100 hours (80 additional hours of water application) the maximum amounts remain at the same values.

and after one hundred hours of washing the soil with water alone (the total 100 hrs include the initial 10 hrs of application of the contaminant). Since the concentrations through the soil profile will be different, only the maximum concentrations are to be monitored. The location in the soil profile (of those peak concentrations) was also required. The input pulse concentrations values will be: 5, 10, 15, 20 and 25 mg/L, combined with the following water flux values: 1, 2, 3, 4, 5 cm/hr, giving a total of 25 scenarios.

Figures 5.1 and 5.2 summarize the results provided by the web-based simulation environment. These two charts resulted from post-processing of the numerical results displayed by the interface in the form of concentration-depth curves. The results corresponding to the solute irreversibly sorbed in the soil showed that the maximum concentrations were always located at the top of the soil profile and decreased almost linearly to negligible amounts at the

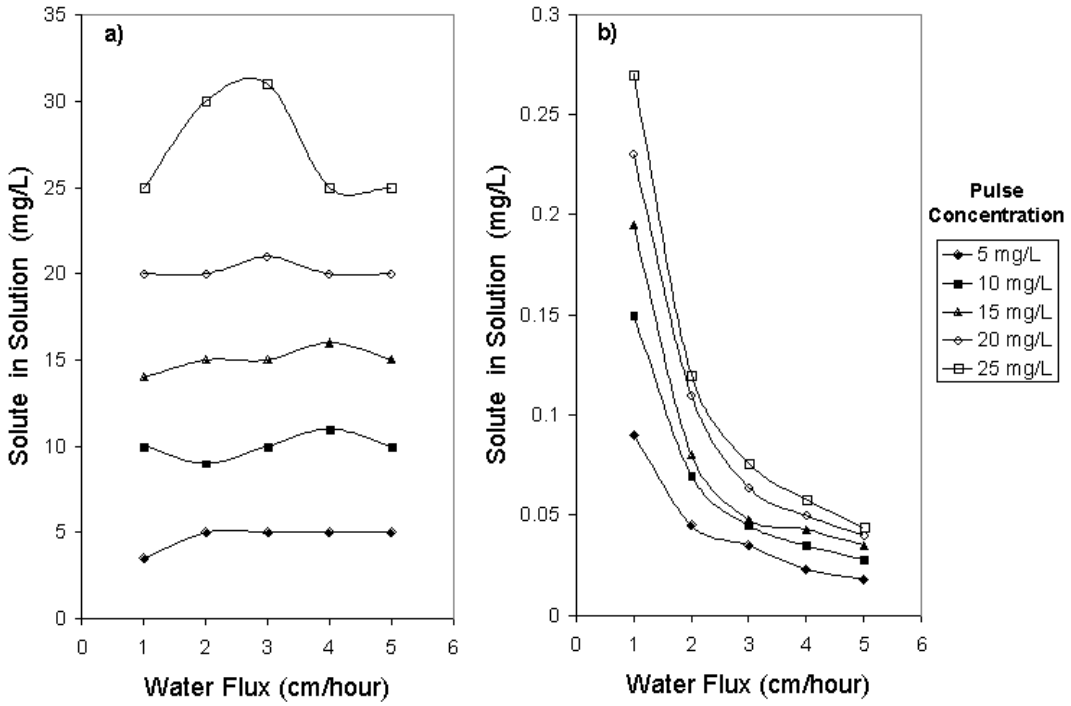

Figure 5.2: Maximum concentrations in solution. (a) After 10 hours of the 20-hour input pulse the maximum concentrations of the dissolved solute in soil are almost equal to the concentrations of the input pulse. (b) Solution concentrations after 100 hours are lower than 0.3 mg/L regardless of the input pulse concentration. For water fluxes greater than 3 cm/hr, the maximum concentrations are lower than 0.1 mg/L for all the input pulse concentrations.

bottom of the soil layer. The maximum concentrations of solute irreversibly sorbed in the soil for all the scenarios, after 10 and 100 hours, are shown in Figure 5.1. The figures show that the solute is uniformly sorbed through the experiment. Washing down the soil would not be an effective remediation measure. The concentrations of sorbed solute, however, are lower than 0.2 mg/kg in all cases.

Peak concentrations of the solute in solution were estimated to be located at the top of the soil profile after 10 hours of contaminant application. Those values decreased smoothly up to negligible amounts at the bottom of the profile. For all the scenarios analyzed, the maximum concentrations after 10 hours (Figure 5.2 a) were almost identical to the input pulse concentrations. After 100 hundred hours, the peak concentrations were estimated to be at the

bottom of the soil profile, with the top sections being almost contaminant-free. Figure 5.2 b shows that for high water fluxes (3 to 5) the maximum concentrations are lower than 0.1 mg/L, regardless of the input pulse concentration.

To summarize, the one-dimensional MRTM model estimates that a soil layer, with the physical and chemical conditions specified by the input data given here, will uniformly and irreversibly retain contaminant X in amounts lower than 0.2 mg/kg, regardless of the input pulse concentration (ranging from 5 to 25 mg/L). Similarly, after 80 additional hours of water flowing through the soil, water fluxes higher than 3 cm/hr result in concentrations lower than 0.1 mg/L exiting at the bottom of the soil layer.

The JNI-based simulation system efficiently allows the reuse of legacy code and computational kernels written in Fortran or C through a Java/C interface that includes web-based Swing features and JNI. The use of a C-Fortran interface to build a shared (or dynamic) library and overcome JNI's inability of using Fortran code directly is also proven to be successful.

The almost instantaneous visualization provided by the web interface resulted in an efficient and easy analysis of a hypothetical soil contamination problem. This simulation system is more appropriate for fast exploration of scenarios, and provides numerical output (along with visualization with the Java interface) for more detailed analysis. Web-based features enable users to observe chemical reaction results following changes in various parameters and to easily identify trends and patterns from computational simulation.

5.2 Three-dimensional MRTM Verification

Since laboratory experiments are difficult to implement for three-dimensional chemical transport and retention, this research compares the three-dimensional MRTM numerical results with the one-dimensional MRTM numerical results, so as to verify the correctness of the MRTM model extension to the three-dimensional domain. To create comparable results, a nominal case was analyzed in which all spatial and temporal variables and parameters are set to a range of 0 to 1. This normalization step provides means to compare results avoiding the dimensional analysis

that is usually involved. To perform the comparison, the contaminant concentration(solute)-depth curves are plotted in the vertical direction for the following two scenarios.

5.2.1 Case 1

A nominal contaminant-free soil layer of 1 cm depth, having a volumetric moisture content of 0.4 cm^3/cm^3 and 1.25 g/cm^3 bulk density, received a load of a diluted contaminant X during 1 hour. The dispersion coefficient in the vertical direction was estimated to be 0.01 cm^2/hr, and the water flux 1 cm/hr.

The following set of compound-soil interaction parameters were used: k_d = 1.0 cm^3/g (distribution coefficient), NEQ = 1.1 (Freundlich parameter), k_1 = 0.01 hr^{-1} (forward kinetic reaction rate), k_2 = 0.02 hr^{-1} (backward kinetic reaction rate), U = 1.3 (non-linear kinetic parameter), k_3 = 0.01 hr^{-1} (forward kinetic reaction rate), k_4 = 0.02 hr^{-1} (backward kinetic reaction rate), W = 1.2 (non-linear kinetic parameter), k_5 = 0.01 hr^{-1} (forward kinetic reaction rate), k_6 = 0.02 hr^{-1} (backward kinetic reaction rate), k_s = 0.005 (irreversible reaction rate).

Figures 5.3, 5.5, 5.7, 5.9, 5.11, and 5.13 show the six-phase results generated by the 1-D MRTM model for ten consecutive time steps (with time increment 0.1 hr). The one-dimensional computational grids are 320 in space and 2500 in time domain. Figures 5.4, 5.6, 5.8, 5.10, 5.12, and 5.14 show the six-phase results generated by the 3-D MRTM model for ten consecutive time steps. The computational grids are $10 \times 10 \times 80$ in space and 20 in time.

As shown in Figure 5.3, the one-dimensional MRTM model predicts that the contaminant concentration in the soil solution is almost null at a nominal depth of 0.1 after one time step. After ten time steps, the contaminant concentration almost disappears at 0.8 of nominal depth. The concentration-depth values through the soil profile, for all time-steps, decrease smoothly from 1.0 (at the top soil layer) to zero. The curves are of sigmoidal shape meaning that the fastest concentration depletion occurs at the middle soil layers. The model predicts that the top soil layers will be saturated with the contaminant for time-steps 4 to 10 from 0 to 0.5 of nominal depth. Even the first three time steps show saturation at smaller portions of top soil layers.

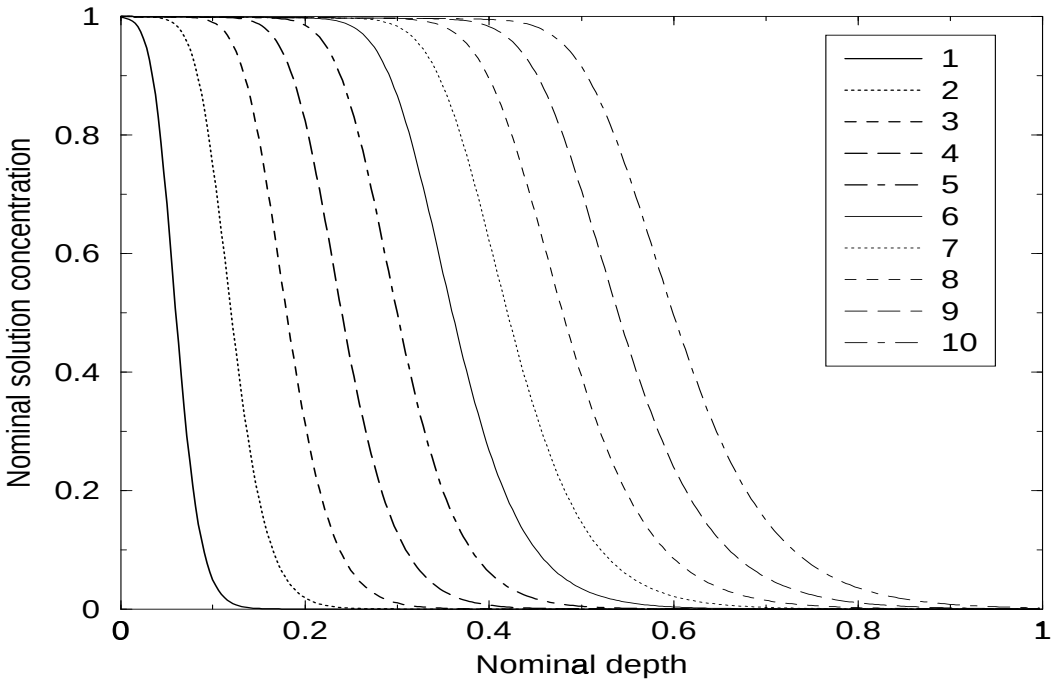

Figure 5.3: Concentration-depth curves for 1D MRTM case 1.

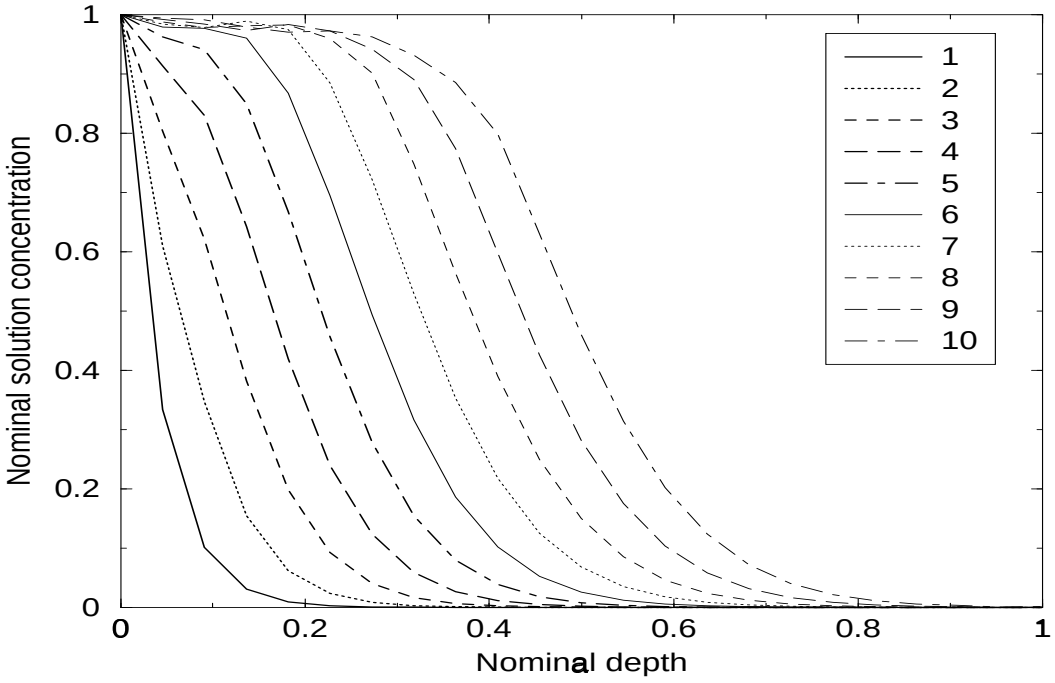

Figure 5.4: Concentration-depth curves for 3D MRTM case 1.

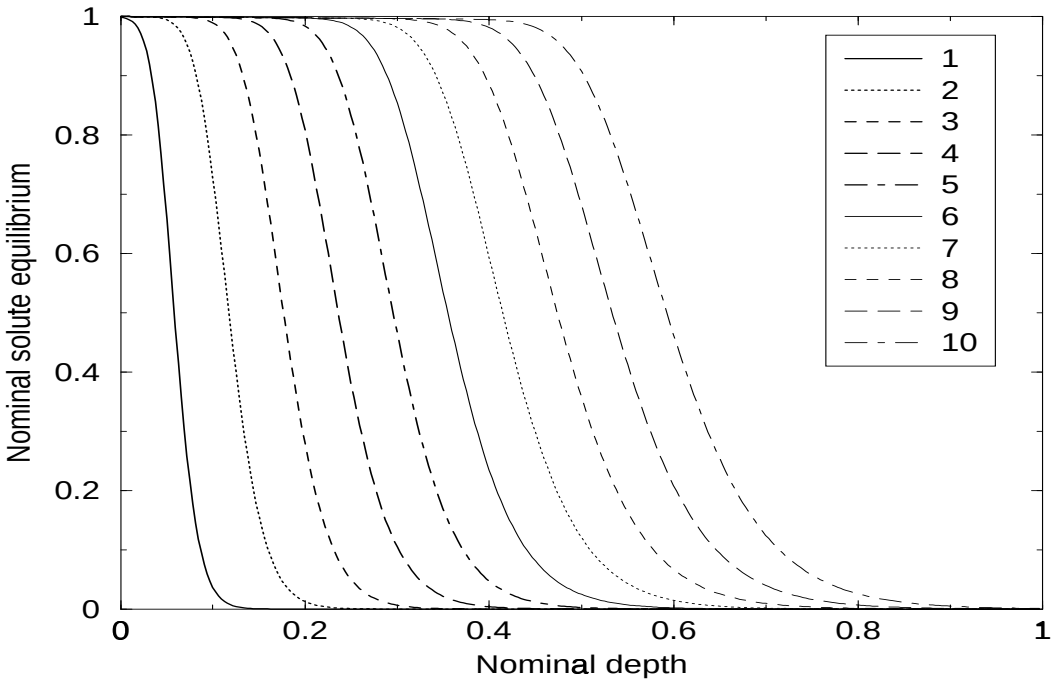

Figure 5.5: Solute-depth (Se) curves for 1D MRTM case 1.

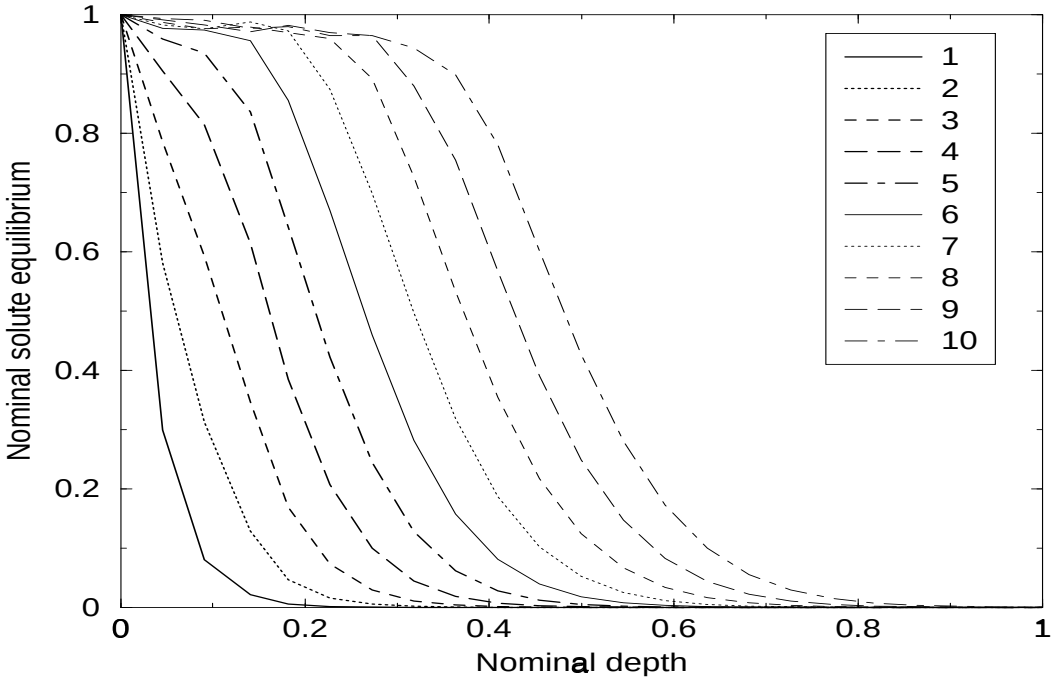

Figure 5.6: Solute-depth (Se) curves for 3D MRTM case 1.

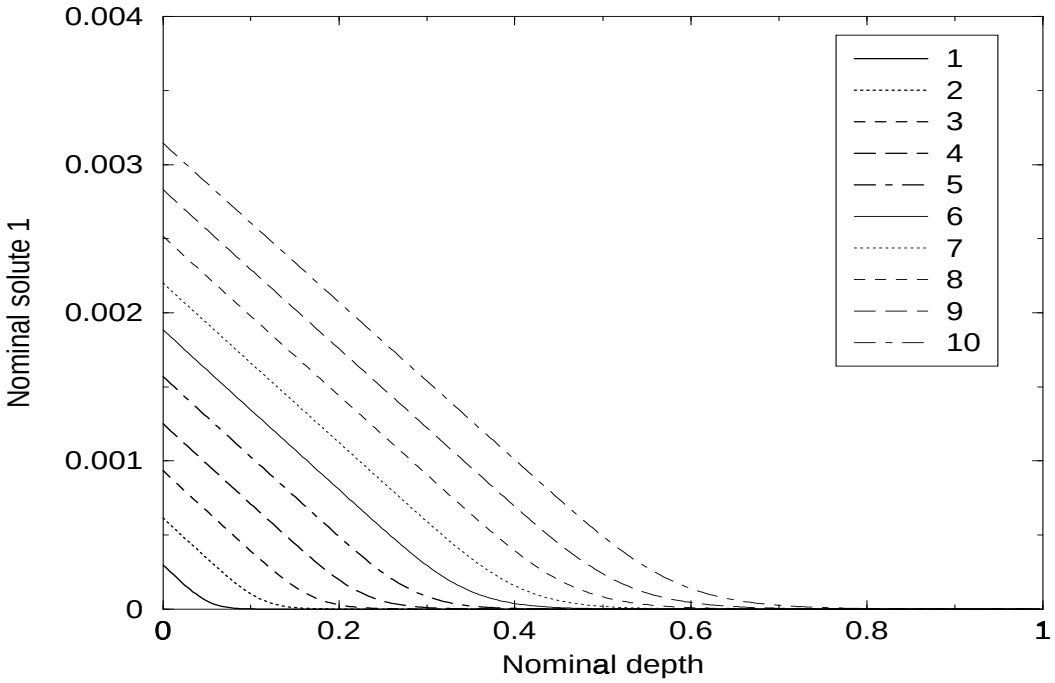

Figure 5.7: Solute-depth (S1) curves for 1D MRTM case 1.

