HydroSphere

A Three-dimensional Numerical Model Describing Fully-integrated Subsurface and Overland Flow and Solute Transport

R. Therrien, Université Laval
S. Panday, Hydrogeologic Inc.
R.G. McLaren, University of Waterloo
E.A. Sudicky, University of Waterloo
D. Demarco, Hydrogeologic Inc.
G. Matanga, USBR
P. Huyakorn, Hydrogeologic Inc.

June 23, 2003

Abstract

Effective management of watersheds and ecosystems requires a comprehensive knowledge of hydrologic processes, and impacts of point-source and non-point source pollution on water quality. Simulation models are being used increasingly to provide predictive capability in support of environmental and water resource assessment and restoration projects. However, the models used are often based on simplifications to complex hydrologic and transport processes. Such models incorporate restrictive assumptions pertaining to spatial variability, dimensionality and interaction of various components of flow and transport processes. Realizing the limitations of current models for complex, real-world applications, a fully integrated overland and subsurface flow and transport code has been developed jointly by Groundwater Simulations Group and Hydrogeologic, Inc. The subsurface module is based on the University of Waterloo and Université Laval three-dimensional (3-D) subsurface flow and transport code FRAC3DVS. The overland flow module is based on the Surface Water Flow Package of the MODHMS simulator, which is itself an enhancement of the widely popular U.S. Geological Survey code MODFLOW. The resulting code, named **HydroSphere**, provides a rigorous simulation capability that combines fully-integrated hydrologic/water quality/subsurface flow and transport capabilities with a well-tested set of user interface tools.

A unique feature in **HydroSphere** is that when the flow of water is simulated in a fully-integrated mode, water derived from rainfall inputs is allowed to partition into components such as overland and stream flow, evaporation, infiltration, recharge and subsurface discharge into surface water features such as lakes and streams in a natural, physically-based fashion. That is, the fully-coupled numerical solution approach allows the *simultaneous* solution of both the surface and variably-saturated flow regimes at each time step. This approach also permits dissolved solutes to be naturally exchanged between the overland and subsurface flow domains such that solute concentrations are also solved for *simultaneously* at each timestep in both regimes. This makes **HydroSphere** a unique and ideal tool to simulate the movement of water and solutes within watersheds in a realistic, physically-based manner.

This manual describes the physical and mathematical concepts underlying **HydroSphere** and the implementation of these concepts in the numerical model. It further provides the user with instructions and guidance on use of the code. Example problems are provided to verify the code and to acquaint the user with its applications.

HydroSphere uses the control volume finite element approach to simulate overland/subsurface flow and transport. Fully 3-D simulations of variably-saturated fractured or granular aquifers may be performed. **HydroSphere** provides several discretization options ranging from simple rectangular and axisymmetric domains to irregular domains with complex geometry and layering. Mixed element types provide an efficient mechanism for simulating flow and transport processes in fractures (2-D rectangular or triangular elements) and pumping/injection wells, streams or tile drains (1-D line elements). External flow stresses can include specified rainfall rates, hydraulic head and flux, infiltration and evapotranspiration, drains, wells, streams and seepage faces. External transport stresses include specified

concentration and mass flux and the dissolution of immiscible substances. **HydroSphere** includes options for adaptive-time stepping and output control procedures and an ILU-preconditioned ORTHOMIN solution package. A Newton-Raphson linearization package provides improved robustness.

HydroSphere is written in FORTRAN 95 and was compiled using the Compaq Visual Fortran® compiler. It will run without modification on any Microsoft Windows® based PC with sufficient RAM.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General

A diverse group of problems exists that requires quantification of the entire hydrologic cycle by integrated simulation of water flow and contaminant migration in the overland and subsurface regimes. Increased demand on limited resources for potable water and other purposes has driven the development of innovative management practices including water recycling for salt-tolerant crops, conjunctive use of surface and subsurface water resources, and artificial recharge of subsurface aquifers during wet periods. A quantification of available water within the hydrologic system and the impacts of withdrawals is essential for addressing these complex water supply issues. Irrigation practices for certain crops require flooding the fields for certain time periods. The complex cycle of irrigation; evaporation; infiltration; discharge to nearby lakes, rivers, and streams, and pumping needs to be quantified in these cases to resolve supply and demand issues. Concerns over drying and restoration of wetlands or the effects of subsurface water withdrawals on surface water features (which may fluctuate across land surface or layering features in an unsaturated zone) also require an integrated, fully-coupled analysis of the various flow regimes. Ecosystems of lakes, rivers, and bays depend on certain minimum flows as do hydropower generation, recreational use, and downstream water districts, states, and countries for their water needs. Regulating water use in hydraulically connected watershed and surficial aquifer systems necessitates an understanding of surface/subsurface water interactions and of overall seasonal hydrologic cycle behavior.

Since the early 1970s, there has been an evolution of hydrologic models for single-event and continuous simulations of rainfall-runoff processes. Earlier models quantify various hydrologic components using simplified procedures (including a unit hydrograph method, empirical formulas, system lumping, and analytical equations) that are incapable of describing flow physics and contaminant transport in any detail. In the past, numerical models based on complex multi-dimensional governing equations have not received much attention because of their computational, distributed input and parameter estimation requirements. Today, with the availability of powerful personal computers, efficient computational meth-

ods, and sophisticated GIS, remote sensing and advanced visualization tools, the hydrologic community is realizing the tremendous potential and utility of physically-based numerical simulators. As pointed out by *Woolhiser* [1996, p. 126], "there seems to be little disagreement regarding the usefulness of physically based models for understanding hydrologic systems." Models of this type are widely held to offer the greatest opportunity to examine hydrologic impact of land use change [*Refsgaard*, 1997; *Sharika et al.*, 2000]. Distributed hydrologic models also have immense potential and utility for "forecasting the movement of pollutants and sediments" [*Beven*, 1985]

FRAC3DVS is an efficient and robust numerical model that solves the three-dimensional variably-saturated subsurface flow and solute transport equations in non-fractured or discretely-fractured media and which was developed at the University of Waterloo and at Université Laval. It has enjoyed widespread acceptance with both academics and groundwater professionals since its initial release in 1995 and many new features have been added to the code since then. However, FRAC3DVS did not have the ability to simulate fully-coupled subsurface/surface water systems in an efficient and straight-forward manner. Recharge spreading layer and MODFLOW-type river node schemes are simplified approaches that have been tried with limited success. In order to remedy this situation, a modified version of Hydrogeologic Inc.'s MODHMS surface water flow packages has been incorporated into the FRAC3DVS framework. These packages were originally developed to enhance the capabilities of MODFLOW so that it could handle more complex field problems in a robust and efficient manner.

The resulting model, which we call **HydroSphere**, is documented herein. Provided in the following chapters are detailed descriptions, formulations, verification and application examples, input instructions, output formats and sample data files for the various components of the model.

HydroSphere is supported by a flexible, user-friendly modeling interface which may be used to seamlessly prepare input data sets, or visualize and interpret simulation results.

1.2 Integrated Hydrologic Model Conceptualization

HydroSphere is based on a rigorous conceptualization of the hydrologic system comprising overland and subsurface flow regimes with interactions. The model is designed to take into account all key components of the hydrologic cycle (Figure 1.1). For each time step, the model solves overland and subsurface flow and mass transport equations simultaneously and provides complete water balance and solute budgets. Referring to Figure 1.1, the overland water budget can be written as:

$$P = (Q_{S2} - Q_{S1}) - Q_{GS} + I + ET_S + Q_S^W + \Delta S_S/\Delta t \tag{1.1}$$

and the subsurface water budget as:

$$I = (Q_{G2} - Q_{G1}) + Q_{GS} + ET_G + Q_G^W + \Delta S_G/\Delta t \tag{1.2}$$

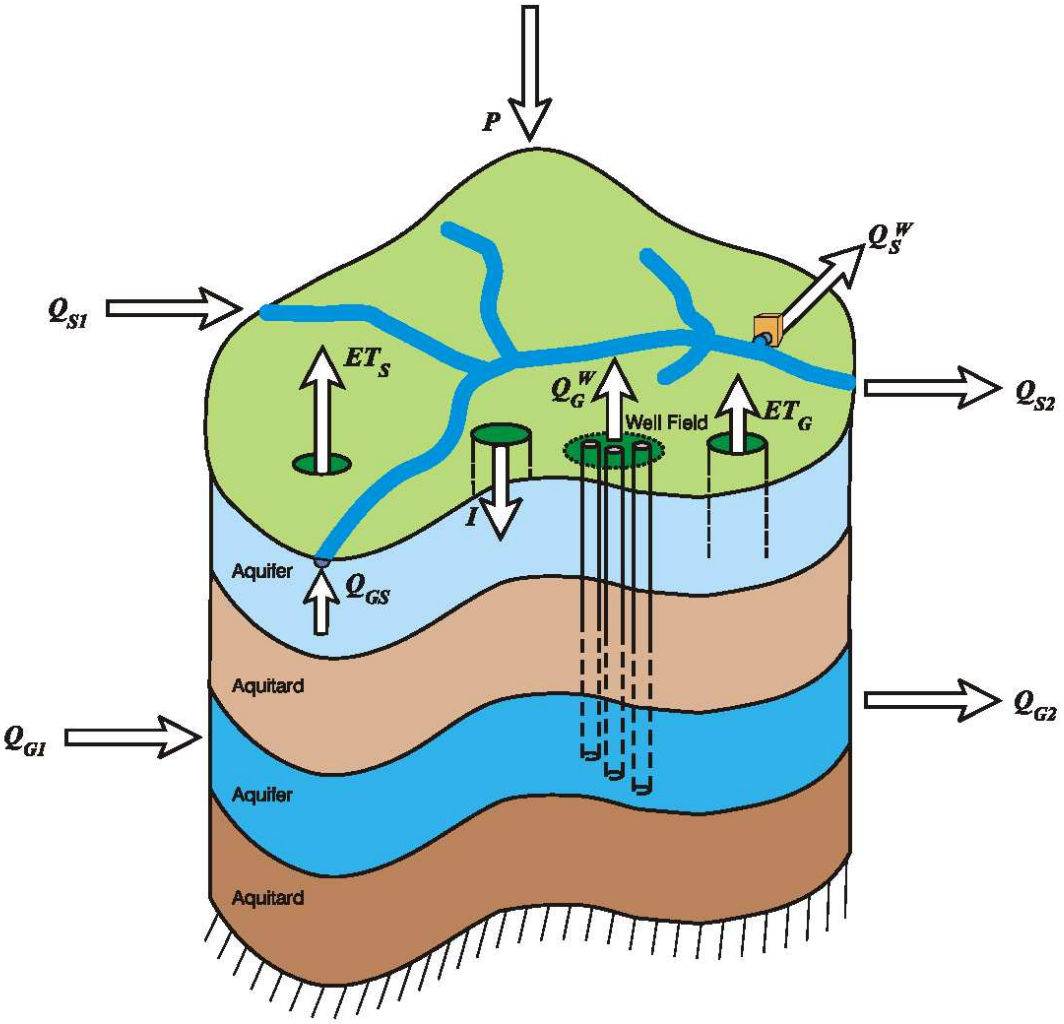


Figure 1.1: Regional Hydrologic Cycle [Adapted from *Viessman and Lewis*, 1996].

giving the total hydrologic budget as the sum of equations 1.1 and 1.2:

$$P = (Q_{S2} - Q_{S1}) + (Q_{G2} - Q_{G1}) + (ET_S + ET_G) + (Q_S^W + Q_G^W) + (\Delta S_S + \Delta S_G)\Delta t \quad (1.3)$$

where P is the net precipitation (actual precipitation - interception), Q_{S1} and Q_{S2} are the overland water inflow and outflow, Q_{GS} is the overland/subsurface water interactive flow, I is the net infiltration, ET_S is the evapotranspiration from the overland flow system, Q_S^W is the overland water withdrawal, ΔS_S is the overland water storage over time step Δt, Q_{G1} and Q_{G2} are the subsurface water inflow and outflow, ET_G is the evapotranspiration from the subsurface flow system, Q_G^W is the subsurface water withdrawal and ΔS_G is the subsurface water storage over time step Δt.

In order to accomplish the integrated analysis, **HydroSphere** utilizes a rigorous, mass conservative modeling approach that fully couples the overland flow and solute transport equations with the 3-D, variably-saturated subsurface flow and solute transport equations. This approach is significantly more robust than previous conjunctive approaches that rely on linkage of separate surface and subsurface modeling codes.

1.3 FRAC3DVS-Based Formulation

HydroSphere has been developed by extending the FRAC3DVS code to accommodate surface water flow and solute transport. We elect to use the diffusion-wave approximation of the Saint Venant equation for surface water flow [*Viessman and Lewis*, 1996], thereby neglecting the inertial terms of the momentum equation. An integrated hydrologic analysis is accomplished by the coupled solution of the diffusion-wave equation governing 2-D (areal) overland flow (including stream flow) and the Richards' equation governing 3-D unsaturated/saturated groundwater flow.

The overland and subsurface flow and transport domains are discretized simultaneously as shown in Figure 1.2. The gridding options for 2D surface flow and transport include the depicted rectangular grid that allows variable grid spacing, as well as a triangular grid option for geometric flexibility. The coupled overland and subsurface domains consist of: (1) a single layer of overland nodes, shown by triangles in Figure 1.2, situated on the land surface, (2) layers of subsurface soil and aquifer nodes, shown by circles, representing the vadose zone, subsurface aquifers and aquitards, and (3) a set of one-dimensional nodes, shown by squares, to represent surficial channels or subsurface tile-drains. The 2D overland flow grid is draped over the subsurface 3D mesh to maintain the areal topography of the land surface and to ensure that the nodes in the overland grid are coincident with those at the top of the subsurface mesh.

The 3D saturated-unsaturated flow and transport equations for the vadose and saturated zones are solved using the control volume finite element method. The top layer of surface nodes discretizes the 2-D overland flow regime, which is assembled into the matrix equations in a fully-implicit manner using a diffusion-wave approximation to the Saint Venant equations. Two methods are used to couple the two flow and transport domains. The first uses a numerical superposition principle whereby the top layer of nodes represents both overland