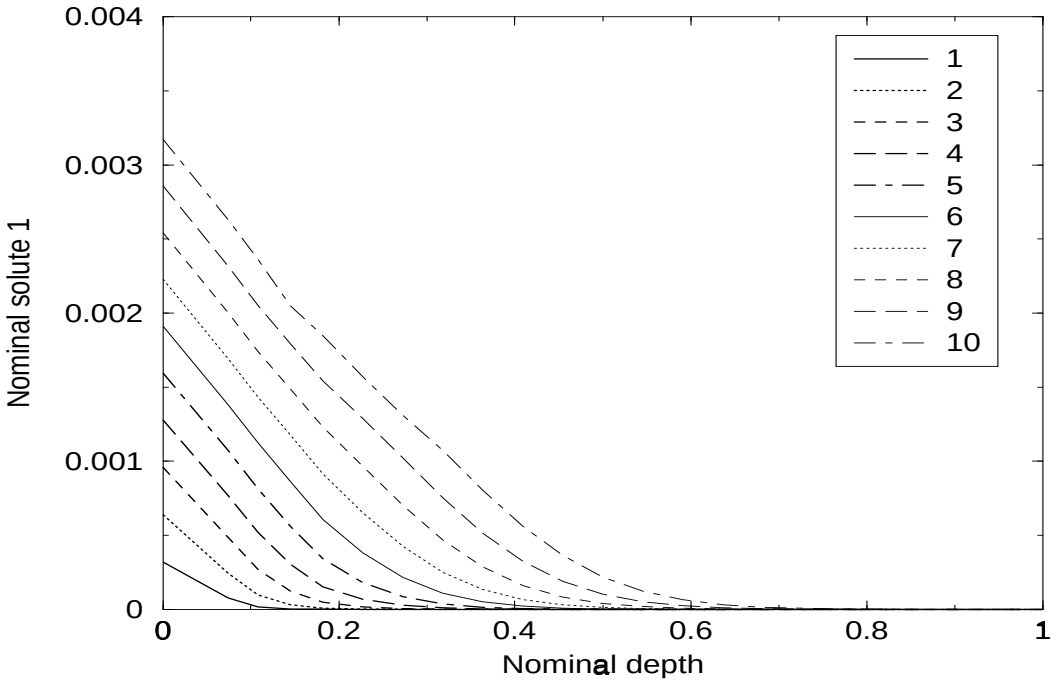

Figure 5.8: Solute-depth (S1) curves for 3D MRTM case 1.

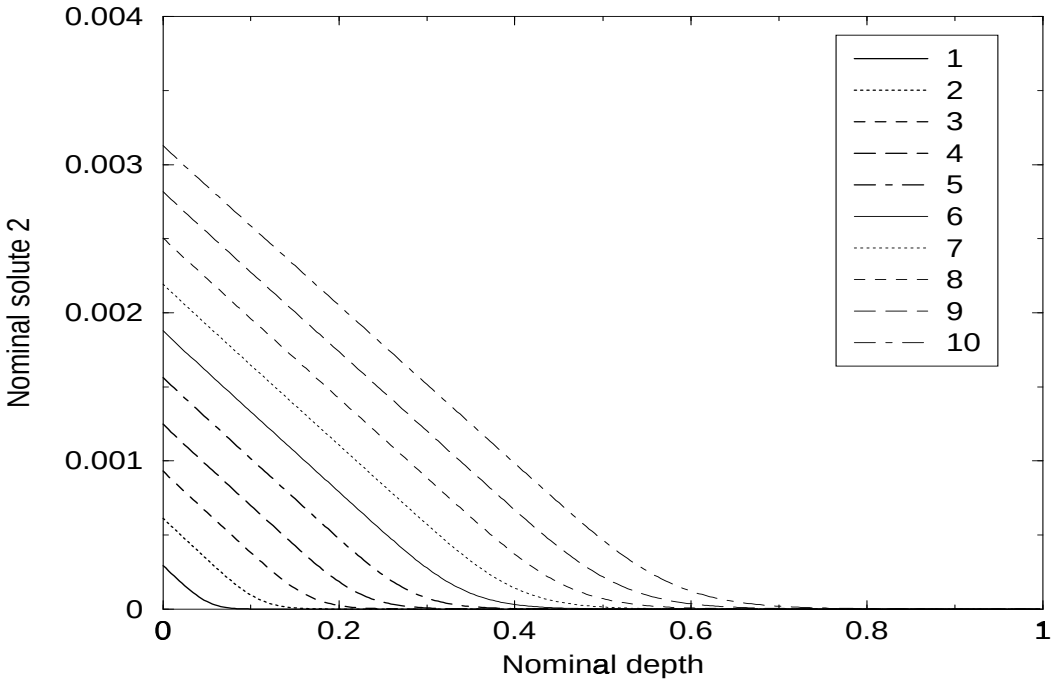

Figure 5.9: Solute-depth (S2) curves for 1D MRTM case 1.

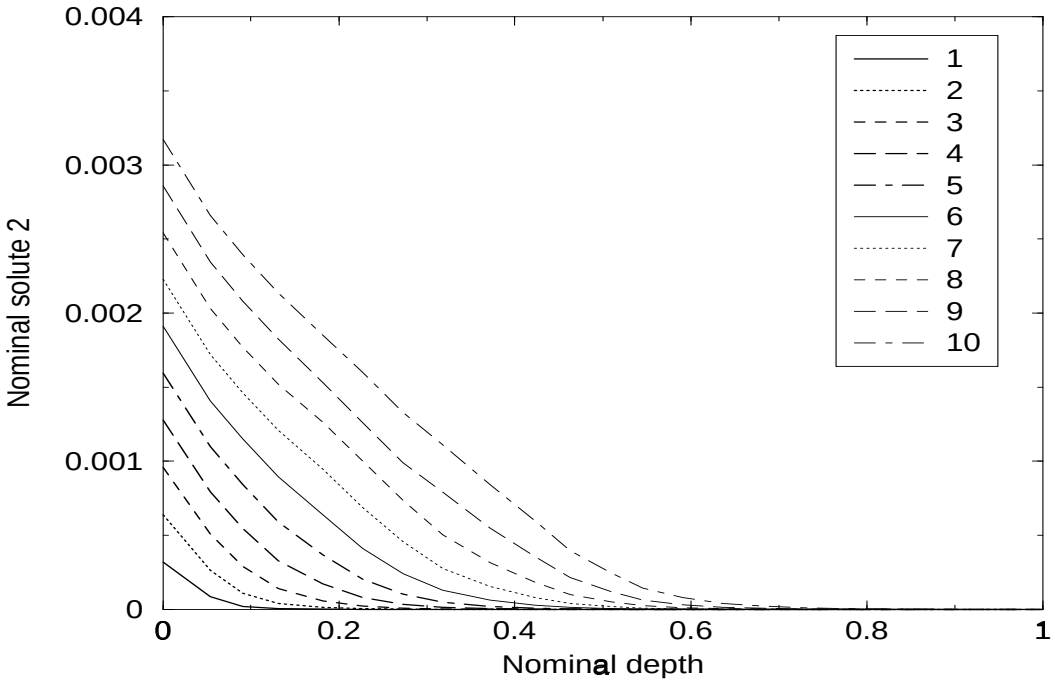

Figure 5.10: Solute-depth (S2) curves for 3D MRTM case 1.

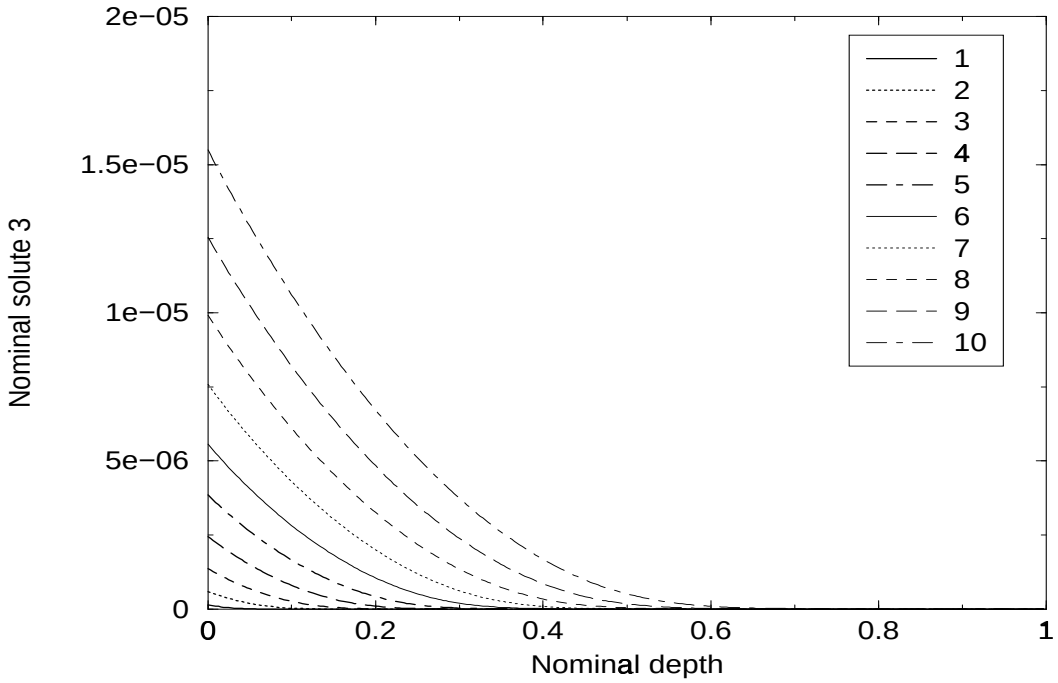

Figure 5.11: Solute-depth (S3) curves for 1D MRTM case 1.

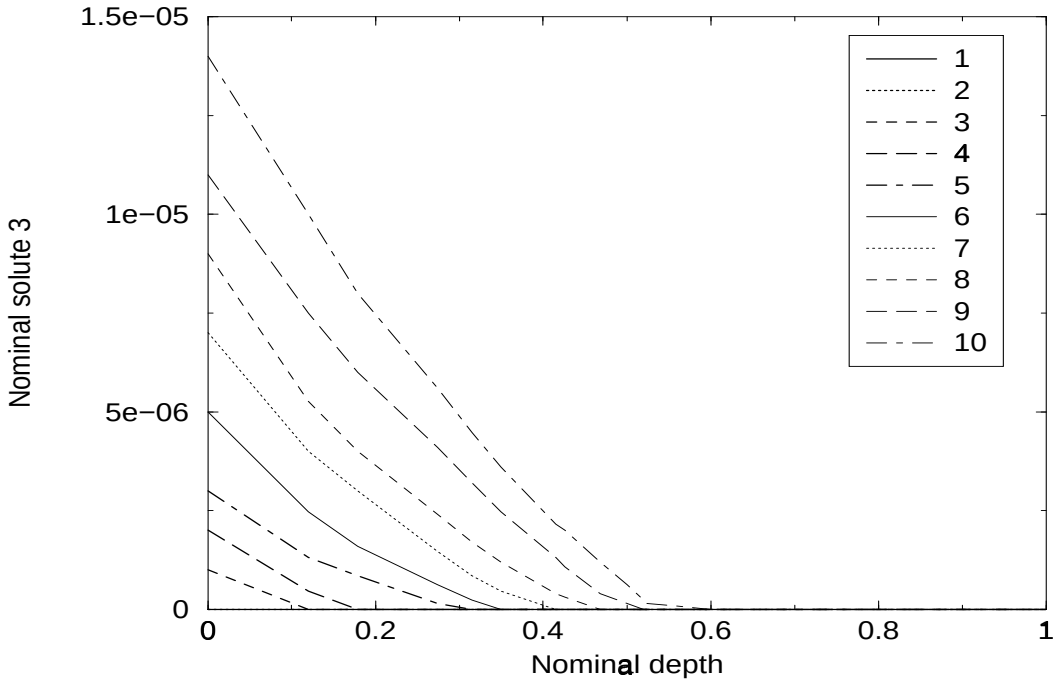

Figure 5.12: Solute-depth (S3) curves for 3D MRTM case 1.

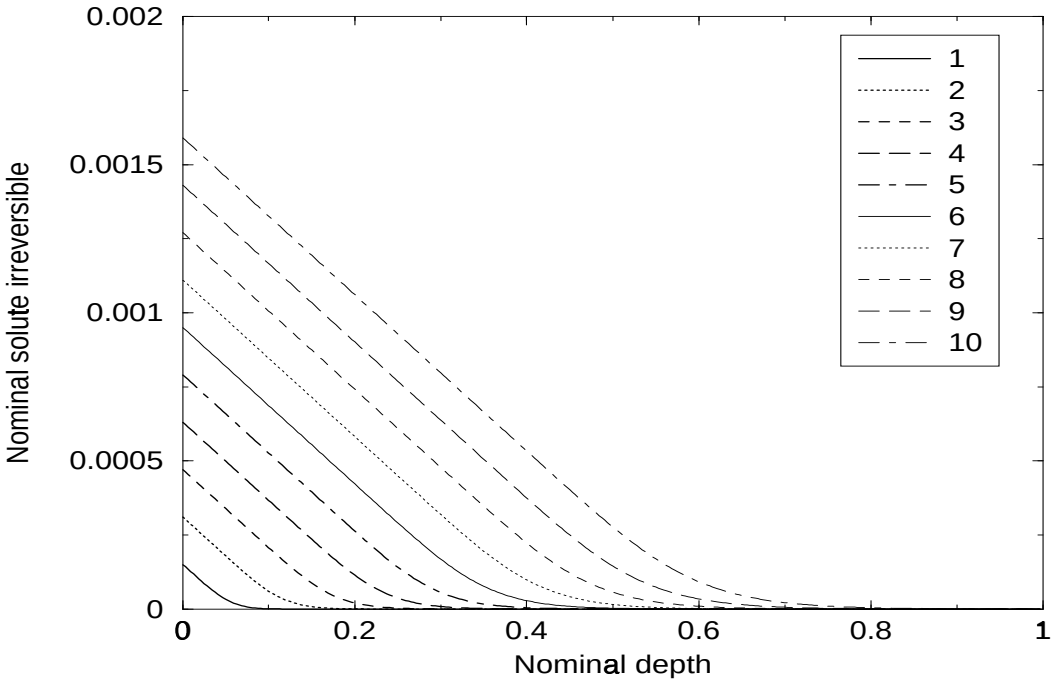

Figure 5.13: Solute-depth (Sirr) curves for 1D MRTM case 1.

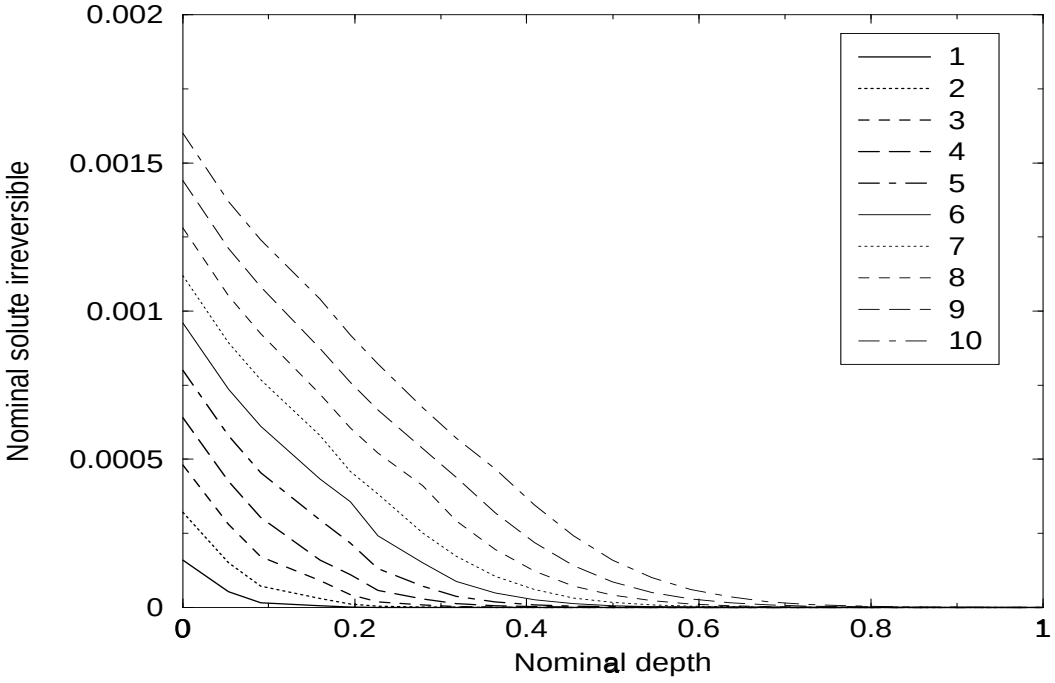

Figure 5.14: Solute-depth (Sirr) curves for 3D MRTM case 1.

As shown in Figure 5.4 of the three-dimensional MRTM simulation, the contaminant concentration depletion does not occur up to 0.2 of nominal depth after 1 time-step. After ten time-steps, the contaminant concentration disappears at around 0.8 of nominal depth. For most of the time-steps, the concentration-depth curve is of a sigmoidal shape but the slope of the linear portion of the curve (corresponding to the middle soil layers) is less strong than the slope of the 1-D MRTM simulation. This means that the 3-D MRTM estimates that the middle soil layers are less efficient in retaining the contaminant, hence the soil solution carries the contaminant further down the soil depth. The first three curves are of an exponential shape meaning that the middle soil layers, initially devoid of any solute presence, are very efficient in retaining the contaminant. This efficiency decreases as time goes on. The top layers show that, in almost all cases, maximum concentration in the soil solution does not reach total saturation. The model predicts that those soil top layers will only be partially saturated with the contaminant after the third time-step (greater than 90 % saturation) from 0 to 0.3 of nominal depth.

The 3-D model uses rather coarse computational grids than the 1-D model, in order to decrease the computations. However, the results from both models still seem to be consistent, and reflect the accuracy of the 3-D model. The concentrations estimated by the 3-D MRTM seem to reflect better the actual phenomena since it shows that immediate saturation of the top soil layers does not occur (this would occur only if flow and dispersion in the top horizontal planes would have unrealistic high values).

5.2.2 Case 2

Another numerical simulation comparison is take to reveal how the kinetic reaction rate, dispersion coefficient, and bulk density will affect the chemical transport and retention, with a different set of parameters.

The nominal contaminant-free soil layer of 1 cm depth, having a volumetric moisture content of 0.4 cm^3/cm^3 and 1.1 g/cm^3 bulk density, received a load of a diluted contaminant X during 1 hour. The dispersion coefficient in the vertical direction was estimated to be 0.1 cm^2/hr, and the water flux 1 cm/hr.

The following set of compound-soil interaction parameters was used: k_d = 3.0 cm^3/g (distribution coefficient), NEQ = 1.1 (Freundlich parameter), k_1 = 0.1 hr^{-1} (forward kinetic reaction rate), k_2 = 0.2 hr^{-1} (backward kinetic reaction rate), U = 1.3 (non-linear kinetic parameter), k_3 = 0.02 hr^{-1} (forward kinetic reaction rate), k_4 = 0.04 hr^{-1} (backward kinetic reaction rate), W = 1.2 (non-linear kinetic parameter), k_5 = 0.02 hr^{-1} (forward kinetic reaction rate), k_6 = 0.04 hr^{-1} (backward kinetic reaction rate), k_s = 0.01 (irreversible reaction rate).

Figures 5.15, 5.17, 5.19, 5.21, 5.23, and 5.25 show the six-phase results generated by the 1-D MRTM model for ten consecutive time steps (with time increment 0.1 hr). The one-dimensional computational grids are 320 in space and 2500 in time domain. Figures 5.16, 5.18, 5.20, 5.22, 5.24, and 5.26 show the six-phase results generated by the 3-D MRTM model for ten consecutive time steps. The computational grids are $10 \times 10 \times 80$ in space and 20 in time.

Case 2 increases the dispersion coefficient from 0.01 cm^2/hr to 0.1 cm^2/hr, and decreases the bulk density from 1.25 g/cm^3 to 1.1 g/cm^3. As shown in Figures 5.15 (1-D) and 5.16 (3-D), the concentration-depth values through the soil profile, for all time-steps, decrease smoothly from top soil layer to bottom. The curves have more linear shape tendency than those in Case 1. There is less saturation at top layers than Case 1 too. These phenomena can be explained by the effects of strong dispersion and loose soil density properties.

Case 2 also increases the kinetic reaction rates, and causes much stronger retentions than Case 1. The retention trends are reflected from Figures 5.19, 5.21, 5.23, 5.25 for 1-D cases, and Figures 5.20, 5.22, 5.24, 5.26 for 3-D cases. Similarly to Case 1, the 3-D model uses rather coarse computational grids than the 1-D model, in order to decrease the computations. The results from both models still seem to be consistent, and reflect the accuracy of the 3-D model.