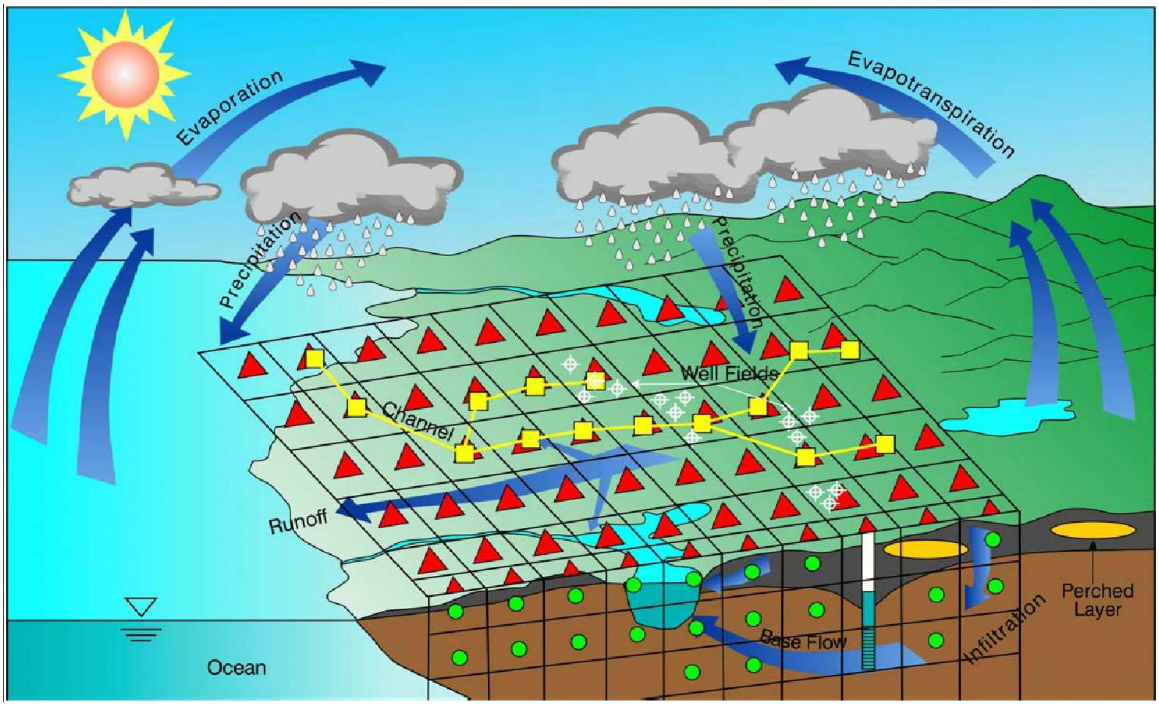


Figure 1.2: Integrated Numerical Simulation of Hydrologic System

and subsurface domains. The second method uses Darcy flux (for flow) and Fickian (for transport) relations to transfer water from the surface nodes to the first layer of subsurface nodes, with the assumption that they are separated by a (possibly) thin layer of porous material across which the leakance occurs. A single system of matrix equations arising from both discretized flow and transport regimes is then assembled for the entire hydrologic setting with appropriate boundary conditions being applied to the combined system. These could include specified rainfall rates, hydraulic head conditions, complex evapotranspiration functions, interception storage, land use, irrigation, and point sources and sinks. The fully-integrated set of nonlinear discrete equations is linearized using Newton-Raphson schemes, and is solved simultaneously in an iterative fashion at every time step.

1.4 Attributes of HydroSphere

HydroSphere is a powerful numerical simulator specifically developed for supporting water resource and engineering projects pertaining to hydrologic systems with overland and subsurface flow and mass transport components.

In terms of simulation capability and computational aspects, the **HydroSphere** code has the following attributes:

- Complete hydrologic cycle modeling using detailed physics of surface and subsurface flow in one integrated code. The overland regime consists of 2-D areal overland flow.

The subsurface regime consists of 3-D unsaturated/saturated flow. Both regimes naturally interact with each other through physical flow considerations between them.

- The fully-implicit coupling approach used by the code provides for a robust mass conserved solution scheme which is essential for systems with strong interactions between regimes.
- Advanced computational algorithms and a flexible, user-friendly interface allow the code to perform unprecedented, fully-integrated, 3-D simulation/animation on a personal computer.
- Physically-based accounting of all components of the hydrologic cycle water budget.
- Capability of modeling non-reactive and reactive chemical species transport in the associated overland and subsurface flow fields.
- Arbitrary combinations of porous, discretely-fractured, dual-porosity and dual-permeability media for the subsurface.
- Accurate handling of fluid and mass exchanges between fractures and matrix including matrix diffusion effects and solute advection in the matrix.
- Accurate delineation and tracking of the water table position, taking into account flow in the unsaturated zone, delayed yield and vertical flow components.
- Handling of non-ponding or prescribed ponding recharge conditions.
- Handling of seepage face boundary conditions.
- Automatic and correct apportioning of the total flow rate of a multi-layer well to the well nodes, including the simulation of water flow and solute mixing within the water column in the well.
- Accomodation of wellbore storage.
- Chain-reactions of radionuclide components.
- Fluid and solute mass balance tracking.
- Unstructured finite-element grids.
- Axi-symmetric grid option.
- 7-point finite-difference option.
- 8-node block or 6-node prism elements, 3- and 4-node plate elements for fractures and 2-node line elements for wells and tile drains.
- Adaptive time-stepping schemes with automatic generation and control of time steps.
- Straightforward organization and control of simulation output.

- Robust and efficient ILU-preconditioned iterative sparse-matrix solver.
- Robust and efficient Newton-Raphson linearization option.
- Flexible pre- and post-processing capabilities

For field applications and research investigations, **HydroSphere** can be used to perform event-based and continuous simulations on widely varying spatial scales ranging from single soil column profiles to large-scale basins, which may include several catchments. Examples of field applications of **HydroSphere** include:

- Integrated water resource assessment
- Watershed hydrologic analysis, including impacts of land-use or climate-change impacts on both surface and groundwater.
- Floodplain hydrologic analysis
- Fluvial hydraulic analysis.
- Contaminant migration and fate in both surface and groundwater.

1.5 Operation and Input Options

HydroSphere computational modules are built upon the widely popular FRAC3DVS code. Thus, the **HydroSphere** conjunctive overland-subsurface flow simulator enjoys the benefit of having already available and affordable GUI tools for grid generation and subsurface flow model input as well as Tecplot for 3-D visualization and animation. In order to handle spatial data analysis and visualization of surface water domain, GIS tools such as ArcView and ArcInfo may be used.

The modular code features and input/output structures of **HydroSphere** follow the original FRAC3DVS code. There are four steps involved in solving a given problem using **HydroSphere**.

1. Build the necessary data files for the pre-processor **grok**(as discussed later in Chapter 5).
2. Run **grok** to generate the input data files for **HydroSphere**.
3. Run **HydroSphere** to solve the problem and generate output data files.
4. Postprocess the output files to visualize and analyze the results and produce reports.

As a minimum, Step 1 involves creating data files which contain information for discretizing the problem domain, defining material properties for each element and specifying flow boundary conditions.

Simulation output pertaining to overland flow regime calculations is reported to the main **HydroSphere** output file in a similar manner to that for the subsurface flow calculations. Binary files are also created individually, as is done in the standard FRAC3DVS subsurface flow analysis, for further investigation or post processing.

1.6 Document Organization and Usage Guide

This document is organized into 5 chapters. The remaining chapters and their purposes are outlined below.

Chapter 2 presents the mathematical theory which describes the various physical processes which are presented in the model.

Chapter 3 shows how the mathematical theory is implemented in the **HydroSphere** code, with an emphasis on the numerical techniques used to address non-linearities which arise when dealing with, for example, variably-saturated flow.

Chapter 4 contains descriptions of several verification problems which were designed to test and demonstrate the capabilities of **HydroSphere** in solving a variety of flow and transport phenomena.

Chapter 5 introduces the user to the basic operations of the preprocessor and describes the input instructions used to determine:

- problem description
- simulation control
- grid generation
- selection of grid components (nodes, elements etc.)

as well as for the saturated-unsaturated subsurface flow, overland flow and solute transport problems.

A complete list of references cited in each chapter can be found in chapter 7.

A comprehensive index can be found at the end of this document.

Chapter 2

Theory

2.1 Subsurface Flow

2.1.1 General

The current implementation of **HydroSphere** assumes that the subsurface flow equation in a porous medium is always solved during a simulation, either for fully-saturated or variably-saturated flow conditions. We therefore first present the basic subsurface flow equation solved by **HydroSphere** which can be expanded to incorporate, among other features, discrete fractures, a second interacting porous continuum (e.g. fractures or macropores), wells and tile drains.

The following assumptions are made for subsurface flow:

- The fluid is essentially incompressible
- The porous medium and fractures (or macropores), if present, are non-deformable
- The system is under isothermal conditions
- The air phase is infinitely mobile

2.1.2 Governing Equations

2.1.2.1 Porous Medium

The following modified form of Richards' equation is used to describe three-dimensional transient subsurface flow in a variably-saturated porous medium: **rgm Sorab suggested 'In general, throughout the document, use of ϕ or "n" for porosity will conform to standard terminology usage, while use of θ_s is not common. Ed suggested we**

keep θ_s.

$$-\nabla \cdot (w_m \mathbf{q}) + \sum \Gamma_{\text{ex}} \pm Q = w_m \frac{\partial}{\partial t}(\theta_s S_w) \tag{2.1}$$

where the fluid flux $\mathbf{q}$ [L T^{-1}] is given by:

$$\mathbf{q} = -\mathbf{K} \cdot k_r \nabla(\psi + z) \tag{2.2}$$

and where $k_r = k_r(S_w)$ represents the relative permeability of the medium [dimensionless] with respect to the degree of water saturation S_w [dimensionless], ψ is the pressure head [L], z is the elevation head [L] and θ_s is the saturated water content [dimensionless], which is assumed equal to the porosity. Fluid exchange with the outside of the simulation domain, as specified from boundary conditions, is represented by Q [T^{-1}], which is a volumetric flux per unit volume representing a source (positive) or a sink (negative) to the porous medium system.

The hydraulic conductivity tensor, $\mathbf{K}$ [L T^{-1}], is given by:

$$\mathbf{K} = \frac{\rho g}{\mu}\mathbf{k} \tag{2.3}$$

where g is the gravitational acceleration [L T^{-2}], μ is the viscosity of water [M L^{-1} T^{-1}], $\mathbf{k}$ is the permeability tensor of the porous medium [L^2] and ρ is the density of water [M L^{-3}], which can be a function of the concentration C [M L^{-3}] of any given solute such that $\rho = \rho(C)$.

Water saturation is related to the water content θ [dimensionless] according to:

$$S_w = \frac{\theta}{\theta_s} \tag{2.4}$$

The volumetric fraction of the total porosity occupied by the porous medium (or primary continuum) is given by w_m [dimensionless]. This volumetric fraction is equal to 1.0 except when a second porous continuum is considered for a simulation, which is the case when the dual continuum option is used to represent existing fractures or macropores.

In equation (2.1), Γ_{ex} represent fluid exchange rates [T^{-1}] between the subsurface domain and all other types of domains supported by the model. Currently, these additional domains are overland, wells, tile drains, discrete fractures and dual continuum. The definition of Γ_{ex} (positive for flow into the porous medium) depends on the conceptualization of fluid exchange between the domains and will be defined in later sections that discuss these respective flow domains. In the equations shown for the other domains, we will use the notation ex=f, ex=d, ex=w, ex=t, ex=o, for the fracture, dual continuum, well, tile drain and overland domains, respectively.

The primary variable of solution for the nonlinear flow equation (2.1) is the pressure head, and constitutive relations must be established that relate the primary unknown ψ to the secondary variables S_w and k_r. The relative permeability may be expressed in terms of either the pressure head or the water saturation. Commonly used functions incorporated in the model are those presented by *van Genuchten* [1980] and *Brooks and Corey* [1964].

Based on earlier work by *Mualem* [1976], *van Genuchten* [1980] proposed the following saturation-pressure relation:

$$S_w = S_{wr} + (1 - S_{wr}) \left[1 + (\alpha P_c)^\beta\right]^{-\nu} \qquad \left(\nu = 1 - \frac{1}{\beta}\right) \tag{2.5}$$

with the relative permeability obtained from:

$$k_r = S_e^{1/2} \left[1 - \left(1 - S_e^{1/\nu}\right)^\nu\right]^2 \tag{2.6}$$

where α [L^{-1}] and β [dimensionless] are parameters obtained from a fit of (2.5) and (2.6) to experimental results, P_c is the capillary pressure head [L] ($P_c = -\psi$) and $S_e = (S_w - S_{wr})/(1 - S_{wr})$, with S_{wr} being the residual water saturation [dimensionless].

The Brooks and Corey relation for saturation is given by

$$S_w = S_{wr} + (1 - S_{wr}) \left(\frac{\psi_b}{\psi}\right)^{\lambda^*} \tag{2.7}$$

and the relative permeability is obtained from:

$$k_r = S_e^{n^*} \tag{2.8}$$

where λ^* is the pore-size index [dimensionless] and n^* is an exponent equal to $2 + 3\lambda^*$ [dimensionless].

Although other fundamental relations exist, any arbitrary, but physically realistic, function for $S_w(\psi)$ and $k_r(S_e)$ can also be handled through the use of tabular data input for these parameters, which is also made available in the model. Furthermore, hysteresis is not considered here, but may be included in a future version of **HydroSphere**.

To describe subsurface flow in the saturated zone, the storage term forming the right-hand side of equation (2.1) is expanded in a way similar to that presented by *Cooley* [1971] and *Neuman* [1973]. They relate a change in storage in the saturated zone to a change in fluid pressure through compressibility terms as is conventionally done in hydrogeological applications. They also assume that the bulk compressibility of the medium is constant for saturated and nearly-saturated conditions. For unsaturated conditions, it is assumed that the compressibility effects on storage of water are negligible compared to the effect of changes in saturation. The following expression is obtained for the storage term in (2.1) [*Cooley*, 1971; *Neuman*, 1973]:

$$\frac{\partial}{\partial t}(\theta_s S_w) \approx S_w S_s \frac{\partial \psi}{\partial t} + \theta_s \frac{\partial S_w}{\partial t} \tag{2.9}$$

where S_s is the specific storage coefficient of the porous medium [L^{-1}].

2.1.2.2 Discrete Fractures

A fracture is idealized here as the space between two-dimensional parallel surfaces, with the tacit assumptions that the total head is uniform across the fracture width. The equation

for variably-saturated flow in a fracture of width w_f [L] can be written by extending the saturated fracture flow equations [*Berkowitz et al.*, 1988; *Sudicky and McLaren*, 1992] and using the analogy of Richards' equation (2.1) for the porous matrix. With this extension, the governing two-dimensional flow equation is a fracture has the form:

$$-\overline{\overline{\nabla}} \cdot (w_f \mathbf{q_f}) - \Gamma_f = w_f \frac{\partial S_{wf}}{\partial t} \tag{2.10}$$

where the fluid flux $\mathbf{q_f}$ [L T^{-1}] is given by:

$$\mathbf{q_f} = -\mathbf{K_f} \cdot k_{rf} \overline{\overline{\nabla}} (\psi_f + z_f) \tag{2.11}$$

and where $\overline{\overline{\nabla}}$ is the two-dimensional gradient operator defined in the fracture plane, k_{rf} is the relative permeability of the fracture [dimensionless], ψ_f and z_f are the pressure and the elevation heads within the fracture [L], and S_{wf} is the water saturation for the fracture [dimensionless]. The saturated hydraulic conductivity of a fracture $\mathbf{K_f}$ [L T^{-1}], having a uniform aperture w_f is given by [*Bear*, 1972]:

$$\mathbf{K_f} = \frac{\rho g \, w_f^2}{12 \mu} \tag{2.12}$$

where the fluid density can be a function of the concentration C_f of any given solute in the fracture [M L^{-3}], such that $\rho = \rho(C_f)$.