5.3 Contaminant Trace with Three-dimensional MRTM Simulation

One of the immediate applicabilities of the three-dimensional MRTM model is to simulate advection and dispersion of contaminants in soil columns with low permeability and strong retention mechanisms. In those cases the contaminant would not be expected to travel to the lowest layers of the soil profile very soon. When the trace compound is hazardous (e.g., a heavy

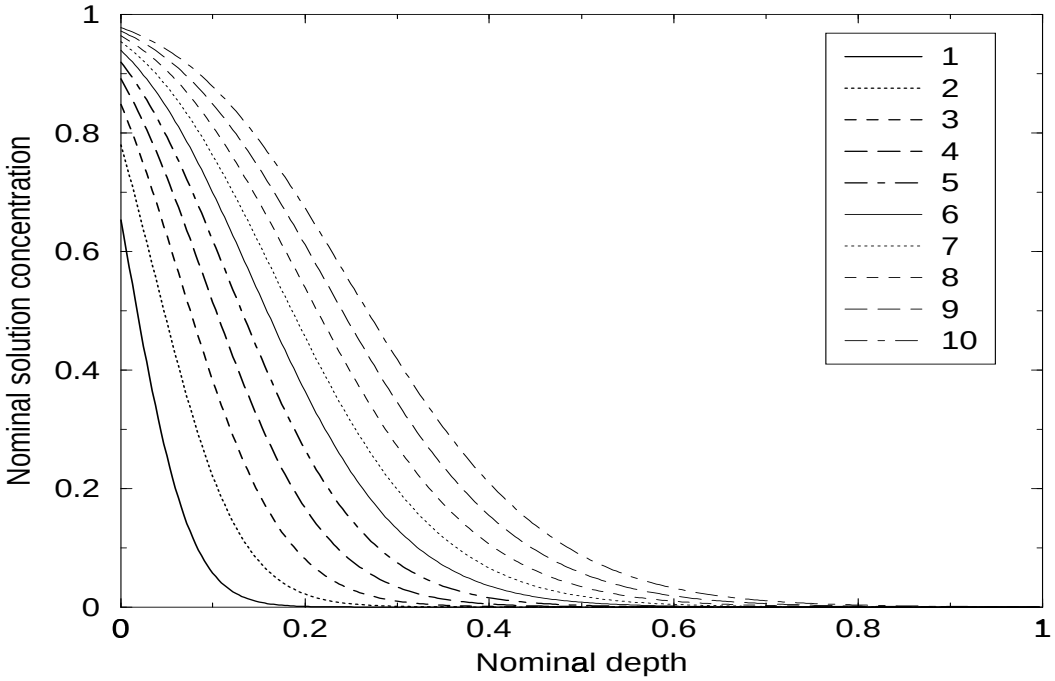

Figure 5.15: Concentration-depth curves for 1D MRTM case 2.

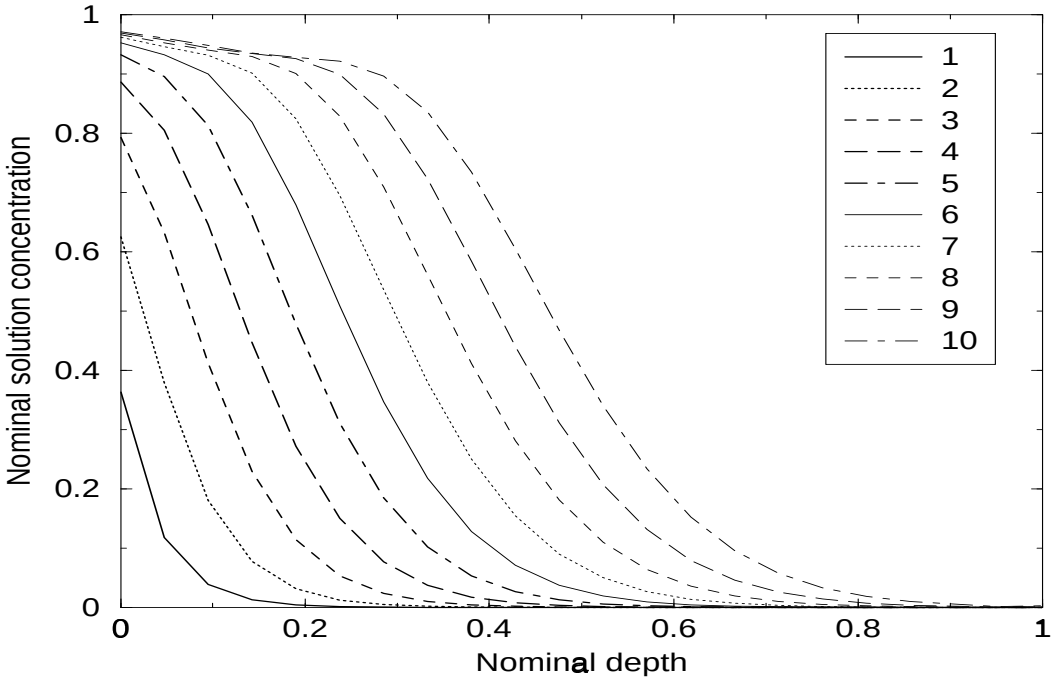

Figure 5.16: Concentration-depth curves for 3D MRTM case 2.

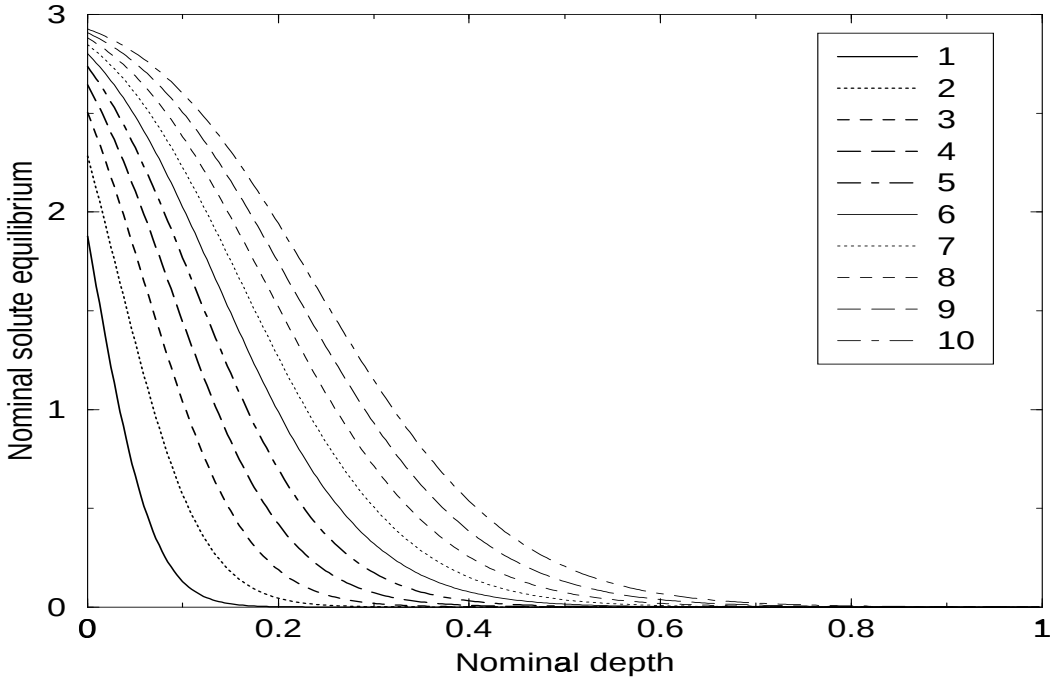

Figure 5.17: Solute-depth (Se) curves for 1D MRTM case 2.

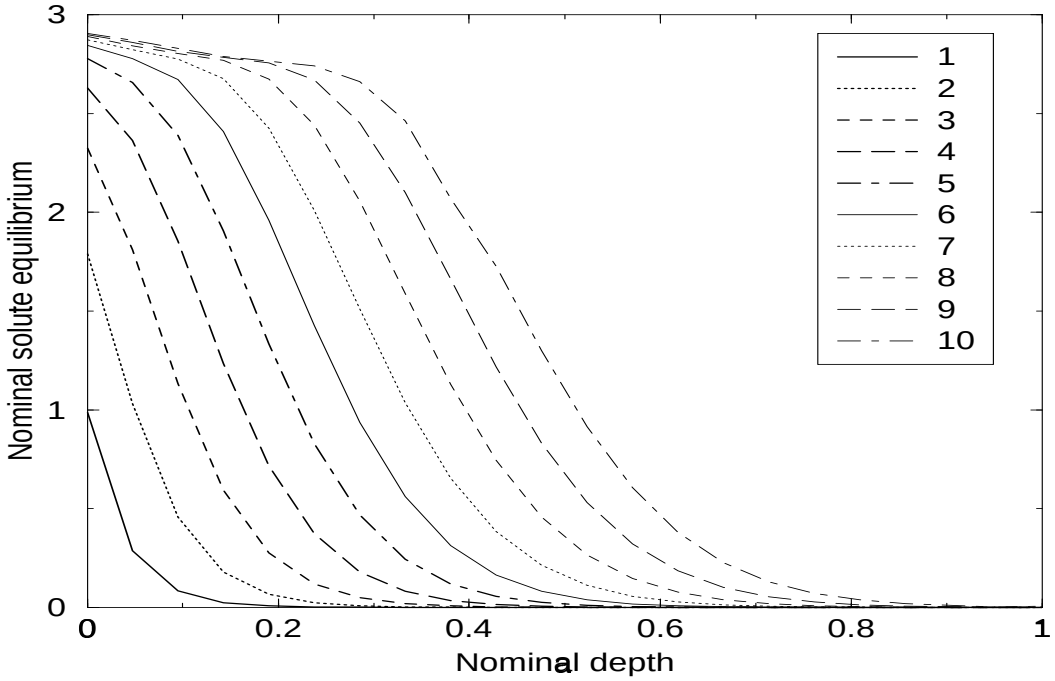

Figure 5.18: Solute-depth (Se) curves for 3D MRTM case 2.

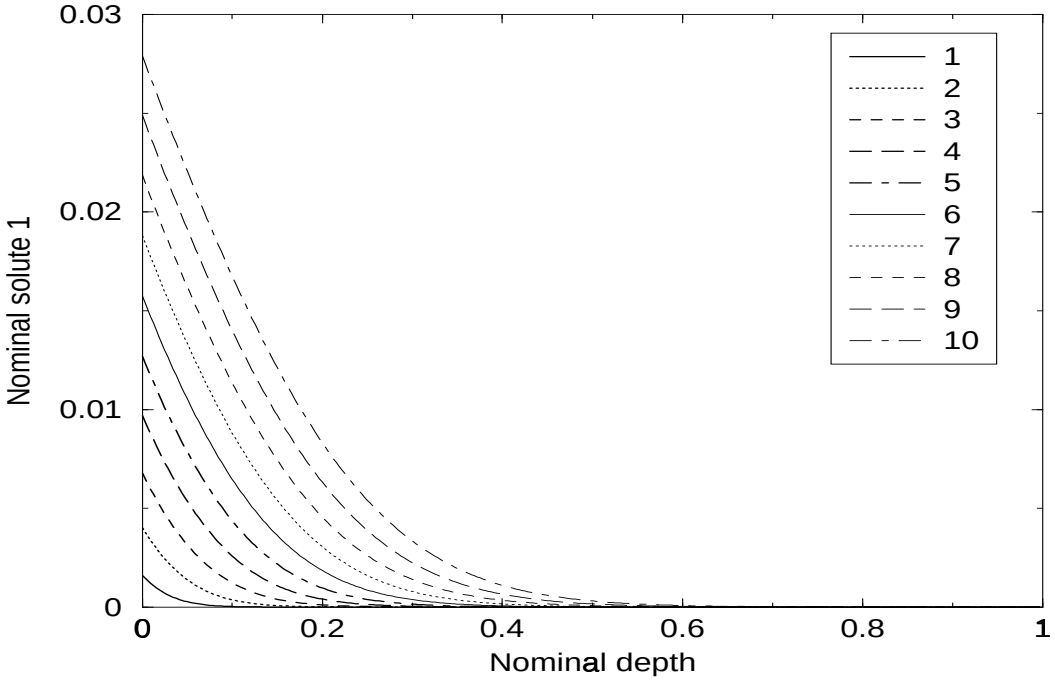

Figure 5.19: Solute-depth (S1) curves for 1D MRTM case 2.

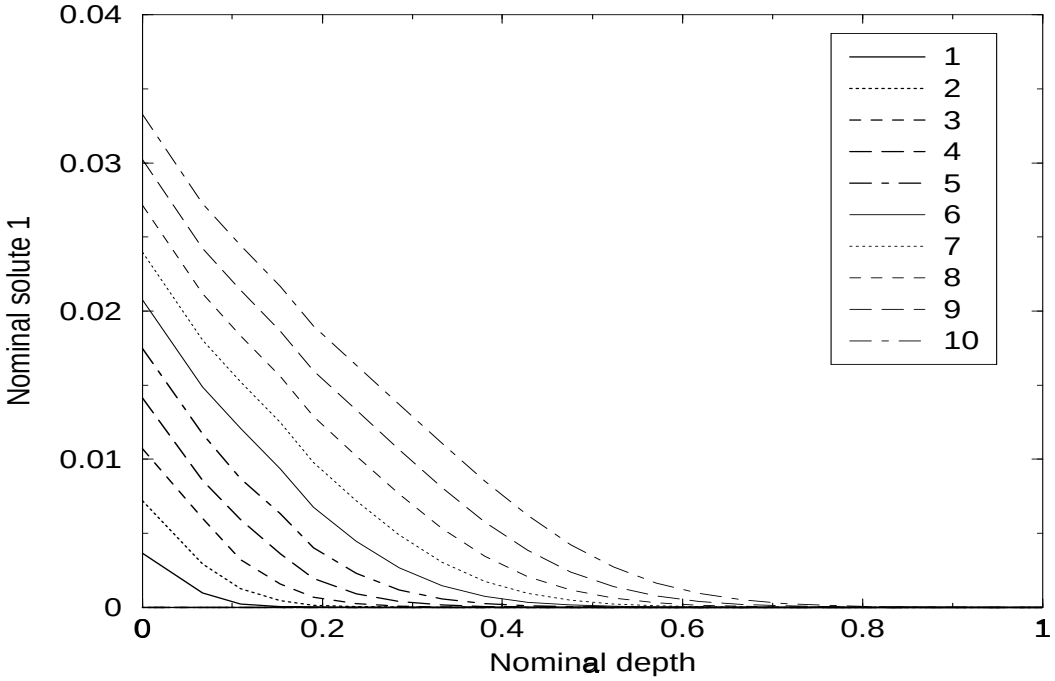

Figure 5.20: Solute-depth (S1) curves for 3D MRTM case 2.

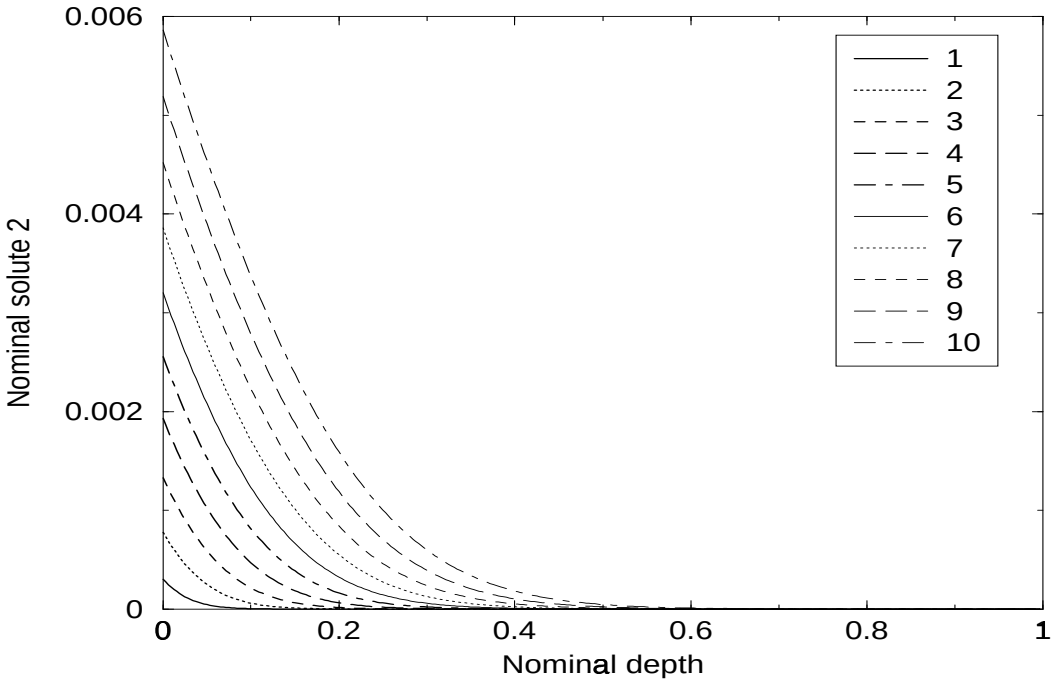

Figure 5.21: Solute-depth (S2) curves for 1D MRTM case 2.

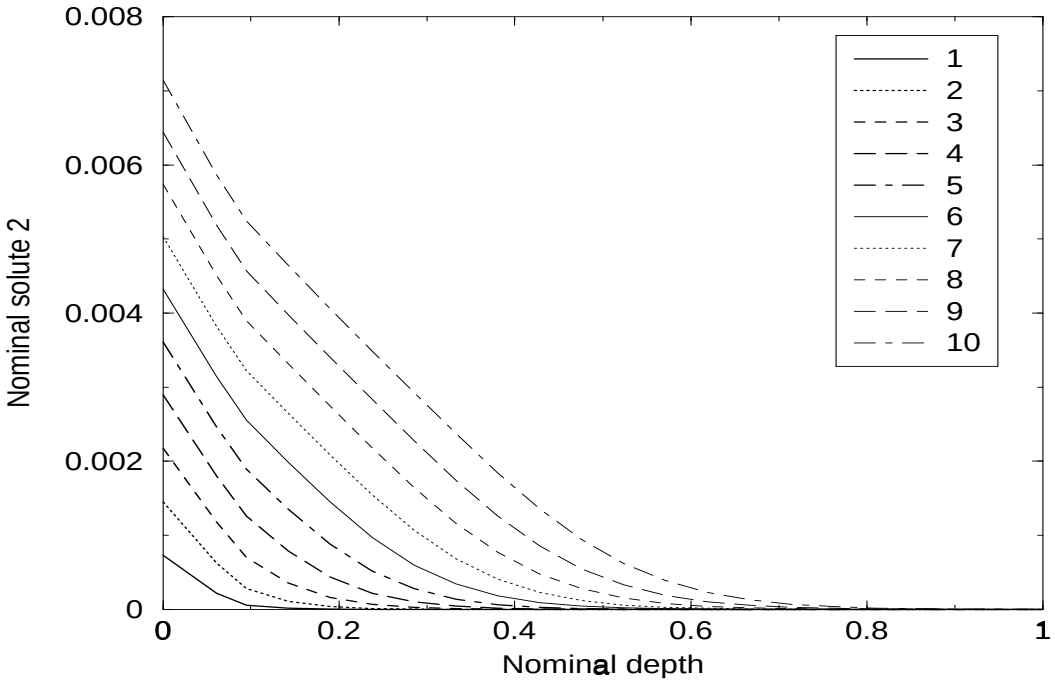

Figure 5.22: Solute-depth (S2) curves for 3D MRTM case 2.

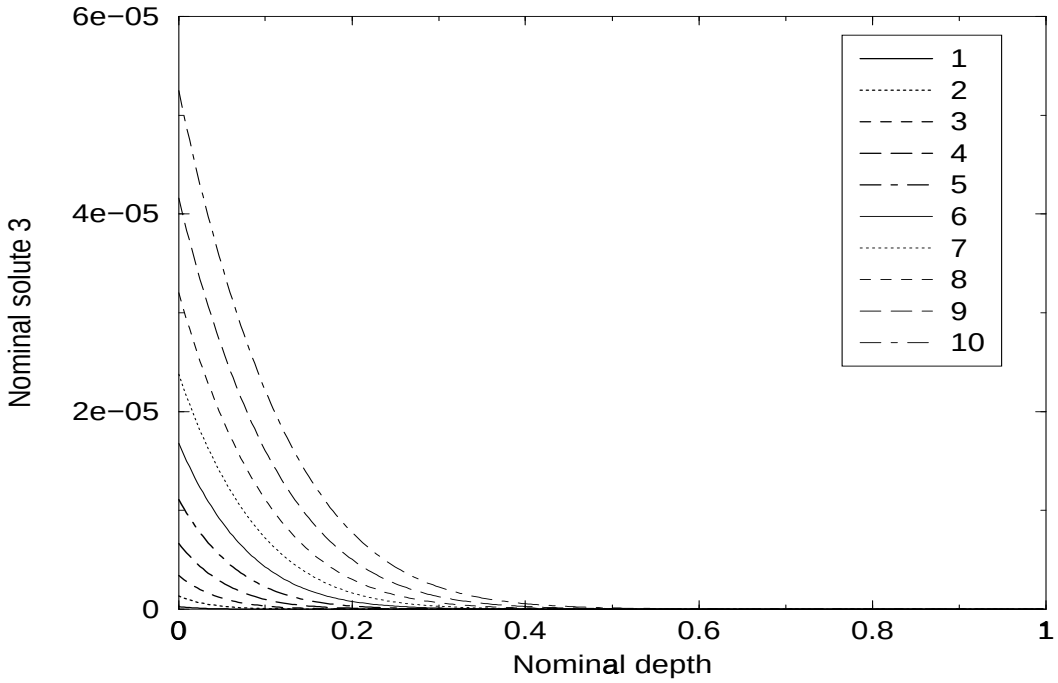

Figure 5.23: Solute-depth (S3) curves for 1D MRTM case 2.