Constitutive relations are also required to describe variably-saturated flow in the fractures. There is a very limited number of studies where these relationships have been derived experimentally [e.g. *Reitsma and Kueper*, 1994]. Several theoretical studies have, nevertheless, been performed in order to characterize the nature of the relationships. *Wang and Narasimhan* [1985] and *Rasmussen and Evans* [1988] generated synthetic relations between pressure, saturation and relative permeability for a single fracture surface containing a distribution of apertures. Their results were based on capillary theory and they used the well-known cubic law to represent flow in the fracture. *Pruess and Tsang* [1990] considered the problem of two-phase flow in a rough-walled fracture surface. They subdivided the fracture surface into sub-elements and assigned a spatially-correlated aperture to each. The occupancy of each element by either the wetting or the non-wetting phase fluid was based on a prescribed entry pressure relationship. Once the fluid occupancy was assigned to each element, the relative permeabilities were obtained by performing two single-phase flow simulations: one for the region occupied by the wetting phase and the other for the non-wetting phase.

In the **HydroSphere** model, relative permeability and saturation-pressure head relationships for the fractures are given by either the van Genuchten (equations 2.5 and 2.5), the Brooks-Corey model (equations 2.7 and 2.7) or they can be given in tabular forms, which gives flexibility to the user and does not restrict data entry to fixed functions.

Another process that could to be considered when describing flow in a variably-saturated fractured porous media is the reduction of the area available for flow across a fracture-matrix interface. Some portions of a fracture, as it desaturates, cannot transmit any water and

thus reduce the area available for matrix flow across the fracture. We use the approach of *Wang and Narasimhan* [1985] who represented this phenomenon with a function describing the change in effective fracture-matrix area as a function of pressure. This effective area is only applied to those matrix nodes that are also fracture nodes. Further details on the numerical implementation of the effective area concept will be discussed in the section on numerical methods.

Using arguments similar to those invoked for the porous medium equation, the storage term in equation (2.10) describing variably-saturated flow in the fractures becomes:

$$\frac{\partial}{\partial t} S_{wf} \approx S_{wf} S_{sf} \frac{\partial \psi_f}{\partial t} + \frac{\partial S_{wf}}{\partial t} \tag{2.13}$$

where S_{sf} is the specific storage coefficient for the fractures [L^{-1}]. Because it is assumed here that the fractures are non-deformable and fluid-filled, there is no contribution to the storage term from fracture compressibility. Thus, the specific storage coefficient for a fracture under saturated conditions is related to the water compressibility, α_w [L T^2 M^{-1}], according to:

$$S_{sf} = \rho g \alpha_w \tag{2.14}$$

The validity of assuming non-deformable fractures is likely to be reasonable if the fractures have a high normal stiffness or if changes in the effective stress field within the system due to pumping, for example, are small.

2.1.2.3 Dual Continuum

The **HydroSphere** model can simulate variably-saturated fluid flow in a second continuum, based on the formulation presented by *Gerke and van Genuchten* (1993). This second continuum could represent, for example, fractures or macropores that are present in a porous matrix. Similarly to flow in the porous medium, three-dimensional variably-saturated flow in the second continuum is described by a modified form of Richards' equation:

$$-\nabla \cdot (w_d \mathbf{q_d}) - \Gamma_d \pm Q_d = w_d \frac{\partial}{\partial t} (\theta_{sd} S_{wd}) \tag{2.15}$$

where the fluid flux $\mathbf{q_d}$ [L T^{-1}] is given by:

$$\mathbf{q_d} = -\mathbf{K_d} \cdot k_{rd} \nabla (\psi_d + z_d) \tag{2.16}$$

and where k_{rd} is the relative permeability of the medium [dimensionless] with respect to the degree of water saturation S_{wd} [dimensionless], ψ_d is the pressure head [L], z_d is the elevation head [L] and θ_{sd} is the saturated water content [dimensionless], which is equal to the porosity of the dual continuum. Fluid exchange with the outside of the simulation domain is represented by a volumetric flux per unit volume Q_d [T^{-1}]. The volumetric fraction of the total porosity occupied by the dual continuum is given by w_d [dimensionless]. We assume here that the sum of volumetric fraction of the dual continuum w_d and that of the porous medium w_m is equal to 1.0.

The hydraulic conductivity tensor of the dual continuum, $\mathbf{K_d}$ [L T^{-1}], is given by:

$$\mathbf{K_d} = \frac{\rho g}{\mu}\mathbf{k_d} \tag{2.17}$$

where $\mathbf{k_d}$ is the permeability tensor of the dual continuum [L^2] and where the density of water can be a function of the concentration C_d [M L^{-3}] of any given solute in the dual continuum such that $\rho = \rho(C_d)$.

Similarly to the porous medium, the functional relationships relating pressure head to saturation and relative permeability to saturation are described by either the van Genuchten or the Brooks-Corey functions, or are given in tabular form.

For cases where the dual continuum represents fractures, some expressions can be derived to relate the permeability of the fractures to a representative fracture aperture and spacing (for example, *Bear* 1972). For example, for a set of parallel fractures of uniform aperture w_f with a uniform spacing equal to f_s [L], and assuming one-dimensional flow in a direction parallel to the fracture, the equivalent permeability of the set of fractures is given by:

$$\mathbf{k_d} = \frac{w_f^2}{12}\frac{w_f}{f_s} = \frac{w_f^3}{12 f_s} \tag{2.18}$$

rgm Rene, you need to address the storage term S_{sd}? which appears later in eq 3.28

2.1.2.4 Wells

The equation describing 1D free-surface flow along the axis of a well having a finite storage capacity and penetrating a variably-saturated aquifer is [*Therrien and Sudicky*, 2000]:

$$-\overline{\nabla}\cdot(\pi r_s^2 \mathbf{q_w}) \pm Q_w\delta(l-l') - \Gamma_w = \pi\frac{\partial}{\partial t}\left[\left(r_c^2/L_s + r_s^2 S_{ww}\right)\psi_w\right] \tag{2.19}$$

where the fluid flux $\mathbf{q_w}$ [L T^{-1}] is given by:

$$\mathbf{q_w} = -K_w k_{rw}\overline{\nabla}(\psi_w + z_w) \tag{2.20}$$

and where $\overline{\nabla}$ is the one-dimensional gradient operator along the length direction, l, of the well, r_s and r_c are the radius of the well screen and well casing [L], respectively, L_s is the total length of the screen [L], k_{rw} is the relative permeability of the well [dimensionless] and S_{ww} is its saturation [dimensionless]. The pressure and elevation heads in the well screen are given by ψ_w [L] and z_w [L], respectively, and the discharge or recharge rate per unit length Q_w [L^2 T^{-1}] is applied at location l' in the well screen and $\delta(l-l')$ is the Dirac delta function.

The hydraulic conductivity of the well K_w [L T^{-1}] is obtained from the Hagen-Poiseuille formula [*Sudicky et al.*, 1995]:

$$K_w = \frac{r_c^2 \rho g}{8\mu} \tag{2.21}$$

where the fluid density can be a function of the concentration C_w of any given solute in the well [M L^{-3}], such that $\rho = \rho(C_w)$.

The term on the right hand side of (2.19) represents the storage coefficient of the well bore and comprises two parts: the storage resulting from the compressibility of the fluid and storage caused by the variation of the water level in the casing. Although it can assumed that the former contribution is negligible, it is retained in the formulation. Similarly to *Sudicky et al.* [1995], the storage contribution arising from the change in water level is redistributed along the well screen.

The relative permeability and saturation functions are used to restrict flow along the well for cases where the water level drops below the top of the well screen. To simulate the portion of the well above the water table, and where there is no flow in the well bore, a correction term equivalent to the relative permeability of a porous medium is used to reduce the hydraulic conductivity of the well in equation (2.20). Numerical experiments have shown that, if a zero-relative permeability is used for well nodes above the water table (with negative pressure heads), severe numerical difficulties arise in the Newton iteration and lead to divergence of the solution. To avoid these difficulties, but still restrict flow in the well above the water table, the correction term is chosen such that the equivalent hydraulic conductivity of the well screens becomes lower than that of the surrounding porous matrix, by two orders of magnitude. In that way, the well nodes that are above the water level have a negligible contribution to the total flow.