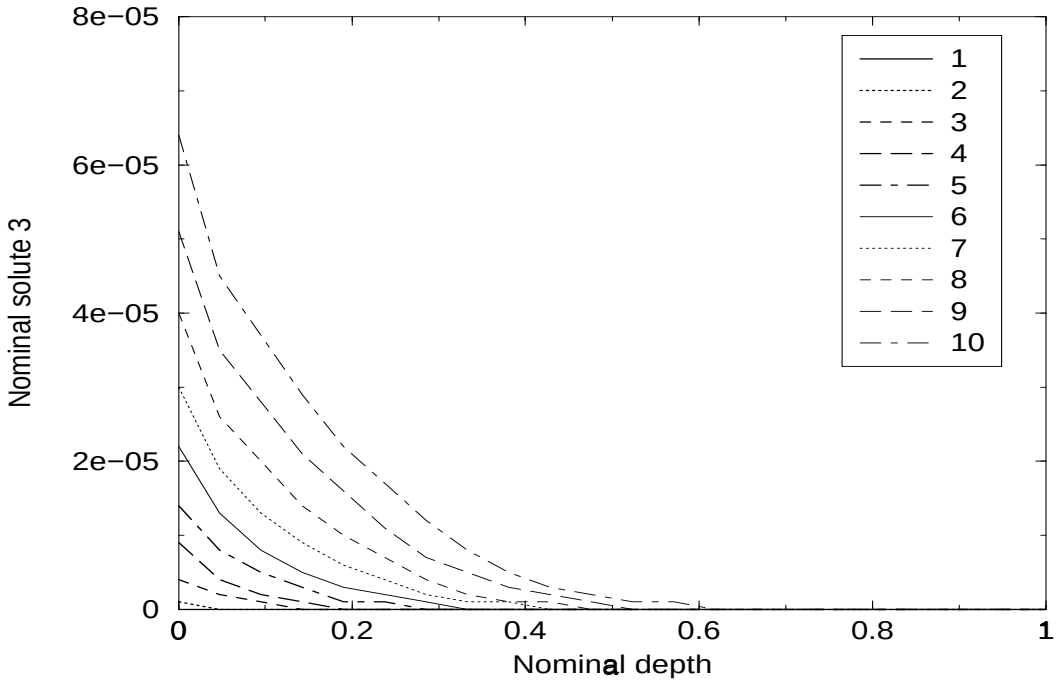

Figure 5.24: Solute-depth (S3) curves for 3D MRTM case 2.

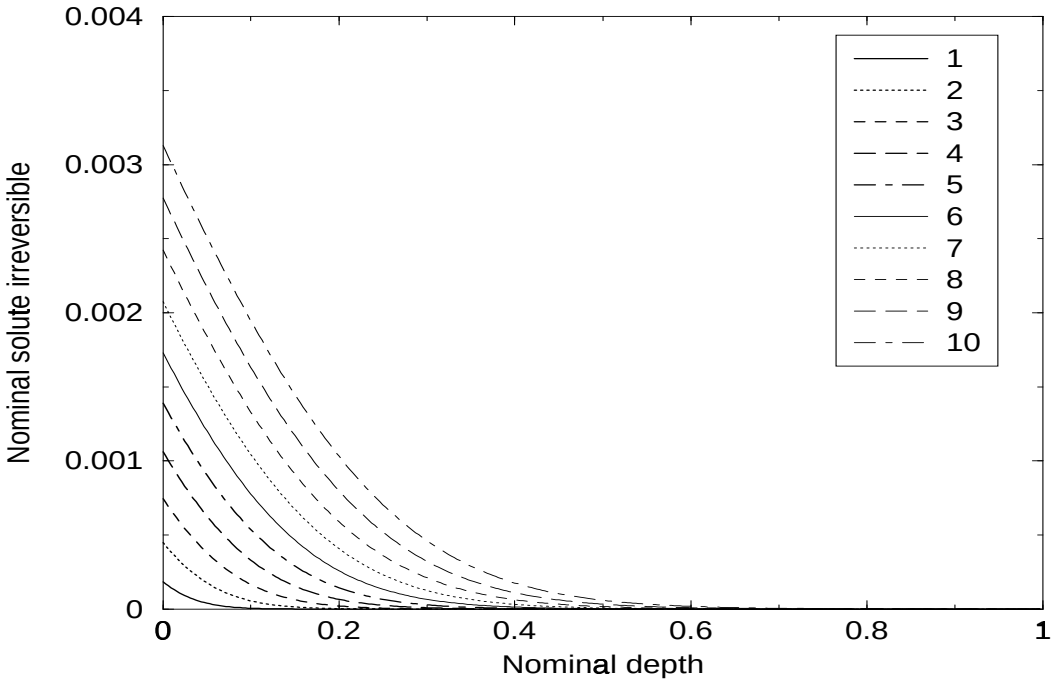

Figure 5.25: Solute-depth (Sirr) curves for 1D MRTM case 2.

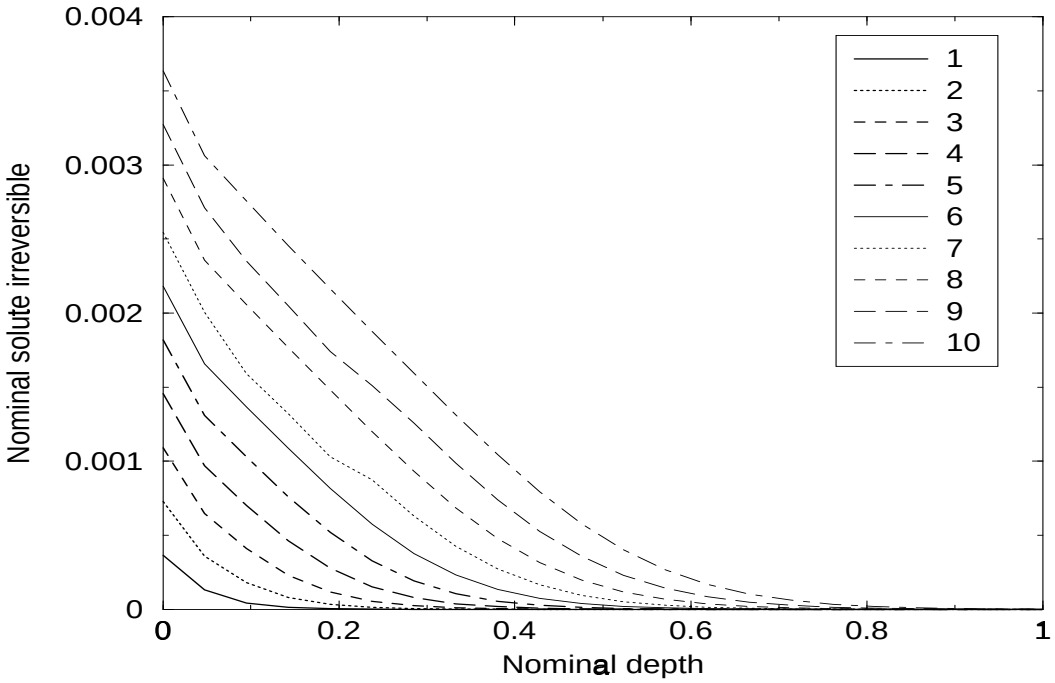

Figure 5.26: Solute-depth (Sirr) curves for 3D MRTM case 2.

metal such as mercury), it is also necessary to monitor the spatial distribution and amounts of very low concentrations. The current 3D-MRTM visualization provides a efficient means to track those types of compounds, simulating the results from different contaminant-input conceptualizations.

The first scenario consists of a localized contaminant load (point source) located at the center of the top soil column layer (point source contaminant inputs may simulate localized leakages of chemicals on the surface of the earth). The second simulation consists of a linearly distributed contaminant load (line source) over the top surface of a soil column. Line sources are commonly used in groundwater contamination problems [70]. Additionally, a two-point source simulation is included for purposes of comparison.

The contaminant chosen for this application is a trace compound such as mercury (Hg). The Environmental Protection agency [71] sets the Criteria Maximum Concentration (CMC) for mercury in fresh water at 1.4 $\mu g/L$, and the Criterion Continuous Concentration (CCC) at 0.77 $\mu g/L$. Given the three types of contaminant sources available in 3D-MRTM, this application seeks to know how different the three dimensional distribution of the contaminant (in the soil solution) would be for each of those input sources.

In all simulations, the column consists of a contaminant-free soil of 1 cm depth, having a volumetric moisture content of 0.4 cm^3/cm^3 and a bulk density of 1.25 g/cm^3. It receives a load of $1mg/L$ (i.e., 1000 $\mu g/L$) of Hg in solution during 1 hour. The dispersion coefficient of the aqueous solution in the soil is isotropic and estimated to be 0.01 cm^2/hr in all directions. The Darcy flux (cm/hr) is anisotropic with: q_x = 0.0 cm/hr, q_y = 0.0 cm/hr, q_z = 0.05 cm/hr for symmetric concentration distribution on XY cutting planes, and q_x = 0.1 cm/hr, q_y = 0.15 cm/hr, q_z = 0.1 cm/hr for asymmetric concentration distribution on XY cutting planes.

The compound-soil interaction is reflected in the following parameters: k_d = 1.0 cm^3/g (distribution coefficient), NEQ = 1.1 (Freundlich parameter), k_1 = 0.01 hr^{-1} (forward kinetic reaction rate), k_2 = 0.02 hr^{-1} (backward kinetic reaction rate), U = 1.2 (non-linear kinetic parameter), k_3 = 0.01 hr^{-1} (forward kinetic reaction rate), k_4 = 0.02 hr^{-1} (backward kinetic

reaction rate), W = 1.3 (non-linear kinetic parameter), k_5 = 0.01 hr^{-1} (forward kinetic reaction rate), k_6 = 0.02 hr^{-1} (backward kinetic reaction rate), k_s = 0.005 (irreversible reaction rate).

The spatial size of the grid used for all simulations was $23 \times 23 \times 23$, and the time domain is divided into 10 steps. Cutting (sliding) planes from number 0 to 22 in three directions are used by the model to show the spatial distribution of the concentration in shades of gray (the maximum concentration corresponds to black). Additionally, the model provides the maximum concentration values and the position of the plane in X (transverse plane), Y (longitudinal plane) or Z (horizontal plane) directions. X-direction is the direction of the arrow that points to the right of the reader. Y-direction is the direction of the arrow that points to the reader in the horizontal plane. Z-direction is the direction of the arrow that points downwards.

The three-dimensional distribution of the contaminant concentration for the point source is shown in Figures 5.27 to 5.38 using cutting planes in different directions. At the top (Figures 5.27 to 5.29), the maximum concentration is clearly equal to the input source concentration ($1 mg/L$ Hg). Two sets of anisotropic advection coefficients are applied with the same set of isotropic dispersion coefficients. Figures 5.30 to 5.32 show that the concentration loads are distributed symmetrically around the point source, because the advection in X and Y directions are set null. Figures 5.33 to 5.38 show that the concentration loads move smoothly from the center to corner, due to the effect from advection in X and Y directions. The maximum concentration at cutting plane 6 in Z direction at time step 2 (0.0009 mg/L in Figure 5.35) is below the CCM and CCC criteria for Hg, which is 0.0014 mg/L (1.4 $\mu g/L$). While the maximum concentration at cutting plane 21 in Y direction at time step 10 (0.0043 mg/L in Figure 5.37) is above both CCM and CCC criteria.

The transport associated with a line source of contaminant is also studied for the same data set (anisotropic media, isotropic dispersion). Figures 5.39 to 5.50 provide numerical details of the simulation (maximum concentration value), and show the three-dimensional contaminant distribution with different cutting planes in three directions. At the top (Figures 5.39 to 5.41), the maximum concentration is clearly equal to the input source concentration ($1 mg/L$ Hg). Figures

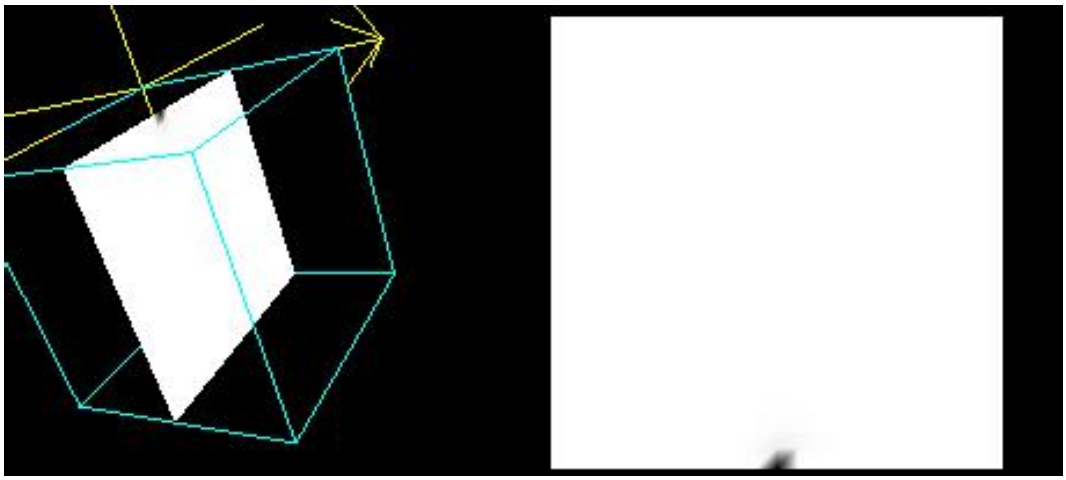

Figure 5.27: Point source: max 1 mg/L, time step 1, cutting plane 11 (X).

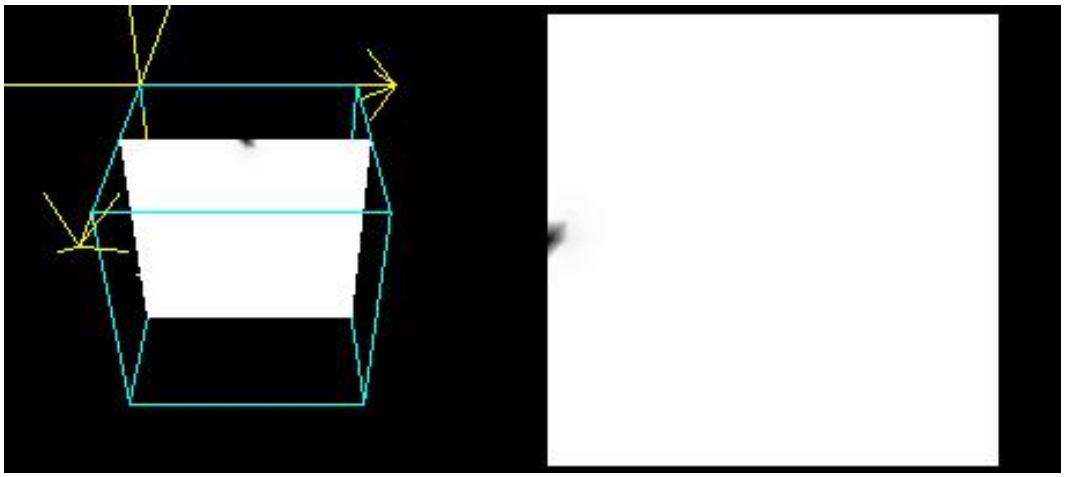

Figure 5.28: Point source: max 1 mg/L, time step 1, cutting plane 11 (Y).

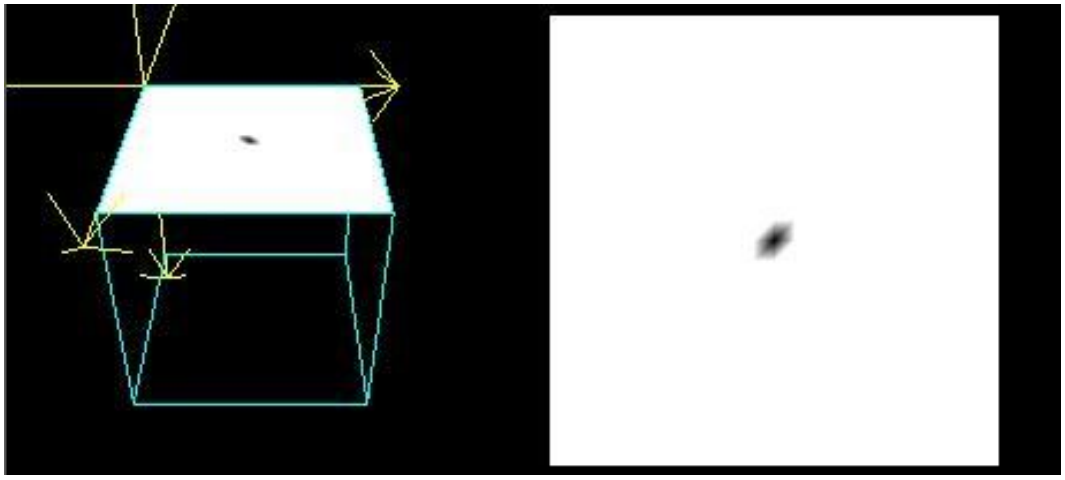

Figure 5.29: Point source: max 1 mg/L, time step 1, cutting plane 0 (Z).

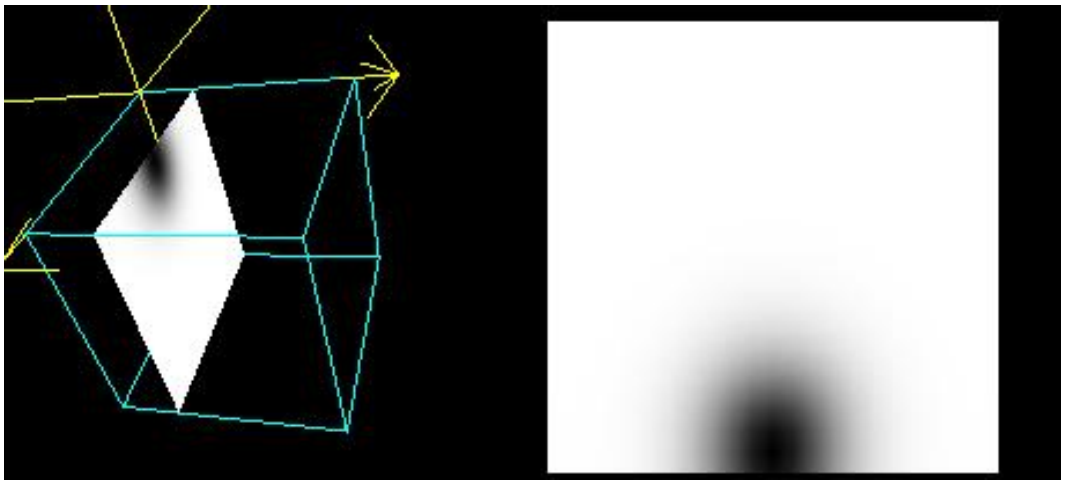

Figure 5.30: Point source: max 0.0015 mg/L, time step 10, cutting plane 6 (X).

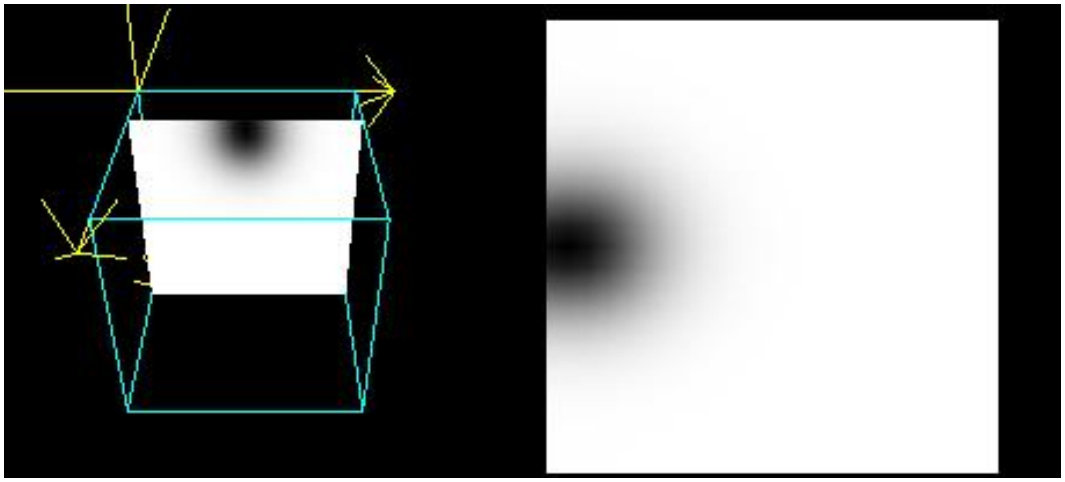

Figure 5.31: Point source: max 0.0015 mg/L, time step 10, cutting plane 6 (Y).

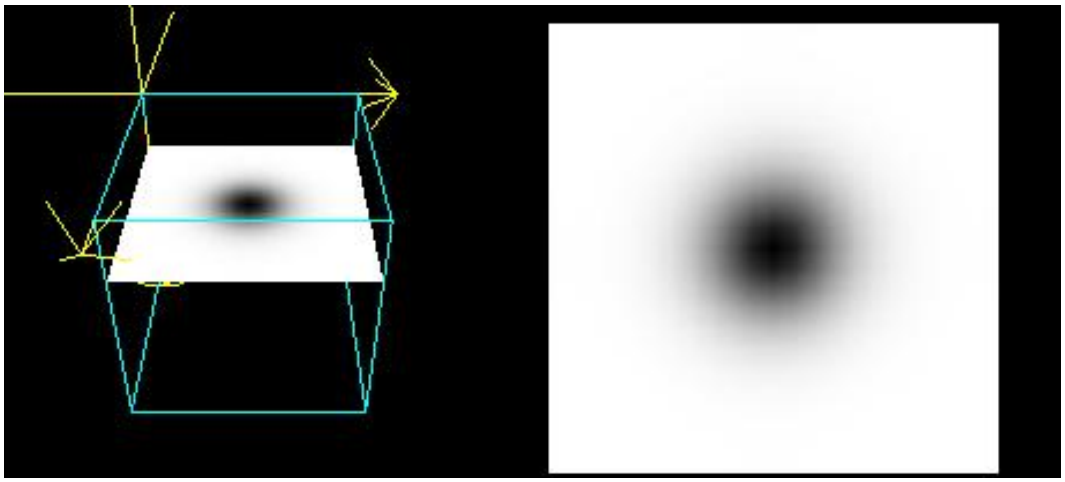

Figure 5.32: Point source: max 0.0032 mg/L, time step 10, cutting plane 6 (Z).

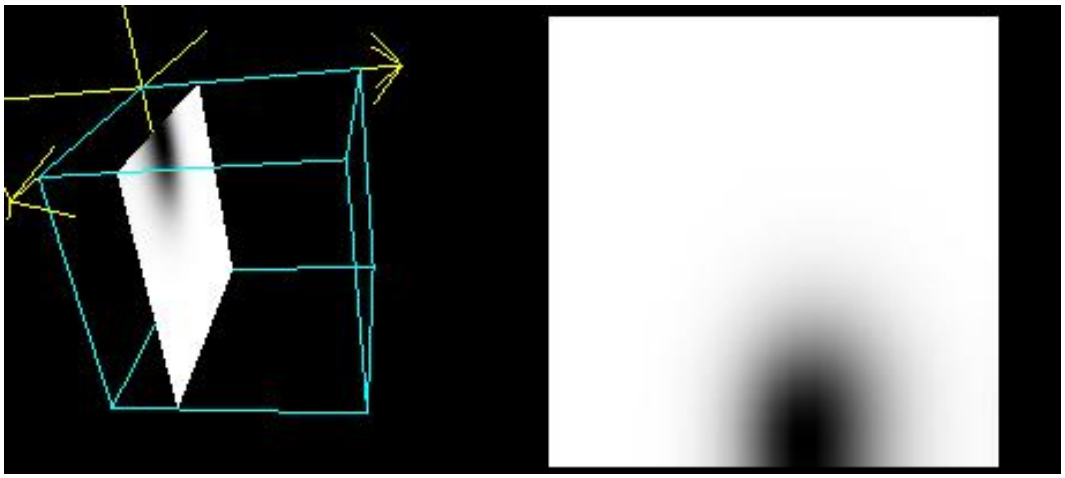

Figure 5.33: Point source: max 0.0002 mg/L, time step 3, cutting plane 8 (X).

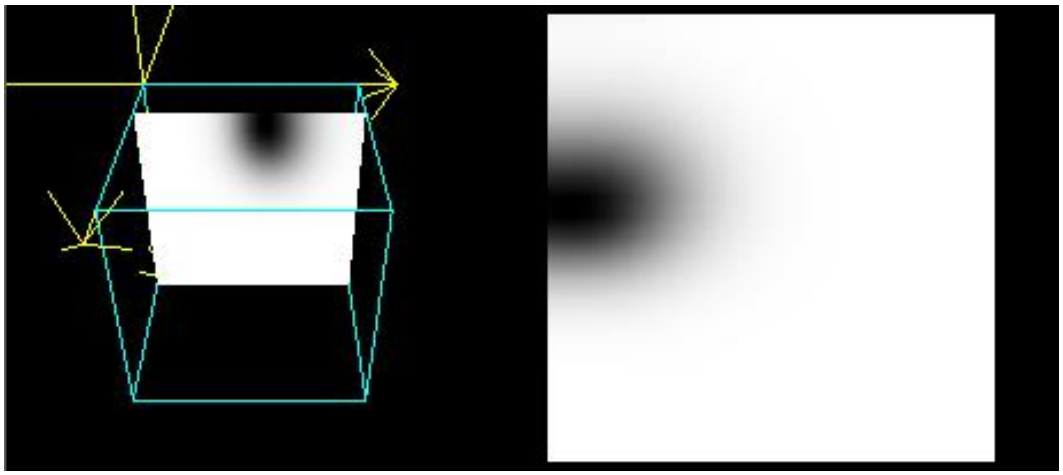

Figure 5.34: Point source: max 0.0001 mg/L, time step 3, cutting plane 6 (Y).

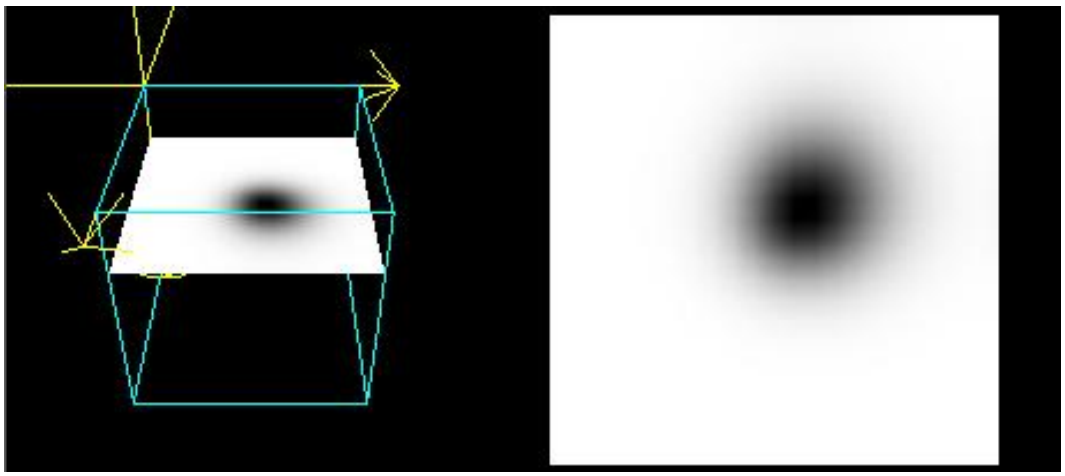

Figure 5.35: Point source: max 0.0009 mg/L, time step 3, cutting plane 6 (Z).

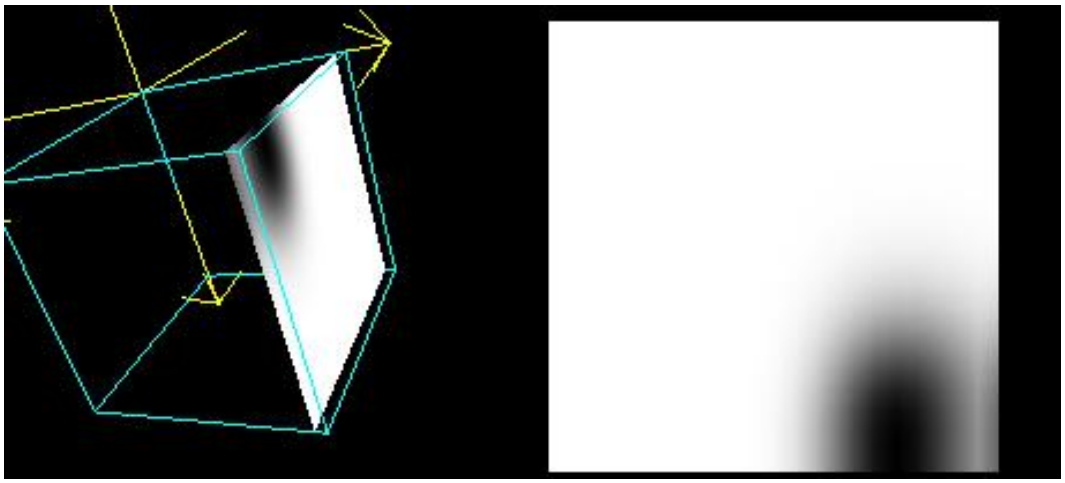

Figure 5.36: Point source: max 0.0014 mg/L, time step 10, cutting plane 21 (X).

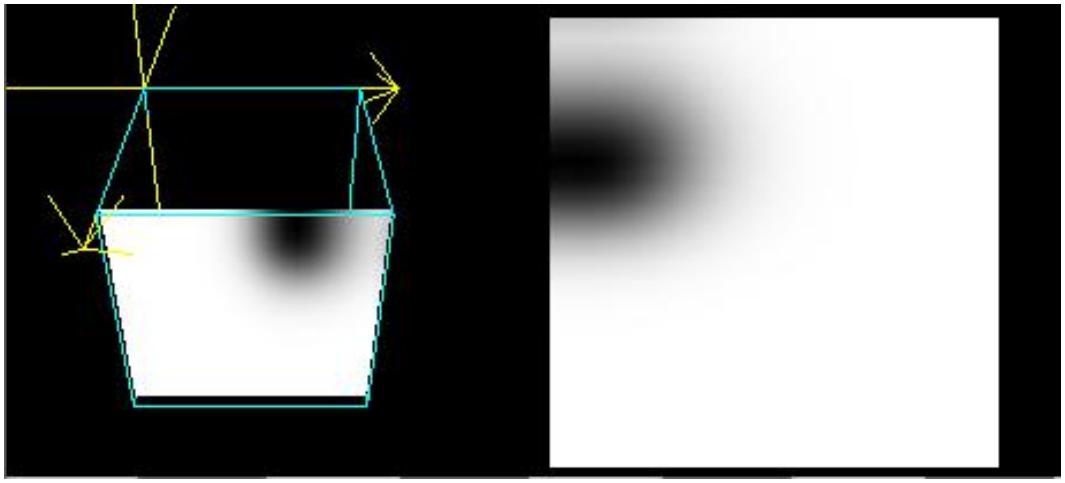

Figure 5.37: Point source: max 0.0043 mg/L, time step 10, cutting plane 21 (Y).

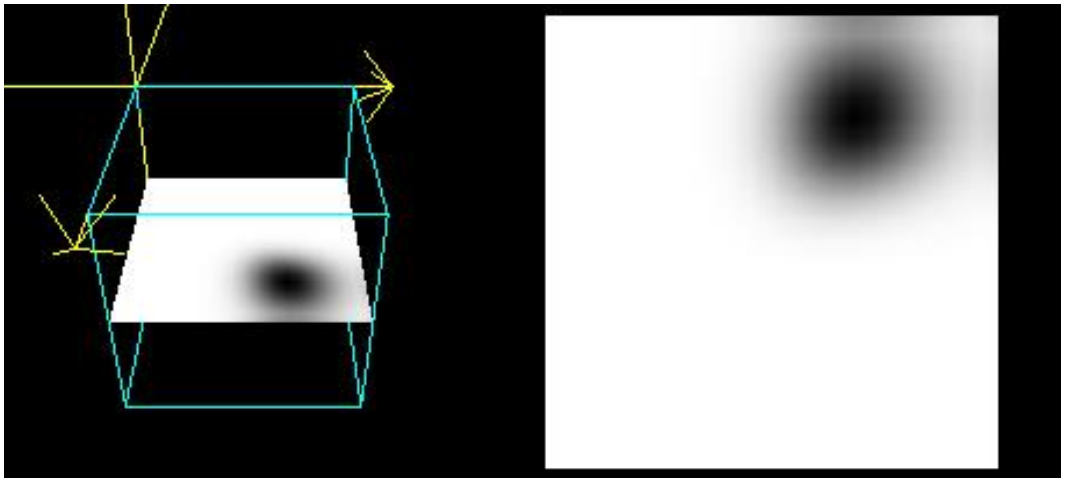

Figure 5.38: Point source: max 0.0005 mg/L, time step 10, cutting plane 11 (Z).

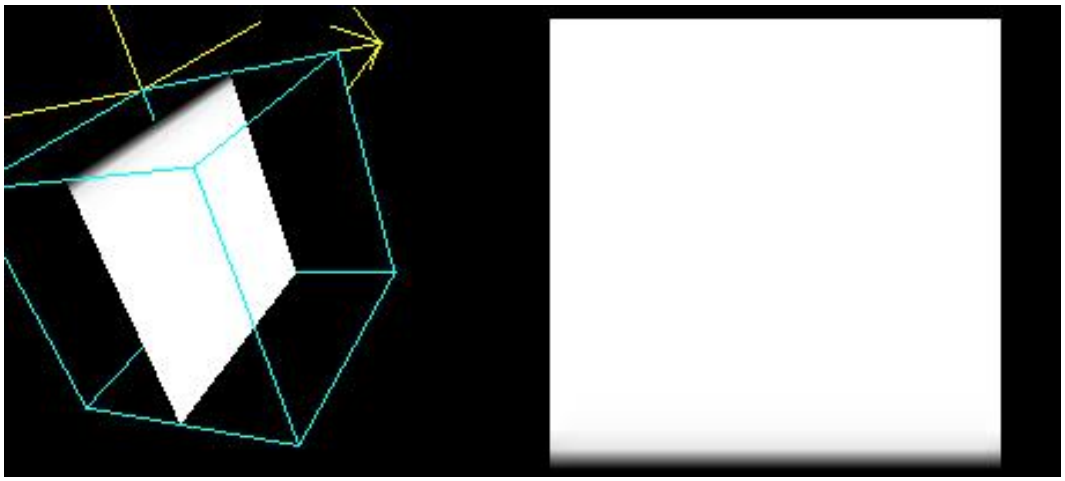

Figure 5.39: Line source: max 1 mg/L, time step 1, cutting plane 11 (X).

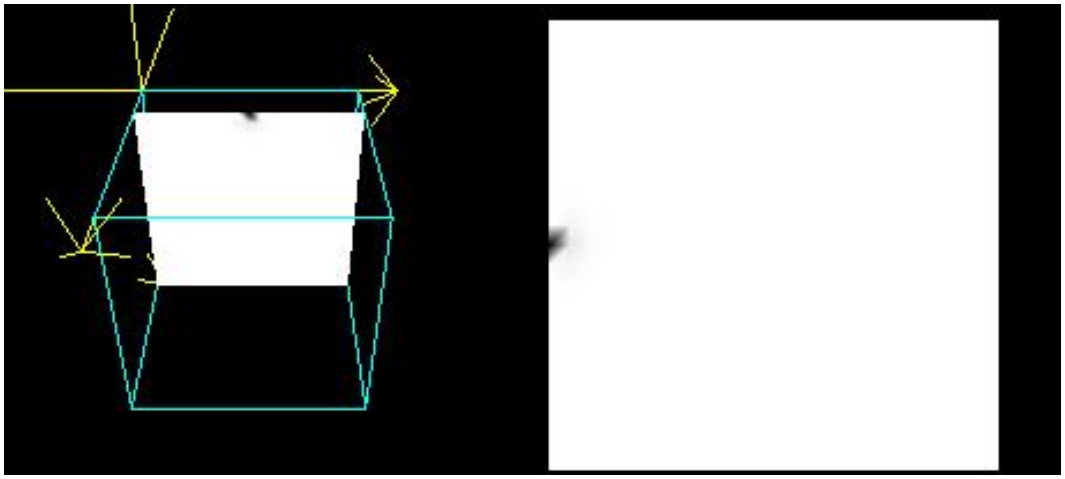

Figure 5.40: Line source: max 1 mg/L, time step 1, cutting plane 5 (Y).

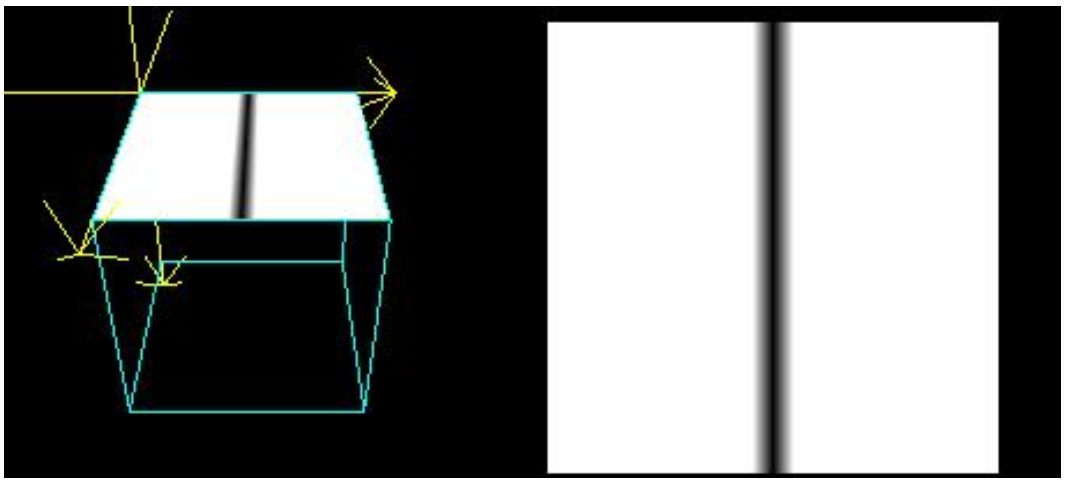

Figure 5.41: Line source: max 1 mg/L, time step 1, cutting plane 0 (Z).

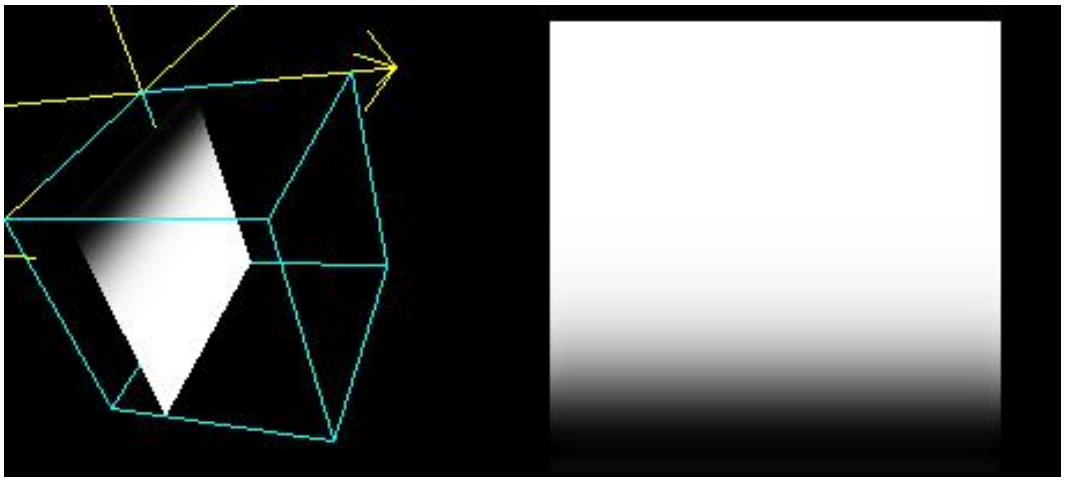

Figure 5.42: Line source: max 0.0088 mg/L, time step 10, cutting plane 6 (X).

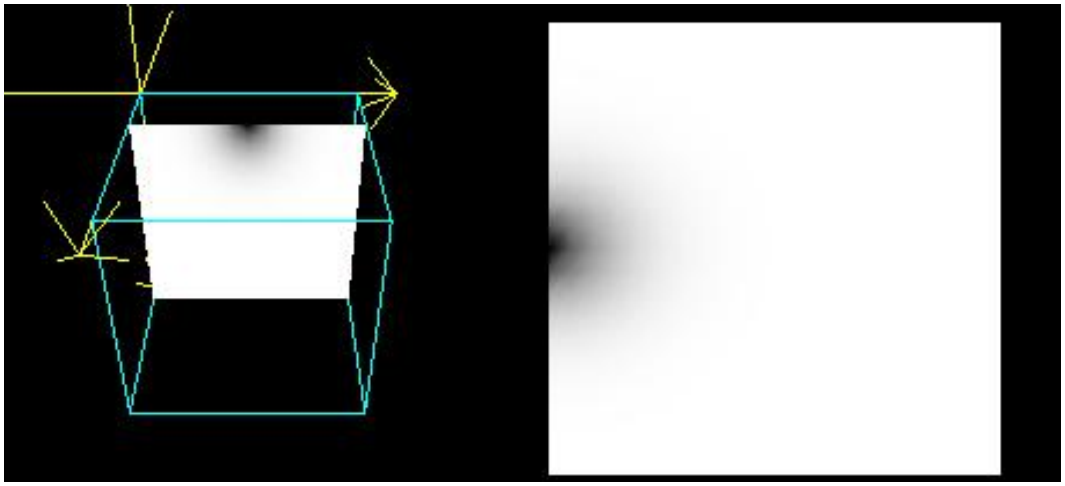

Figure 5.43: Line source: max 0.2359 mg/L, time step 10, cutting plane 6 (Y).

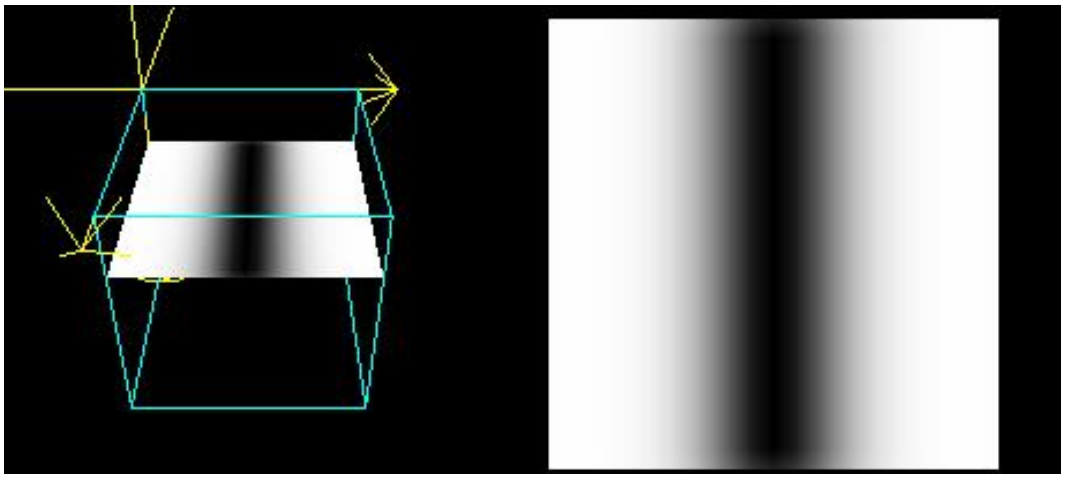

Figure 5.44: Line source: max 0.0181 mg/L, time step 10, cutting plane 6 (Z).