2.1.2.5 Tile Drains

Flow in tile drains can be described by the general equation of continuity for flow in an open channel [*Dingman*, 1994]:

$$-\overline{\nabla}\cdot(A\mathbf{q_t}) + Q_t\delta(l-l') - \Gamma_t = \frac{\partial A}{\partial t} \tag{2.22}$$

where $\overline{\nabla}$ is the one-dimensional gradient operator along the length direction, l, of the tile drain, Q_t is the specified fluid flow rate in or out of the drain at location l' [L^3 T^{-1}], $\delta(l-l')$ is the Dirac delta function, l is the distance along the drain [L], and A is the cross-sectional area in the wetted portion of the tile [L^2].

The fluid flux, $\mathbf{q_t}$ [L T^{-1}], along the channel may be expressed by several different formulations depending on conditions of flow along the channel. Assuming a laminar flow regime gives:

$$\mathbf{q_t} = -K_t k_{rt}\nabla(\psi_t + z_t) \tag{2.23}$$

where k_{rt} is the relative permeability of the drain, ψ_t is the depth of fluid, z_t is the drain elevation [L] and K_t is the drain conductivity [L T^{-1}] approximated for shallow laminar flow as [*Dingman*, 1994]:

$$K_t = \frac{\rho g \psi_t^2}{3\mu} \tag{2.24}$$

where the fluid density can be a function of the concentration C_t of any given solute in the tile drain [M L^{-3}], such that $\rho = \rho(C_t)$.

The discharge $\mathbf{q_t}$ along the channel may alternatively be formulated from friction formulas and the diffusion-wave approximation to the shallow water flow equations. For this case, $\mathbf{q_t}$ is again given by equation 2.23, with K_t now being a function of the friction factor and the hydraulic radius R_t [L] of the channel with

$$R_t = \frac{A}{P_t} \tag{2.25}$$

and where P_t is the wetted perimeter of the channel section [L]. Thus, for Manning's equation, K_t is given by:

$$K_t = \frac{R_t^{2/3}}{n}\left[\frac{\partial \psi_t}{\partial l} + \frac{\partial z_t}{\partial l}\right]^{-1/2} \tag{2.26}$$

where n is the Manning coefficient $[TL^{-1/3}]$. For the Chezy equation, K_t is given by:

$$K_t = C_c R_t^{1/2}\left[\frac{\partial \psi_t}{\partial l} + \frac{\partial z_t}{\partial l}\right]^{-1/2} \tag{2.27}$$

where C_c is the Chezy coefficient $(L^{1/2}/T)$. For the Darcy-Weisbach equation, K_t is given by:

$$K_t = \sqrt{8g/f}\, R_t^{1/2}\left[\frac{\partial \psi_t}{\partial l} + \frac{\partial z_t}{\partial l}\right]^{-1/2} \tag{2.28}$$

Finally, the volumetric discharge Q_t along the channel ($Q_t = \mathbf{q_t} A$) may be influenced by the presence of hydraulic structures such as dams, weirs, spillways, culverts, gates, pumping stations, etc. To accommodate a variety of structures with a wide range of designs, **HydroSphere** allows the use of rating curves, which are tabulated input of Q_t vs ψ_t relationships. These rating curves may be experimentally determined or calculated from formulas readily available for the various structures considered.

2.2 Overland Flow

2.2.1 General

This section describes the mathematical theory of the surface water flow package of the **HydroSphere** simulator. Overland flow on catchment basins is an important component of the hydrologic cycle, governing flow to and from the subsurface, channel networks, rivers, lakes and reservoirs. Lake and reservoir flow dynamics and hydrologic conditions of wetlands are also areal overland flow processes.

Areal overland flow is represented in **HydroSphere** by a depth-averaged flow equation, which is the diffusion-wave approximation of the Saint Venant equation for surface water flow. Before presenting the diffusion-wave equation solved by **HydroSphere** we present the simplifications needed to obtain this equation from the full two-dimensional Saint Venant equation.

2.2.2 Governing Equations

2.2.2.1 Overland Flow

The two-dimensional Saint Venant equations for unsteady shallow water flow consist of 3 equations, which are given by the following mass balance equation **rgm Sorab says 'why deviate from using the gradient symbol?**

$$\frac{\partial \phi_o h_o}{\partial t} + \frac{\partial(\overline{v}_{xo} d_o)}{\partial x} + \frac{\partial(\overline{v}_{yo} d_o)}{\partial y} + \Gamma_o \pm Q_o = 0 \tag{2.29}$$

coupled with the momentum equation for the x-direction

$$\frac{\partial}{\partial t}(\overline{v}_{xo} d_o) + \frac{\partial}{\partial x}(\overline{v}_{xo}^2 d_o) + \frac{\partial}{\partial y}(\overline{v}_{xo}\overline{v}_{yo} d_o) + g d_o \frac{\partial d_o}{\partial x} = g d_o (S_{ox} - S_{fx}) \tag{2.30}$$

and the momentum equation for the y-direction

$$\frac{\partial}{\partial t}(\overline{v}_{yo} d_o) + \frac{\partial}{\partial y}(\overline{v}_{yo}^2 d_o) + \frac{\partial}{\partial x}(\overline{v}_{xo}\overline{v}_{yo} d_o) + g d_o \frac{\partial d_o}{\partial x} = g d_o (S_{oy} - S_{fy}) \tag{2.31}$$

where d_o is the depth of flow [L], z_o is the bed (land surface) elevation [L], h_o is the water surface elevation [L] ($h_o = z_o + d_o$), $\overline{v}_{xo}$ and $\overline{v}_{yo}$ are the vertically averaged flow velocities in the $x-$ and $y-$directions [L T^{-1}], Q_o is a volumetric flow rate per unit area representing external source and sinks [L T^{-1}], and ϕ_o is an overland flow surface porosity which is unity for flow over a flat plane, and varies between zero at the land surface and unity at the top of all rills and obstructions, for flow over an uneven surface. This conceptualization is discussed further in section 2.2.2.2. The variables S_{ox}, S_{oy}, S_{fx}, and S_{fy} are dimensionless bed and friction slopes in the $x-$ and $y-$directions, respectively. These slopes can be approximated with either the Manning, the Chezy or the Darcy-Weisbach equations.

Using the Manning equation, the friction slopes are approximated by:

$$S_{fx} = \frac{\overline{v}_{xo}\overline{v}_{so} n_x^2}{d_o^{4/3}} \tag{2.32}$$

and

$$S_{fy} = \frac{\overline{v}_{yo}\overline{v}_{so} n_y^2}{d_o^{4/3}} \tag{2.33}$$

where $\overline{v}_{so}$ is the vertically averaged velocity [L T^{-1}] along the direction of maximum slope s ($\overline{v}_{so} = \sqrt{v_{xo}^2 + v_{yo}^2}$), n_x and n_y are the Manning roughness coefficients [L$^{-1/3}$ T] in the x and y directions.

Using the Chezy equation, the friction slopes are approximated by:

$$S_{fx} = \frac{1}{C_x^2} \frac{\overline{v}_{xo}\overline{v}_{so}}{d_o} \tag{2.32}$$

and

$$S_{fy} = \frac{1}{C_y^2} \frac{\overline{v}_{yo}\overline{v}_{so}}{d_o} \tag{2.33}$$

where C_x and C_y are the Chezy coefficients [$L^{1/2}$ T^{-1}] in the x and y directions.