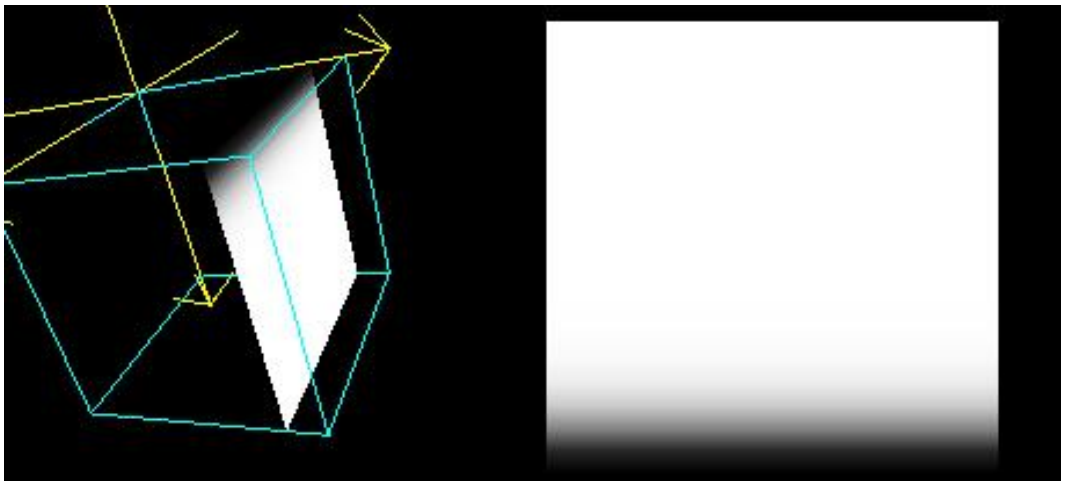

Figure 5.45: Line source: max 0.0003 mg/L, time step 2, cutting plane 17 (X).

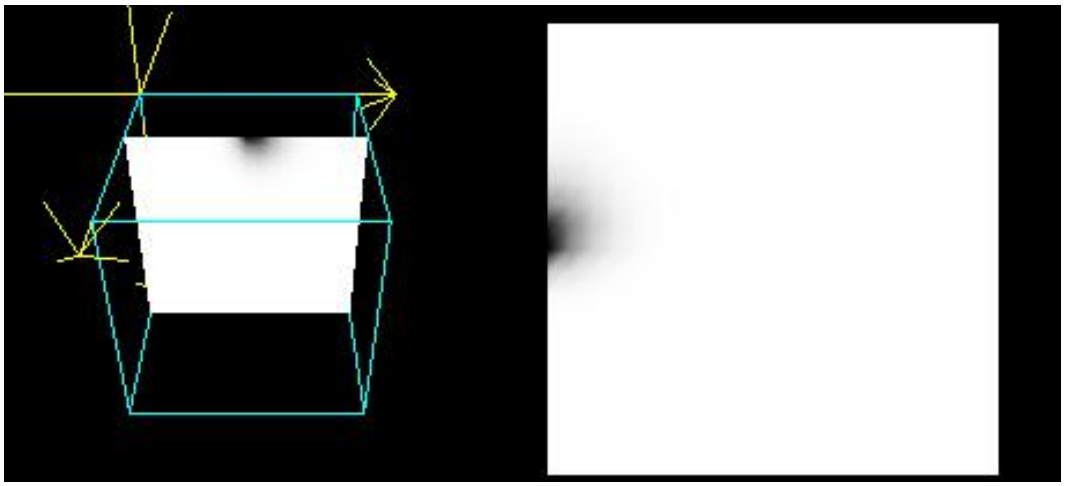

Figure 5.46: Line source: max 0.1783 mg/L, time step 2, cutting plane 8 (Y).

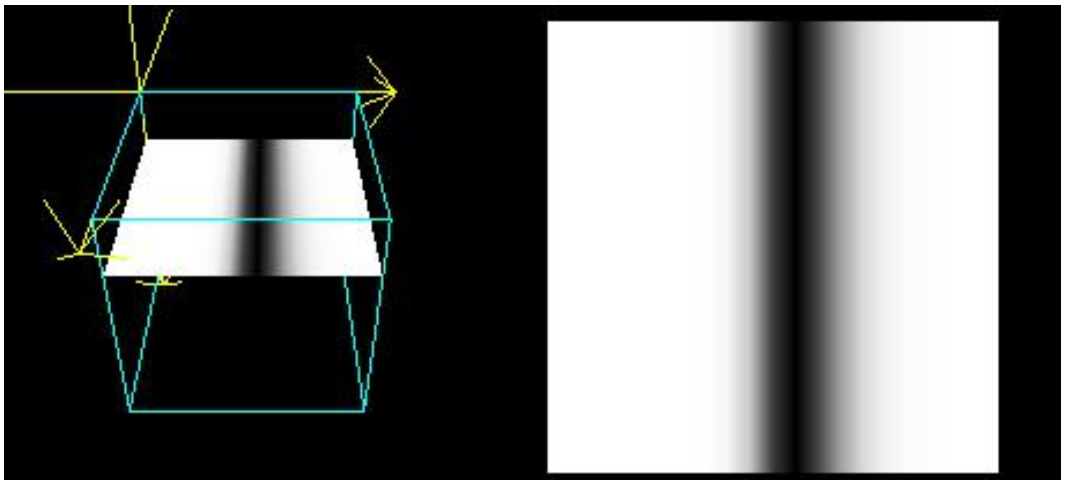

Figure 5.47: Line source: max 0.0014 mg/L, time step 2, cutting plane 5 (Z).

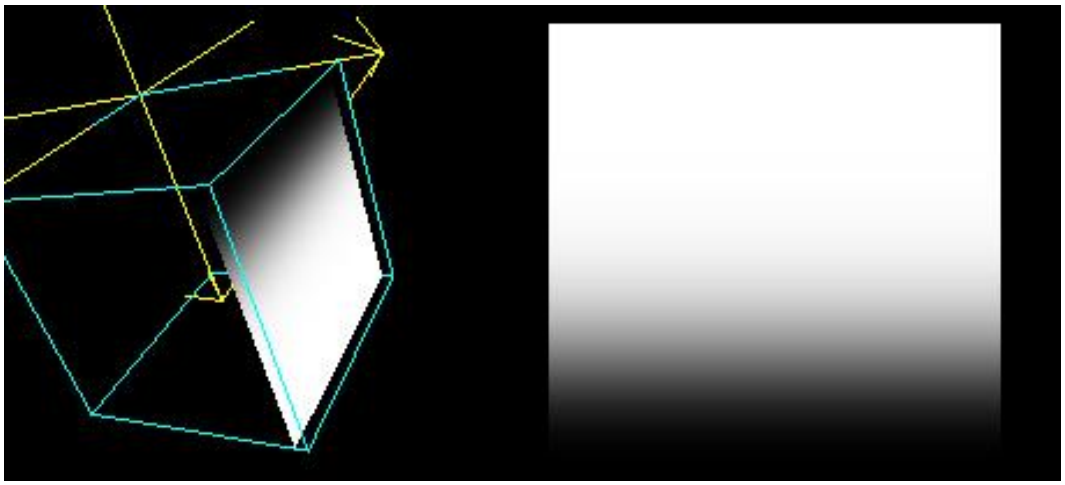

Figure 5.48: Line source: max 0.0156 mg/L, time step 10, cutting plane 19 (X).

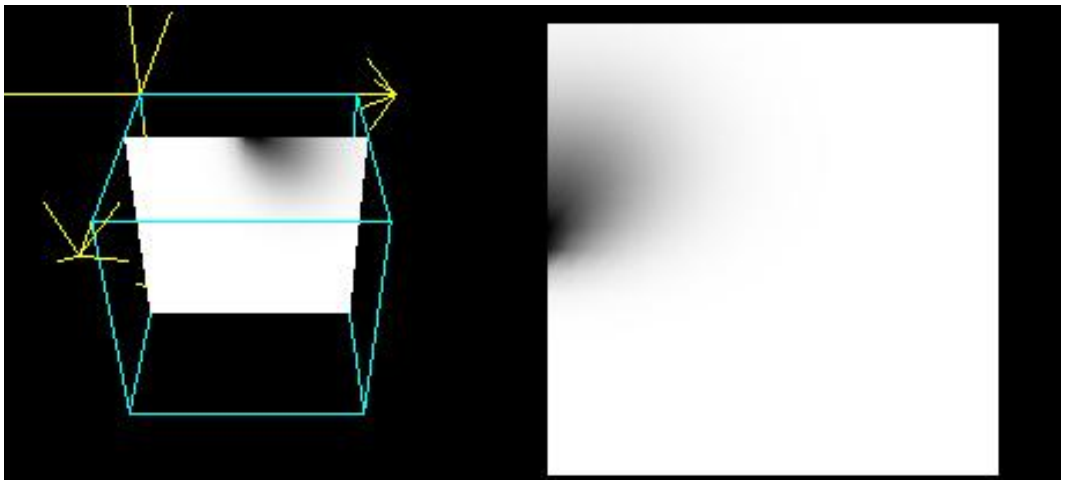

Figure 5.49: Line source: max 0.2483 mg/L, time step 10, cutting plane 8 (Y).

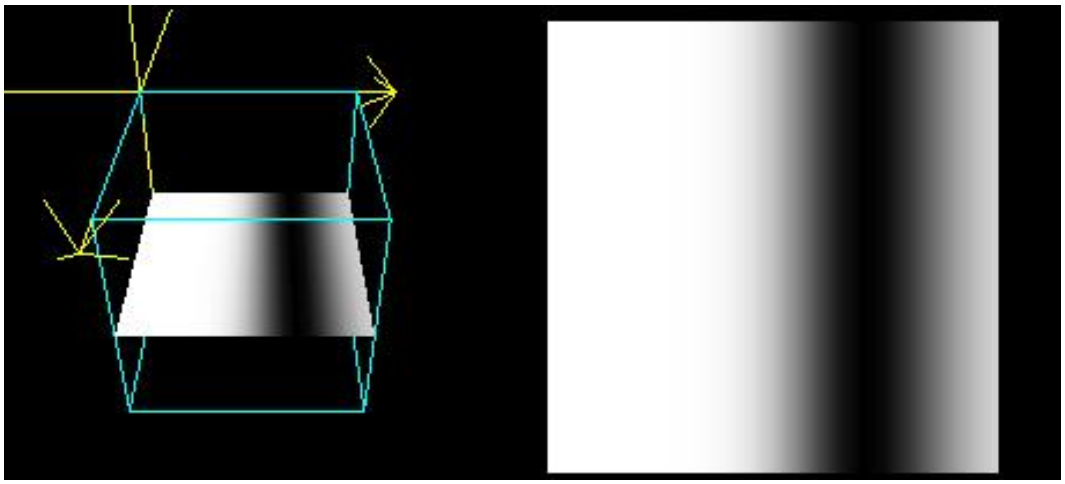

Figure 5.50: Line source: max 0.0015 mg/L, time step 10, cutting plane 11 (Z).

5.42 to 5.44 show that the concentration loads are distributed symmetrically around the line source, because the advection in X and Y directions are set null. Figures 5.45 to 5.50 show that the concentration loads move smoothly from the center to right, due to the effect from advection in X and Y directions. The migration of the contaminant concentration for the line source has a wider distribution in the horizontal (Z) and transversal (X) cutting planes. The contaminant concentration just meets the CCM and CCC criteria at cutting plane 5 in Z direction at time step 2 (0.0014 mg/L in Figure 5.47), while violates both criteria at cutting plane 11 in Z direction at time step 10 (0.0015 mg/L in Figure 5.50). This means that the solute has traveled deeper into the soil.

The simulation results for the two-point source (Figures 5.51 to 5.62), with the same anisotropic media and isotropic dispersion, are similar to the one-point source case in the longitudinal and horizontal cutting planes. At the top (Figures 5.51 to 5.53), the maximum concentration is clearly equal to the input source concentration ($1mg/L$ Hg). Figures 5.54 to 5.56 show that the concentration loads are distributed symmetrically around the two point sources, because the advection in X and Y directions are set null. Figures 5.57 to 5.62 show that the concentration loads move smoothly from the center to corner, due to the effect from advection in X and Y directions. The migration of the contaminant concentration for the two-point sources also shows a wider distribution than the one-point case.

5.4 Three-dimensional MRTM Simulation for Macro-porosity Soil Applications

The 3D-MRTM model is also modified according to the algorithm detailed in [72], so as to estimate the transport of contaminants in soils that present macro-porosity, i.e., conduits that are due to cracks in the soil continuum that promotes preferential flow through the soil. It is a first approach for modeling chemical species transport due to non-uniform flow and transport, and demonstrates the advantages of the three-dimensional MRTM model over the one-dimensional model.

In order to compare the results, the experiments for chemical transport in soils with macro-porosity follow the similar scenario in Section 5.4. The contaminant chosen for this application is

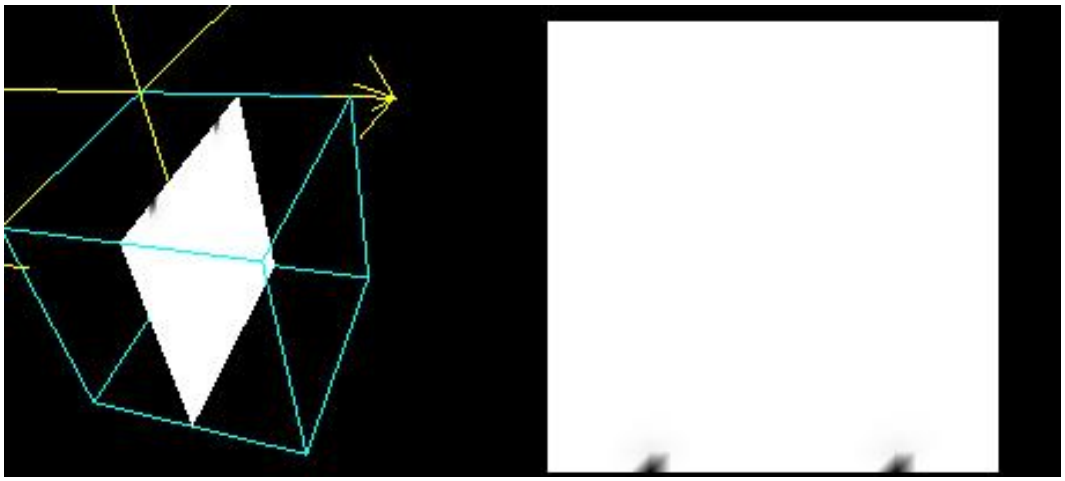

Figure 5.51: Two-point source: max 1 mg/L, time step 1, cutting plane 11 (X).

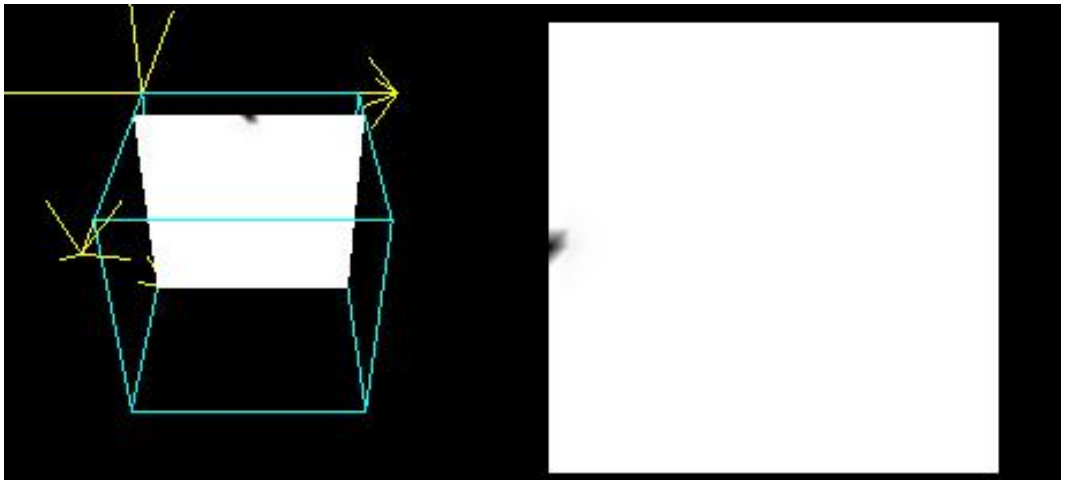

Figure 5.52: Two-point source: max 1 mg/L, time step 1, cutting plane 5 (Y).

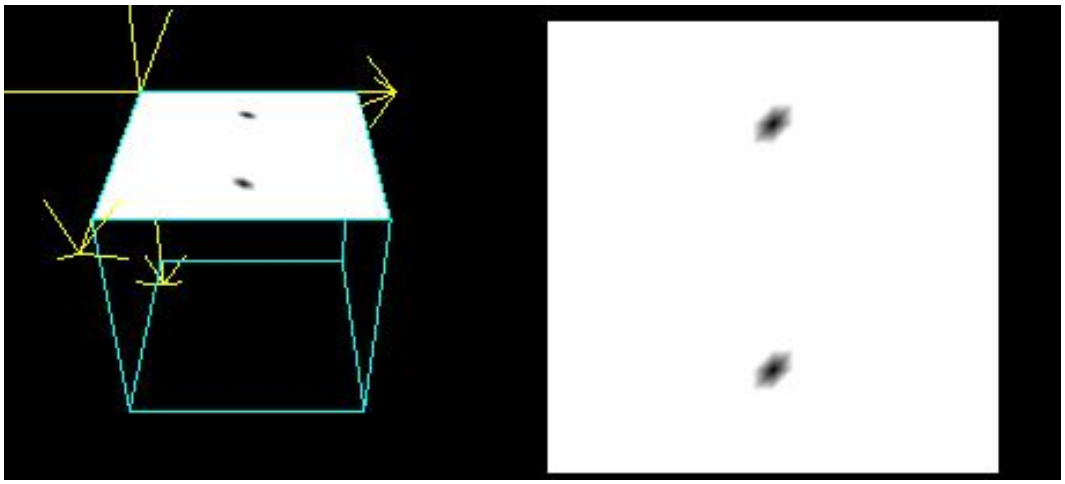

Figure 5.53: Two-point source: max 1 mg/L, time step 1, cutting plane 0 (Z).

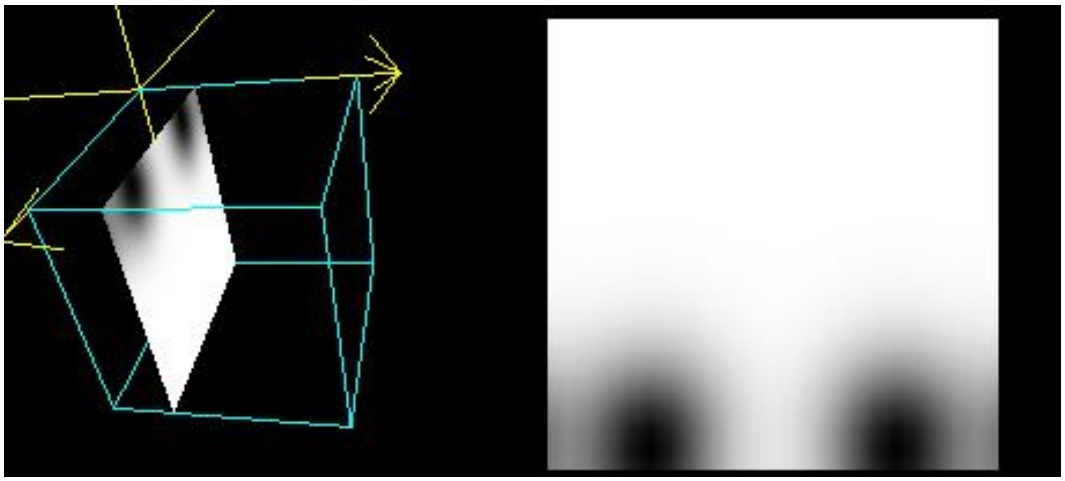

Figure 5.54: Two-point source: max 0.0016 mg/L, time step 10, cutting plane 6 (X).

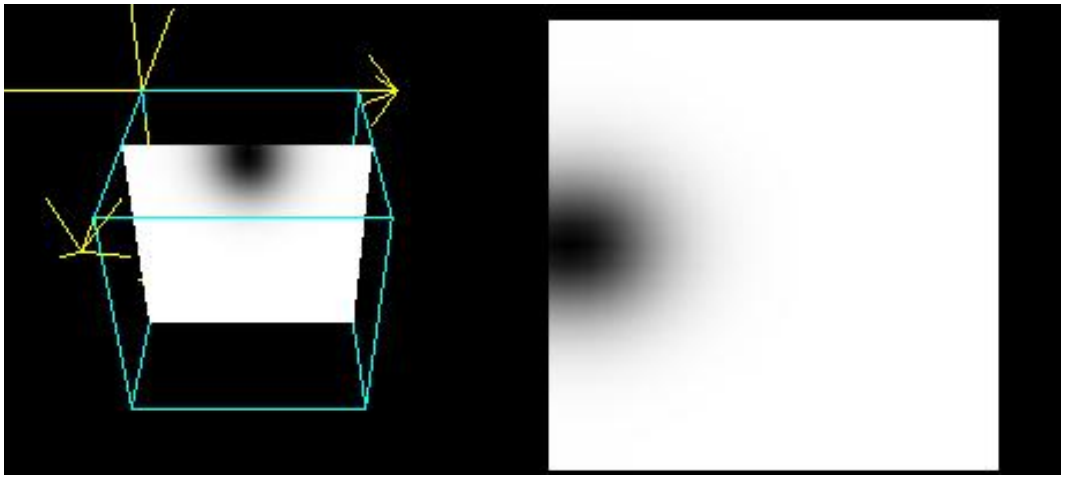

Figure 5.55: Two-point source: max 0.0012 mg/L, time step 10, cutting plane 11 (Y).

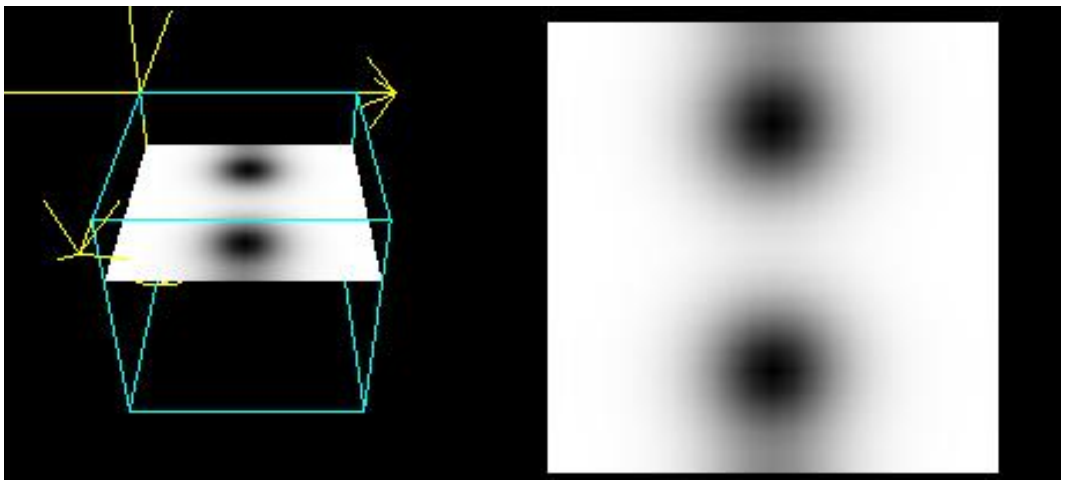

Figure 5.56: Two-point source: max 0.0032 mg/L, time step 10, cutting plane 6 (Z).

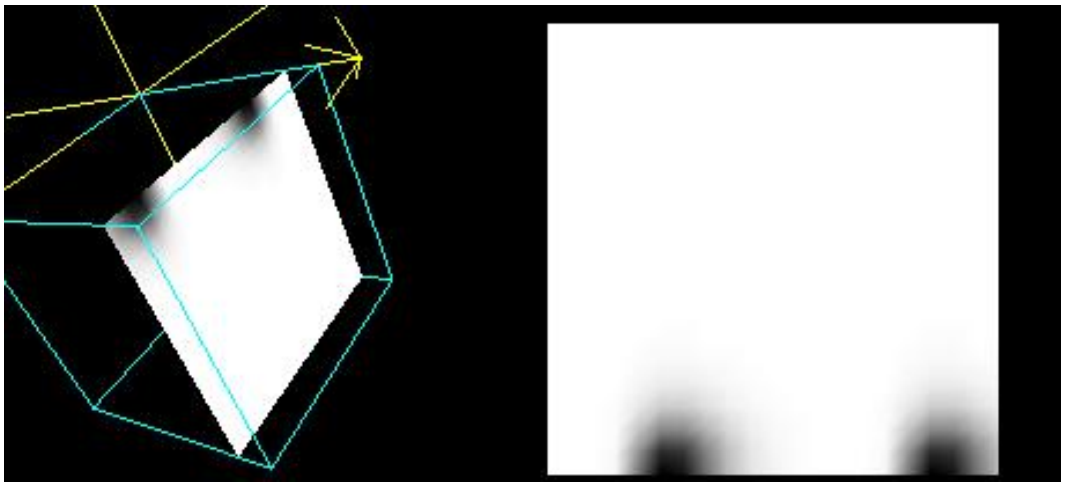

Figure 5.57: Two-point source: max 0.0001 mg/L, time step 2, cutting plane 17 (X).

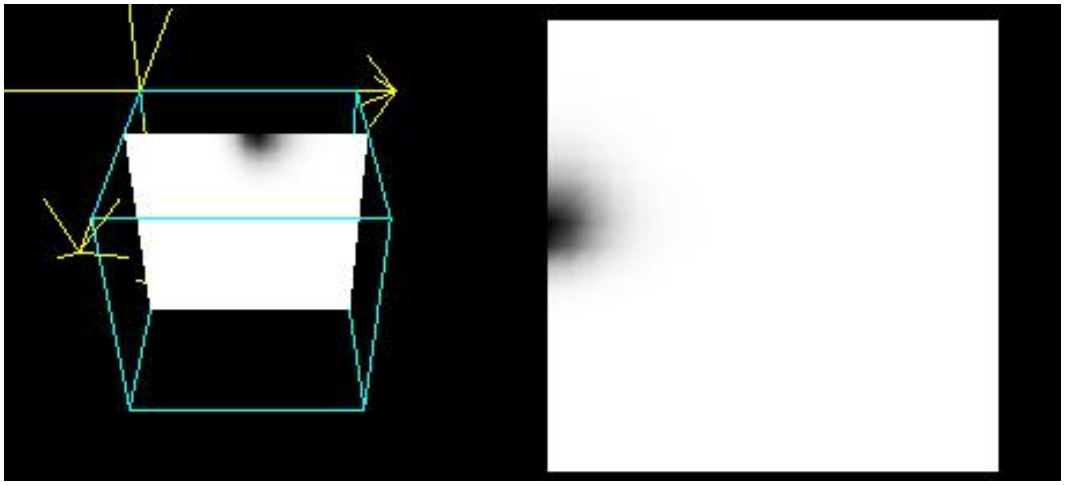

Figure 5.58: Two-point source: max 0.0056 mg/L, time step 2, cutting plane 8 (Y).

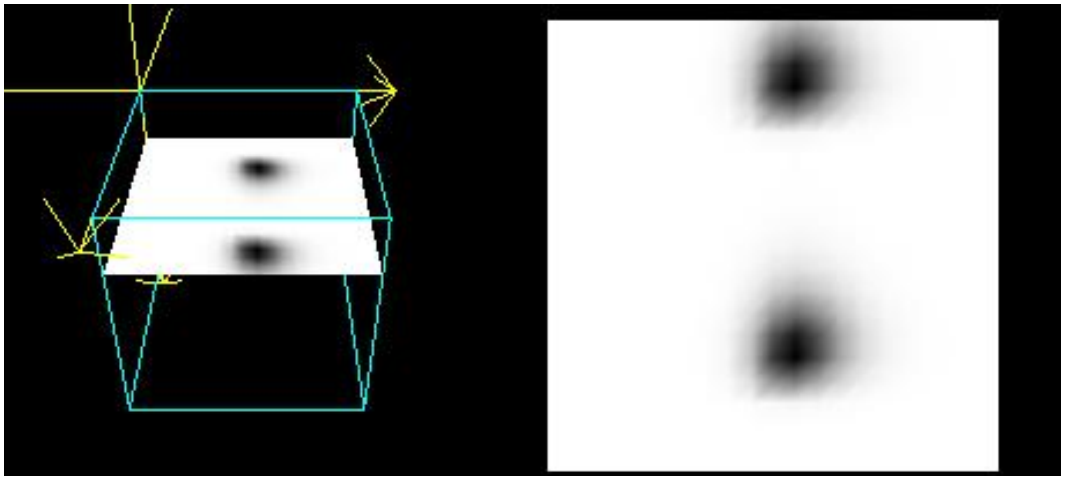

Figure 5.59: Two-point source: max 0.0004 mg/L, time step 2, cutting plane 5 (Z).

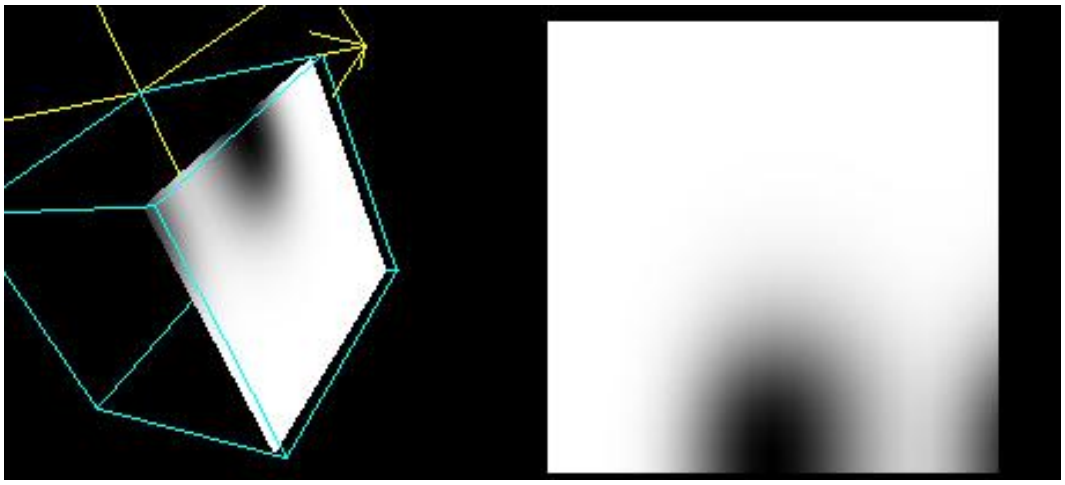

Figure 5.60: Two-point source: max 0.0023 mg/L, time step 10, cutting plane 19 (X).

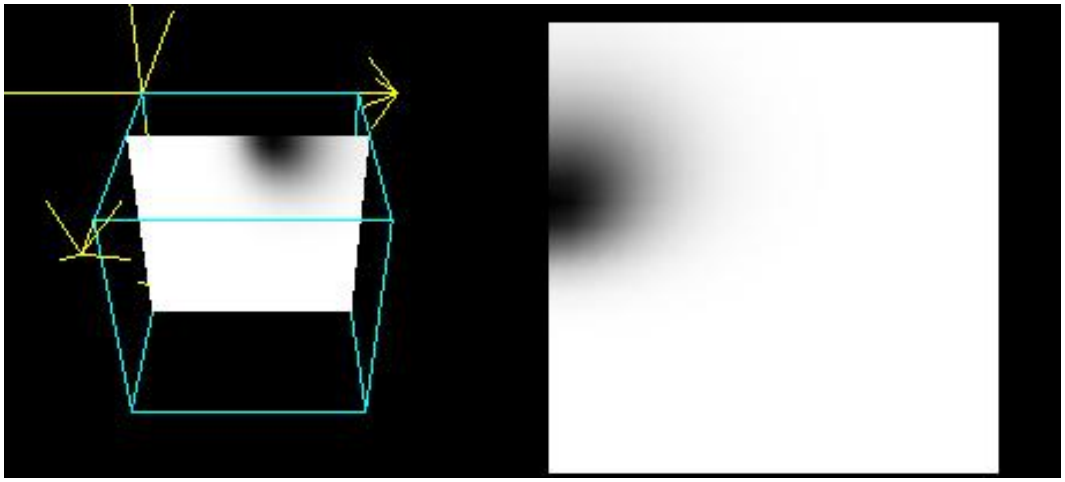

Figure 5.61: Two-point source: max 0.0263 mg/L, time step 10, cutting plane 8 (Y).

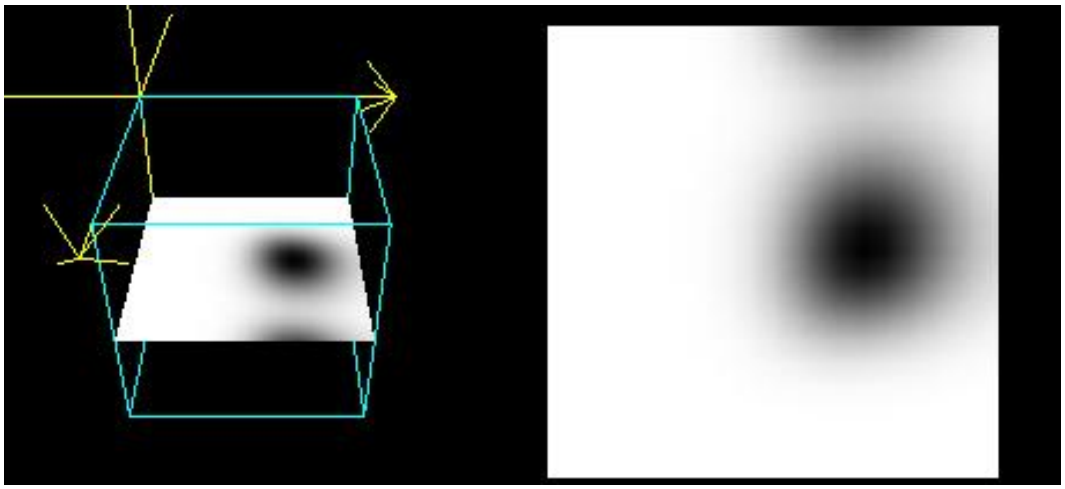

Figure 5.62: Two-point source: max 0.0002 mg/L, time step 10, cutting plane 11 (Z).

mercury (Hg), and line-source contaminant load is applied on the top surface of the soil column. The column consists of a contaminant-free soil of 1 cm depth, having a volumetric moisture content of 0.4 cm^3/cm^3 and a bulk density of 1.25 g/cm^3. It receives a load of $1mg/L$ of Hg in solution during 1 hour. The dispersion coefficient of the aqueous solution in the soil is anisotropic: D_x = 0.01 cm^2/hr, D_y = 0.01 cm^2/hr, and D_z = 0.05 cm^2/hr. The Darcy flux (cm/hr) is also anisotropic: q_x = 0.1 cm/hr, q_y = 0.15 cm/hr, q_z = 0.1 cm/hr. The anisotropic dispersion and convection coefficients will cause asymmetric concentration distribution on the result cutting planes in different directions.

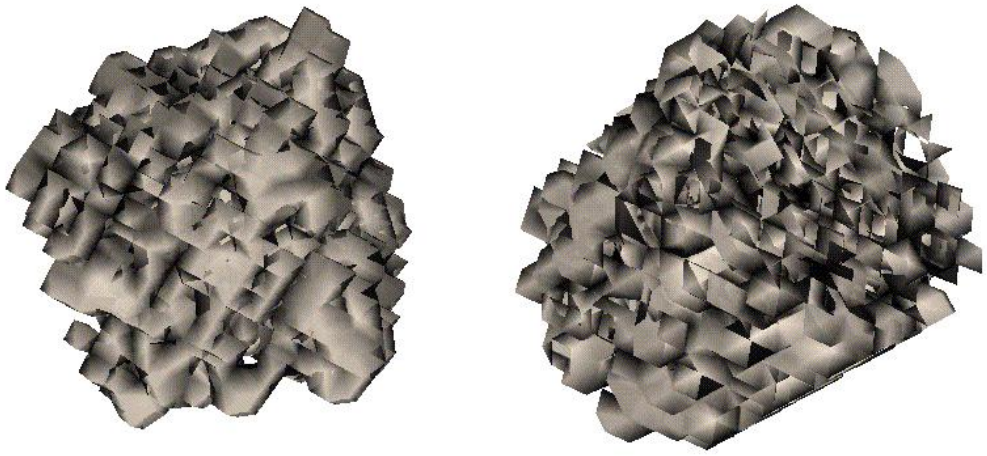

Figure 5.63: Conduits in the soil column.

The macro-porous property are shown in Figure 5.63 with the conduits due to the cracks in soils. The compound-soil interaction is reflected in the following parameters: k_d = 1.0 cm^3/g (distribution coefficient), NEQ = 1.1 (Freundlich parameter), k_1 = 0.01 hr^{-1} (forward kinetic reaction rate), k_2 = 0.02 hr^{-1} (backward kinetic reaction rate), U = 1.2 (non-linear kinetic parameter), k_3 = 0.01 hr^{-1} (forward kinetic reaction rate), k_4 = 0.02 hr^{-1} (backward kinetic reaction rate), W = 1.3 (non-linear kinetic parameter), k_5 = 0.01 hr^{-1} (forward kinetic reaction rate), k_6 = 0.02 hr^{-1} (backward kinetic reaction rate), k_s = 0.005 (irreversible reaction rate).

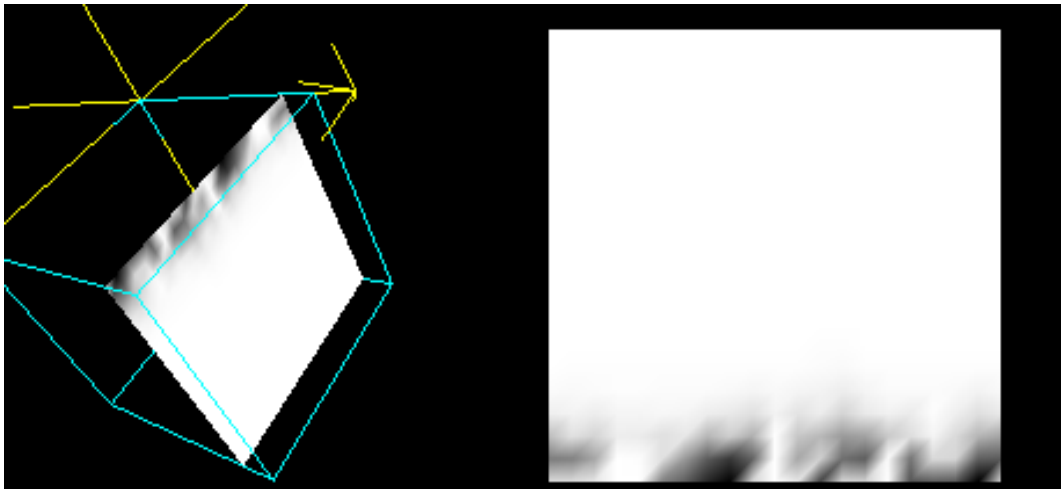

Figure 5.64: Macro-porosity: max 0.0013 mg/L, time step 2, cutting plane 17 (X).

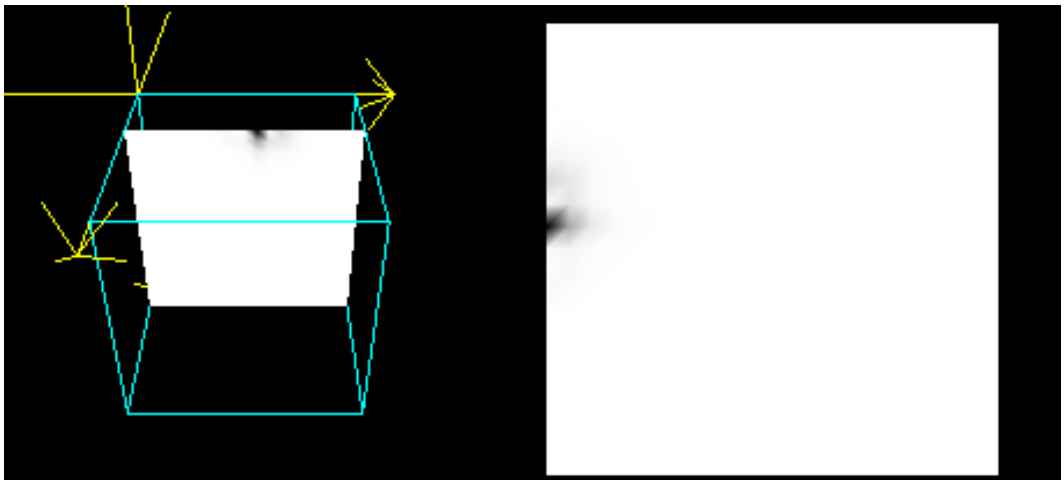

Figure 5.65: Macro-porosity: max 0.5381 mg/L, time step 2, cutting plane 7 (Y).