Using the Darcy-Weisbach equation, the friction slopes are approximated by:

$$S_{fx} = \frac{f_x}{8g}\frac{\overline{v}_{xo}\overline{v}_{so}}{d_o} \tag{2.32}$$

and

$$S_{fy} = \frac{f_y}{8g}\frac{\overline{v}_{yo}\overline{v}_{so}}{d_o} \tag{2.33}$$

where f_x and f_y are dimensionless Darcy-Weisbach friction factors in the x and y directions. The friction factors f_x and f_y may be obtained from a Moody diagram which can be approximated [*Akan and Yen*, 1981] for laminar flow as:

$$f_i = \frac{C_L}{Re_i} \tag{2.34}$$

where the index i represents the x or y direction, C_L is a constant which depends on rainfall intensity as:

$$C_L = 24 + 27.162r^{0.407} \tag{2.35}$$

where r is the rainfall intensity in in/hr, and Re_i is the Reynolds number in direction i given as

$$Re_i = \frac{\overline{v}_{io}d_o}{\gamma} \tag{2.36}$$

where γ is the kinematic viscosity.

Momentum equations 2.30 and 2.31 can be simplified by neglecting the first three terms on the left hand side, representing inertia (*Gottardi and Venutelli*, 1993) and using either approximation of the friction slopes (equations 2.32 and 2.33) to give:

$$\overline{v}_{ox} = -K_{ox}\frac{\partial h_o}{\partial x} \tag{2.37}$$

and

$$\overline{v}_{oy} = -K_{oy}\frac{\partial h_o}{\partial y} \tag{2.38}$$

where K_{ox} and K_{oy} are overland conductances [L T^{-1}] that depend on the equation used to approximate the friction slopes. Conductances for the Manning equation are given by:

$$K_{ox} = \frac{d_o^{2/3}}{n_x}\frac{1}{[\partial h_o/\partial s]^{1/2}} \tag{2.39}$$

and

$$K_{oy} = \frac{d_o^{2/3}}{n_y}\frac{1}{[\partial h_o/\partial s]^{1/2}} \tag{2.40}$$

where s is taken in the direction of maximum slope.

For the Chezy equation, K_{ox} and K_{oy} are given by:

$$K_{ox} = C_x d_o^{1/2} \frac{1}{[\partial h_o/\partial s]^{1/2}} \tag{2.39}$$

and

$$K_{oy} = C_y d_o^{1/2} \frac{1}{[\partial h_o/\partial s]^{1/2}} \tag{2.40}$$

Similarly, for the Darcy-Weisbach relation, K_{ox} and K_{oy} are given by:

$$K_{ox} = \sqrt{\frac{8g}{f_x}} \frac{1}{[\partial h_o/\partial s]^{1/2}} \tag{2.39}$$

and

$$K_{oy} = \sqrt{\frac{8g}{f_x}} \frac{1}{[\partial h_o/\partial s]^{1/2}} \tag{2.40}$$

Comparison of the various expressions for the conductance indicates the following relationship between the coefficients of the Manning, Chezy and Darcy-Weisbach equations:

$$C_x = \frac{d_o^{1/6}}{n_x} = \sqrt{\frac{8g}{f_x}} \quad \text{and} \quad C_y = \frac{d_o^{1/6}}{n_y} = \sqrt{\frac{8g}{f_y}} \tag{2.41}$$

The overland flow equation solved by **HydroSphere** is finally obtained by substituting equations 2.37 and 2.38 into continuity equation 2.29, which gives the following diffusion wave approximation for overland flow:

$$\frac{\partial \phi_o h_o}{\partial t} - \frac{\partial}{\partial x}\left(d_o K_{ox} \frac{\partial h_o}{\partial x}\right) - \frac{\partial}{\partial y}\left(d_o K_{oy} \frac{\partial h_o}{\partial y}\right) + \Gamma_o \pm Q_o = 0 \tag{2.42}$$

with equations 2.39 and 2.40 providing expressions for conductances K_{ox} and K_{oy}. **rgm Sorab, Ed wonders if storage term missing?**

In addition to neglecting the inertial terms, the assumptions associated to the diffusion-wave equation are those of the Saint Venant equations, which are depth-averaged flow velocities, hydrostatic pressure distribution vertically, mild slope, and dominant bottom shear stresses. Furthermore, it is assumed that Manning's, Chezy's, or Darcy-Weisbach's formula are valid to calculate frictional resistance forces for unsteady flow.

2.2.2.2 Treatment of Rill Storage and Storage Exclusion for Rural and Urban Environments

The storage and flow terms of equation 2.42 have been further enhanced to include rill storage and exclusion effects for investigations of urban runoff or agricultural settings. If ϕ_o is always unity, it is assumed that flow occurs over a flat plane as shown in Figure 2.1a. For flood calculations in urban environments, the setting may be much different as shown

in Figure 2.1b, with flow occurring between the grid of buildings in an averaged sense over the grid area. If the flood is high enough to cover the buildings, only then is the full area available for flow and storage of water, otherwise, if not accounted for, lower flood-depths and incorrect discharge would be predicted for an urban flood event. The storage capacity that is reduced by the presence of the urban features is called obstruction storage exclusion. Obstruction storage exclusion may also result from vegetation in rural or agricultural settings. In addition, these features may also affect the conductance of the horizontal flow term due to additional frictional resistance and small scale energy dissipation over the obstruction heights. Rill storage (also known as depression storage) may be an important factor in several urban as well as rural settings as shown in Figure 2.1c. This is the amount of storage that must be filled before any lateral overland flow can occur. Microtopographic relief (as compared to grid-block scale) is included in depression storage and can have a substantial impact on hydrograph shape [*Woolhiser et al.*, 1997]. Finally, for agricultural plots or grass lands (shown in Figure 2.1d), the storage effects of rills as well as storage exclusion of the crop must be taken into account. For these cases, both depression storage as well as obstruction storage exclusion need to be included in the model. In addition, horizontal flow term conductances may also be affected over the obstruction heights.