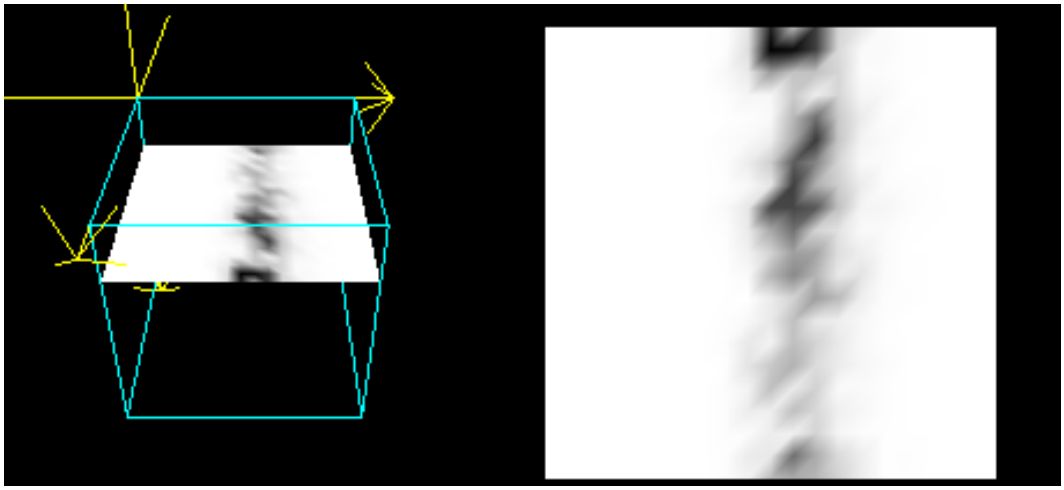

Figure 5.66: Macro-porosity: max 0.0037 mg/L, time step 2, cutting plane 5 (Z).

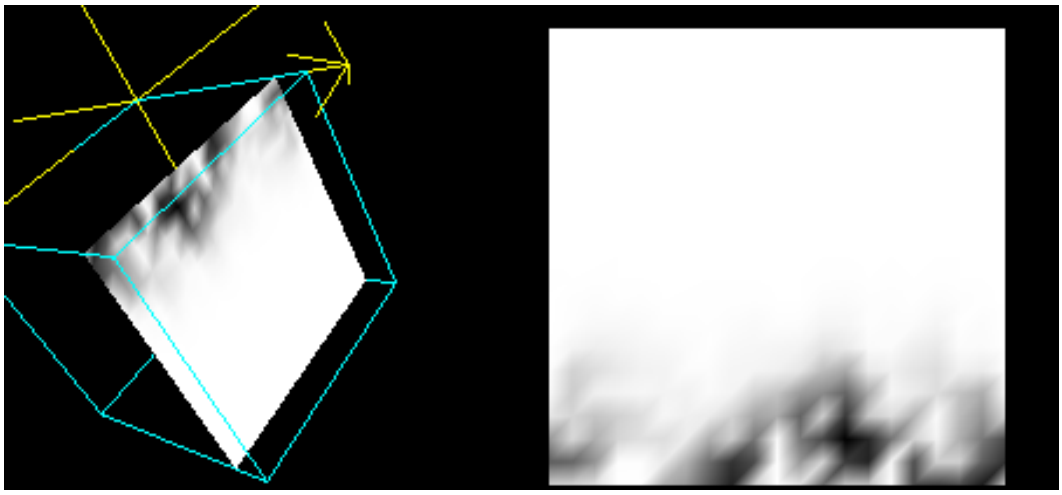

Figure 5.67: Macro-porosity: max 0.0625 mg/L, time step 10, cutting plane 17 (X).

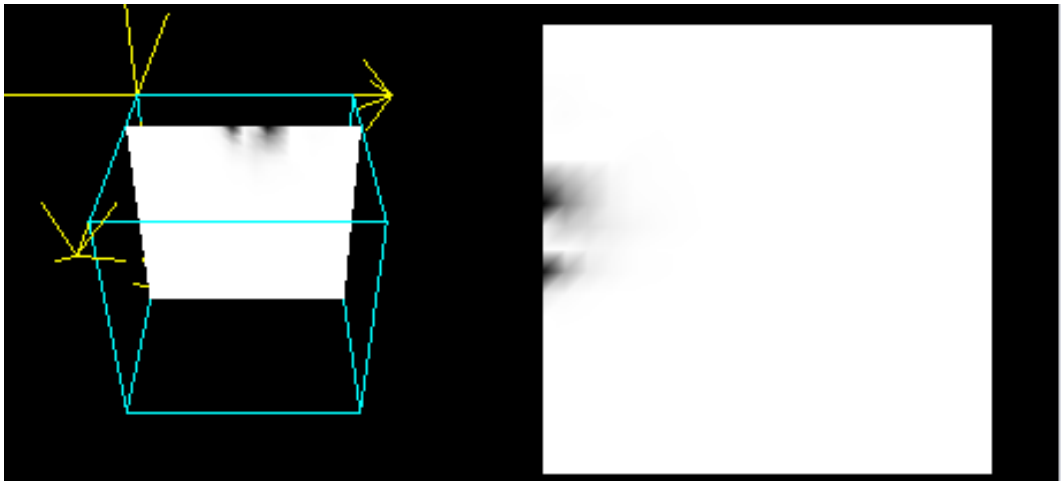

Figure 5.68: Macro-porosity: max 0.8167 mg/L, time step 10, cutting plane 6 (Y).

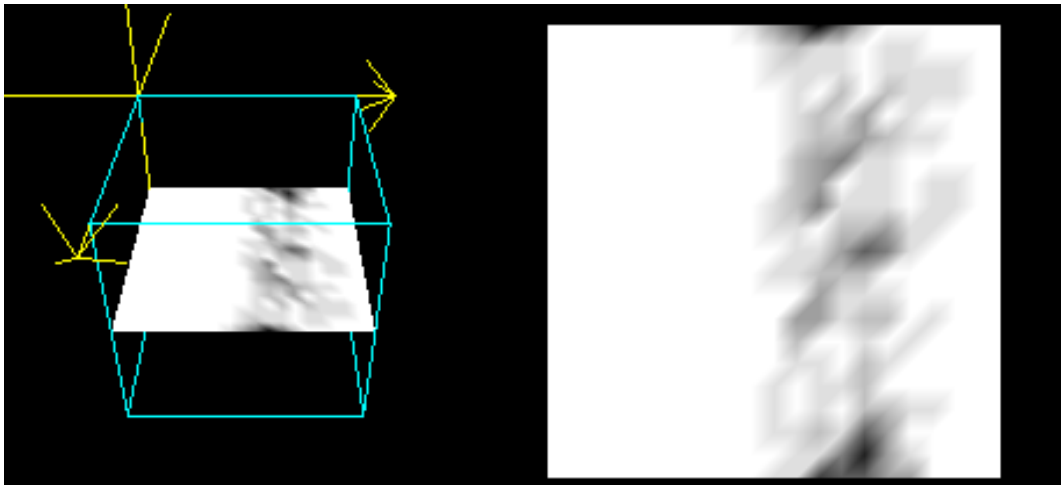

Figure 5.69: Macro-porosity: max 0.0072 mg/L, time step 10, cutting plane 8 (Z).

The spatial size of the grid used for all simulations was $23 \times 23 \times 23$, and the time domain is divided into 10 steps. Cutting (sliding) planes from number 0 to 22 in three directions are used by the model to show the spatial distribution of the concentration in shades of gray (the maximum concentration corresponds to black). The 3D model provides the maximum concentration values and the position of the plane in X (transverse plane), Y (longitudinal plane) or Z (horizontal plane) directions. Figures 5.64 to 5.69 show the migration of contaminant in a soil with macro-porosity in two different time steps. Notice the difference with Figures 5.45 to 5.50 in Section 5.4 corresponding to a homogeneous soil. The contaminant travels faster through the macro-pores in the soil profile, and the movement is asymmetric due to the anisotropic dispersion and convection coefficients. The 3D-MRTM model considers both diffusion and convection effects in the chemical transport process, and improves the diffusion model for macro-porous soils detailed in [72].

In conclusion, the results provided by 3D-MRTM are consistent with the numerical output of 1D-MRTM, since the concentration-depth curves are shown to be similar for the nominal test case that is independent of temporal and spatial scales in Section 5.2. Besides the numerical output that the model generates, the visualization component of the model gives an almost instantaneous look into the spatial distribution of the contaminant. This visualization is made by sliding three planes (horizontal, longitudinal and transversal) across the whole simulation domain. The visualization of the concentrations is scaled from 0.0 to the maximum values so that the trace concentrations can be easily visualized. The numerical value of the maximum concentration is also output in the visualization window, together with the current position of the visualizing plane. The 3-D visualization is shown to be very useful for identifying the extent and severity of the soil contamination due to the trace compound load under three different types of input load distribution (point, line, and two-point sources). The quantification and visualization could be used for remediation purposes. Models of fewer dimensions might not provide comparable information especially on the spatial distribution of the contaminant.

The integrated web-based system greatly reduces the time used by traditional ways to calculate and visualize the simulation results. The database management system also persists the previous computational result data for future comparison. When there is a need to repeatedly run simulation and view results, the web-based environment can show great advantages to save time, since the users don't need to switch between computational programs and visualization programs. The web-based visualization and computational job management system integration provides a flexible and smooth access to the remote high performance computing resources.

CHAPTER VI

CONCLUSION

A three-dimensional multi-reaction transport model (MRTM) is developed in this dissertation to simulate the chemical retention and transport in groundwater and soil. This simulation provides an alternative way to trace contaminant movement outside laboratories. The alternating direction implicit (ADI) method is improved to achieve the second-order accuracy in both time and space with Neumann boundary conditions. Incremental parallel ADI method using OpenMP technologies is also successfully implemented in the shared memory environment with ideal speedup results.

The web-based visualization component of the simulation system gives an almost instantaneous look into the spatial distribution of the contaminant, and is very useful for identifying the extent and severity of the soil contamination due to the trace compound load under different types of input load distribution.

The 3-D MRTM web-based simulation system sets up a generic framework for high performance computing applications using the J2EE enterprise technologies. Platform independence, web accessibility, and multi-tiered architecture provide a thin front-end to the back-end high performance computing resources. Given access to the Internet, users can create and execute their own computing jobs using adequate computational resources anywhere anytime, even from a laptop personal computer. The web tier and business tier, often called middle-tier together, have the responsibility to allocate computing resources and manage result data. As the system grows, the multi-tier design also has the advantage to allocate programming tasks more efficiently to different people with different skills.

REFERENCES

[1] Arthur G. Hornsby, R. Don Wauchope, and Albert E. Herner. *Pesticide Properties in the Environment*. Springer-Verlag, New York, 1996.

[2] A.S. Donigian and P.S.C. Rao. Overview of terrestrial processes and modeling. In S.C. Hern and S.M. Melancon, editors, *Vadose Zone Modeling of Organic Pollutants*, pages 3–35. Lewis Publishers, Inc., Chelsea, Michigan, 1986.

[3] K.H. Baker. Bioremediation of surface and subsurface soils. In K.H. Baker and D.S. Herson, editors, *Bioremediation*, pages 203–259. McGraw-Hill, Inc, New York, 1994.

[4] H.M. Selim and M. Amacher. *Reactivity and Transport of Heavy Metals in Soils*. CRC/Lewis Publishers, Boca Raton, FL, 1997.

[5] H.M. Selim, M.C. Amacher, and I.K. Iskandar. *Modeling the transport of heavy metals in soil*. U.S. Army Cold Regions Research and Engineering laboratory, Hanover, NH, 1990. Monograph 90-2.

[6] Feike J. Leij, Nobuo Toride, and Martinus Th. van Genuchten. Analytical solutions for non-equilibrium solute transport in three-dimensional porous media. *Journal of Hydrology*, 151(2-4):193–228, Nov 1993.

[7] Dmitry Rudakov and Victor C. Rudakov. Analytical modeling of aquifier pollution caused by solid waste depositories. *Ground Water*, 37(3):352–357, 1999.

[8] Geoffrey Deane, Zenitha Chroneer, and Wilbert Lick. Diffusion and sorption of hexachlorobenzene in sediments and saturated soil. *Journal of Environmental Engineering*, 125(8):689–696, 1999.

[9] A.L. Walter, E.O. Frind, D.W. Blowes, C.J. Ptacek, and J.W. Molson. Modeling of multicomponent reactive transport in groundwater: 1. model development and evaluation. *Water Resources Research*, 30(11):3137–3148, 1994.

[10] J. Islam, N. Singhal, and P. Jaffe. Modeling biogeochemical interactions in soils under leaking landfills. In *Proceedings of the 1999 contaminated site remediation conference*, Fremantle, Western Australian, 1999.

[11] H.B. Manguerra and L.A. Garcia. Modeling flow and transport in drainage areas with shallow ground water. *Journal of Irrigation and Drainage Engineering*, 123(3):185–193, May-Jun 1997.

[12] A. Yakirevich, V. Borisov, and S. Sorek. A quasi three-dimensional model for flow and transport in unsaturated and saturated zones: 1. implementation fo the quasi two-dimensional case. *Advances in Water Resources*, 21(8):679–689, Jul 1998.

[13] Genevieve Segol. Hopscotch algorithm for three-dimensional simulation. *Journal of Hydraulic Engineering*, 118(3):385–406, Mar 1992.

[14] Rajesh Srivastava and T.C. Jim Yeh. A three-dimensional numerical model for water flow and transport of chemically reactivesolute through porous media under variably saturated conditions. *Advances in Water Resources*, 15(5):275–287, 1992.

[15] L.F. Konikow, D.J. Goode, and G.Z. Hornberger. A three-dimensional method-of-characteristics solute-transport model (moc3d). *U.S. Geological Survey: Water-Resources Investigations Report 96-4267*, 1996.

[16] K.L. Kipp Jr., L.F. Konikow, and G.Z. Hornberger. An implicit dispersive transport algorithm for the u.s. geological survey moc3d solute-transport model. *U.S. Geological Survey: Water-Resources Investigations Report 98-4234*, 1998.

[17] C.I. Heberton, T.F. Russell, L.F. Konikow, and G.Z. Hornberger. A three-dimensional finite-volume eulerian-lagrangian localized adjoint method (ellam) for solute-transport modeling. *U.S. Geological Survey: Water-Resources Investigations Report 00-4087*, 2000.

[18] Honghai Zeng, Vladimir J. Alarcon, William Kingery, H. Magdi Selim, and Jianping Zhu. A web-based simulation system for transport and retention of dissolved contaminants in soil. *Computers and Electronics in Agriculture*, 33:105–120, 2002.

[19] J.W. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods.* Springer-Verlag, New York, 1995.

[20] Rohit Chandra, Leo Dagum, Dave Kohr, Dror Maydan, Jeff McDonald, and Ramesh Menon. *Parallel Programming in OpenMP*. Morgan Kaufmann Publishers, 2000.

[21] Paul C. Fife. *Mathematical Aspects of Reaction and Diffusion Systems.* Springer-Verlag, New York, 1979.

[22] C.V. Pao. *Nonlinear Parabolic and Elliptic Equations.* Plenum Press, New York, 1992.

[23] J.Smoller. *Shock Waves and Reaction-Diffusion Equations.* Springer-Verlag, New York, 1983.

[24] Michael E. Taylor. *Partial Differential Equations III: Nonlinear Equations.* Springer-Verlag, New York, 1996.

[25] G. Strang. On the construction of and comparison of difference schemes. *SIAM J. Numerical Analysis*, 5:506–517, 1968.

[26] G.D. Smith. *Numerical Solution of Partial Differential Equations.* Clarendon Press, Oxford, 2nd edition, 1978.

[27] W.F. Ames. *Numerical Methods for Partial Differential Equations.* Academic Press, New York, 2nd edition, 1977.

[28] G. Golub and J.M. Ortega. *Scientific Computing: An Introduction with Parallel Computing*. Academic Press, San Diego, CA, 1993.

[29] D.W. Peaceman and H.H. Rachford Jr. The numerical solution of parabolic and elliptic differential equations. *J. Society of Industrial and Applied Mathematics*, 3:28–41, 1955.

[30] Bertil Gustafsson, Heinz-Otto Kreiss, and Joseph Oliger. *Time Dependent Problems and Difference Methods.* John Wiley & Sons, Inc, Canada, 1995.

[31] R. Varga. *Matrix Iterative Analysis.* Prentics Hall, Englewood Cliffs, New Jersey, 1962.

[32] Jianping Zhu. *Solving Partial Differential Equations on Parallel Computers.* World Scientific Publishing Co. Pte. Ltd, 1994.

[33] T. Chan. Domain decomposition algorithms and computational fluid dynamics. *Int. J. Supercomputer Applications*, 2:72–83, 1988.

[34] S. Johnsson, Y. Sadd, and M. Schultz. Alternating direction methods on multiprocessors. *SIAM J. Sci. Stat. Comput.*, 8(5):686–700, 1987.

[35] J.M. Ortega and R.G. Voigt. Solution of partial differential equations on vector and parallel computers. *SIAM Review*, 27:149–240, 1985.

[36] D.W. Peaceman. *Fundamentals of Numerical Reservior Simulation.* Elsevier Scientific Publishing B.V, Amsterdam, 1977.

[37] F. Saied, C.T. Ho, L. Johnsson, and M. Schultz. Solving schrodinger's equation on the intel ipsc by the alternating direction method. *Hypercube Multiprocessors*, pages 680–691, 1987.

[38] A.H. Sameh, S. Chen, and D. Kuck. Parallel poisson and biharmonic solvers. *Computing*, 17:219–230, 1976.

[39] Kathy Walrath and Mary Campione. *The JFC Swing Tutorial: A Guide to Constructing GUIs*. Addison-Wesley, New York, 1999.

[40] Rob Gordon, Robert Gordon, and Alan McClellan. *Essential Jni: Java Native Interface (Essential Java).* Prentice Hall, NJ, 1998.

[41] Liang Sheng. *Java Native Interface: Programmer's Guide and Specification.* Addison-Wesley, New York, 1999.

[42] Jess Garms and Daniel Somerfield. *Professional Java Security.* Wrox Press Inc, first edition, 2001.

[43] Deepak Alur, John Crupi, and Dan Malks. *J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall, NJ, 2001.

[44] Jamie Jaworski. *Java 2 Platform Unleashed.* Sams, 1999.

[45] S.R. Ladd. *Java Algorithms.* McGraw-Hill, New York, 1998.

[46] Bruce Eckel. *Thinking in Java*. Prentice Hall, 1998.

[47] Jonathan Knudsen. *Java 2D Graphics.* O'Reilly, 1999.

[48] Larry R. Nyhoff and Sanford Leestma. *FORTRAN 77 for Engineers and Scientists with an Introduction to FORTRAN 90.* Prentice Hall, NJ, 4th edition, 1995.

[49] Scott Oaks. *Java Security.* O'Reilly, 2nd edition, 2001.

[50] Paul J. Perrone and Venkata R. Chaganti. *Building Java Enterprise Systems with J2EE*. SAMS, Indianapolis, Indiana, 2000.

[51] Mary E.S. Morris and Randy J. Hinrichs. *Web Page Design: A Different Multimedia*. Prentice Hall, NJ, first edition, 1996.

[52] Aaron E. Walsh and Doug Gehringer. *Java 3D API Jump-Start.* Prentice Hall, NJ, 2002.

[53] Marty Hall. *Core Servlets and JavaServer Pages.* Prentice Hall, NJ, 2000.

[54] David M. Geary. *Advanced JavaServer Pages*. Prentices Hall, NJ, 2001.

[55] *Distributed Java 2 Platform Database Development.* Prentice Hall, NJ, first edition, 2001.

[56] Ioana Banicescu. *Load Balancing and Data Locality in the Parallelization of the Fast Multiple Algorithm.* PhD thesis, Polytechnic University, 1996.

[57] Ioana Banicescu and Susan Flynn Hummel. Balancing processor loads and exploiting data locality in n-body simulations. In *Proceedings of Supercomputing'95 Conference.* IEEE Computer Society, 1995.

[58] G.R. Andrews. *Concurrent Programming Principals and Practice.* Benjamin/Cummings, Redwood City, CA, 1991.

[59] M. Ben-Ari. *Principles of Concurrent Programming.* Prentice-Hall, Englewood Cliffs, NJ, 1982.

[60] Virpin Kumar, Ananth Grama, Anshul Gupta, and George Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms.* The Benjamin/Cummings Publishing Company, Inc, Redwood City, CA, 1994.

[61] S. Brawer. *Introduction to Parallel Programming.* Academic Press, Boston, MA, 1989.

[62] A.H. Karp. Programming for parallelism. *IEEE Computer*, 20(9):43–57, 1987.

[63] N. Carriero and D. Gelernter. *How to Write Parallel Programs.* MIT Press, 1990.

[64] Honghai Zeng. Parallelization of the n-body problem with the adaptive weighted fractiling method. Master's thesis, Mississippi State University, 2002.

[65] U. Banerjee. *Dependence Analysis for Supercomputing.* Kluwer Academic Publishers, Boston, MA, 1988.

[66] W.D. Hillis and G.L. Steele. Data parallel algorithms. *Communication of the ACM*, 29(12):1170–1183, December 1986.

[67] B.P. Lester. *The Art of Parallel Programming*. Prentice-Hall, 1993.

[68] Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker, and Jack Dongarra. *MPI: The Complete Reference*. The MIT Press, Cambridge, Massachusetts, 1996.

[69] R. Eigenmann and Michael J. Voss, editors. *Openmp Shared Memory Parallel Programming: International Workshop on Openmp Applications and Tools*, West Lafayette, IN, July 2001.

[70] A.E. Mulligan and D.P. Ahlfeld. Optimal plume capture in unconfined aquifer. In J.A. Smith and S.E. Burns, editors, *Physicochemical Groundwater Remediation*, pages 23–44. Kluwer Academics, NY, 2001.

[71] Environmental Protection Agency. National recommended water quality criteria correction. Technical report, Environmental Protection Agency, 1999. EPA 822-Z-99-001.

[72] Vladimir J. Alarcon, William Kingery, and Jianping Zhu. Computational experiments on diffusion in porous media. *Computers and Electronics in Agriculture*, 2002. in process.