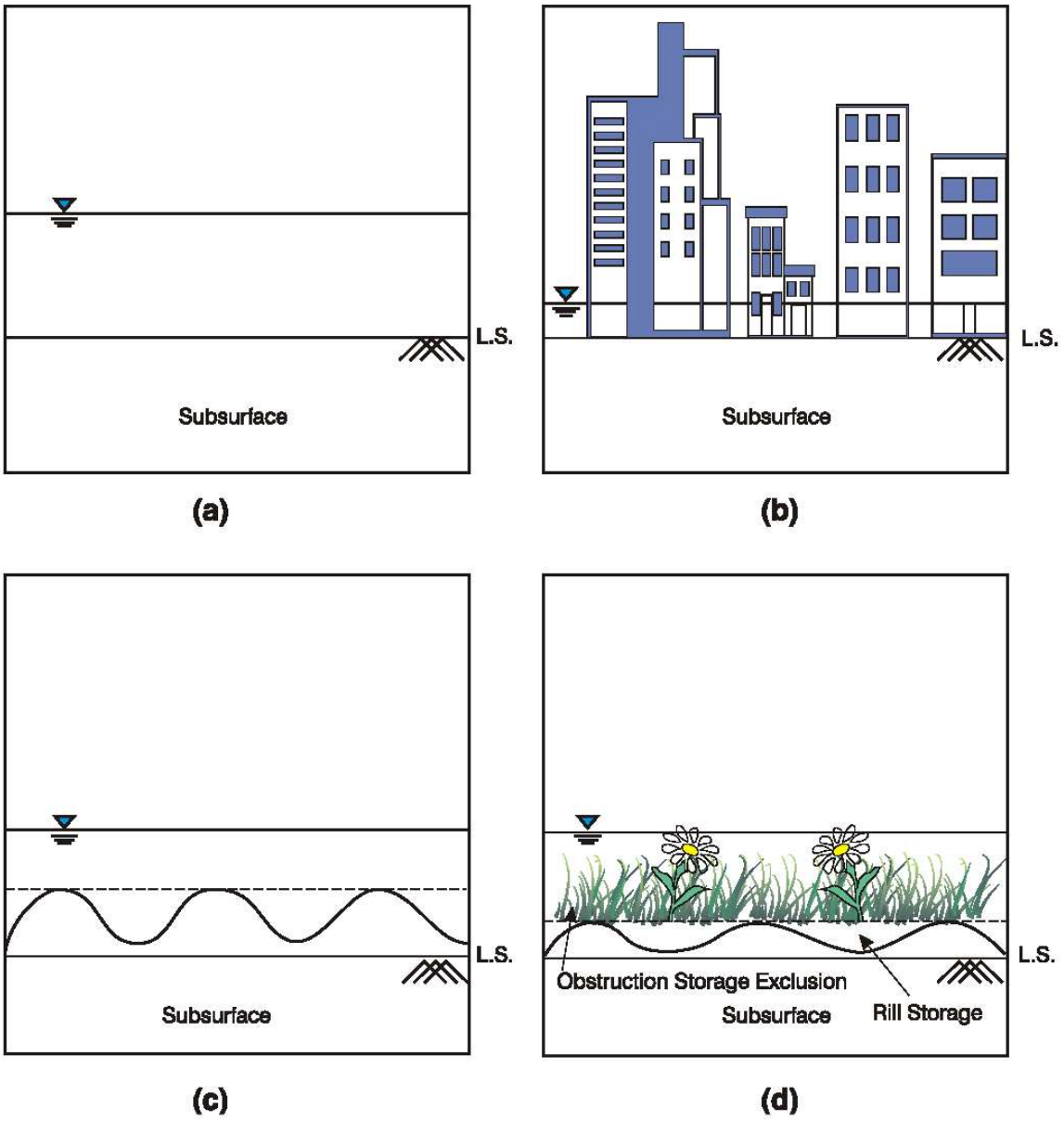
Depression storage and obstruction storage exclusion are both modeled by assuming that the geometry of depressions and exclusions combined, has a maximum elevation and that the area covered by surface water varies between zero and full area, from land surface up to this maximum elevation as shown in Figure 2.2. The variation of area covered by surface water with depth (H_s) is expressed as a volumetric height which is assumed to be parabolic. The slope of this curve is a porosity or void ratio which varies between zero at land surface up to unity at height H_s. Linear or other functions may have been used, however, a parabolic variation provides for continuous derivatives at land surface and at the maximum height H_s, thus assisting during numerical solution by Newton-Raphson or modified-Picard linearization methods. The heights of depression storage H_d and of storage within the obstructions, H_o, may be estimated such that they geometrically represent the mean spacing (equivalent void space) within the respective storage elements. The depression storage height above land surface ($LS + H_d$), is also used to indicate the elevation below which flow depth is zero in the advection terms of equation 2.42, when depression storage is modeled for a system. Thus, overland flow occurs laterally only above elevations of $LS+H_d$, i.e., when water levels are above the depression storage elevations. In addition, conductance terms K_{ox} and K_{oy} may be further reduced by a factor K_s above this elevation, up to the obstruction height of storage exclusion to account for the additional resistance losses. The factor K_s varies from zero to unity as the obstruction heights vary from 0 to H_o.

To simplify the presentation of the discretized overland flow equation in the next chapter, equation (2.42) is rewritten in vectorial notation

$$-\nabla \cdot (d_o \mathbf{q_o}) - \Gamma_o \pm Q_o = \frac{\partial h_o}{\partial t} \tag{2.43}$$

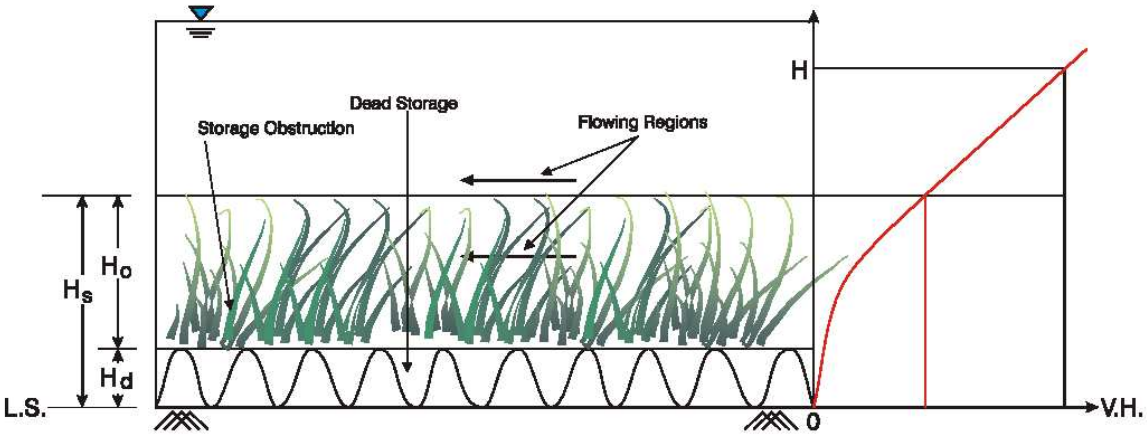
where the fluid flux $\mathbf{q_o}$ [L T^{-1}] is given by:

$$\mathbf{q_o} = -\mathbf{K_o} \cdot k_{ro} \nabla (d_o + z_o) \tag{2.44}$$



Note that Rill storage, obstruction storage exclusion as well as friction coefficients may be varied on a stress period basis to account for crop growth or changes in land use.

Figure 2.1: Treatment of storage terms for various settings.



L.S. = Land surface = elevation of top of underlying subsurface node.

H_d = height of depression storage = height at which overland flow starts to occur.

H_o = height of storage within obstructions

H_s = $H_d + H_o$ = maximum height over which area covered by surface water goes from zero to unity.

V.H. = 'volume height' defined as the height from L.S., of an equivalent volume of water without rills or obstructions.

Figure 2.2: Conceptual model for depression storage and obstruction storage exclusion.

2.3 Flow Coupling

Two different approaches are used to define the water exchange terms Γ_{ex} between two different domains. The first approach is based on superposition (see *Therrien and Sudicky*, 1996), where continuity of hydraulic head is assumed between the two domains concerned, which corresponds to instantaneous equilibrium between the two domains. In that case, the Γ_{ex} term does not need to be evaluated implicitly in the model and we do not present its definition. However, the fluid exchange can be computed after the numerical solution of the discrete equations as a post-processing step. This approach corresponds to the common node scheme mentioned later in the manual.

The second method is more general because it does not assume continuity of hydraulic head between two domains but uses a Darcy flux relation to transfer water from one domain to the other. The Darcy flux is computed from the hydraulic head difference between two domains and assumes that they are separated by a (possibly) thin layer of porous material across which water exchange occurs. This second approach corresponds to the dual node scheme mentioned later in the manual.

The common node approach is currently the only one available in the model to simulate the exchange between the subsurface porous medium and fractures, between the porous medium and wells, and between the porous medium and the tile drains. On the other hand, fluid exchange between the subsurface porous medium and a dual continuum can only be simulated with the dual node approach. Finally, fluid exchange between the subsurface porous medium and the overland system can be simulated with either the common or dual node approach. We present here the definition of the exchange term for the dual node approach.

2.3.1 Porous Medium - Macropore Coupling

When the dual node approach is chosen to represent simultaneous flow in the subsurface porous medium and a second continuum (representing fractures or macropores), the exchange term can be defined as (*Gerke and van Genuchten*, 1993):

$$\Gamma_d = \alpha_{wd}(\psi_d - \psi) \tag{2.45}$$

where α_{wd} is a first-order transfer coefficient for water [L^{-1} T^{-1}] defined by (*Gerke and van Genuchten*, 1993) as

$$\alpha_{wd} = \alpha^*_{wd} K_a k_{ra} \tag{2.46}$$

with

$$\alpha^*_{wd} = \frac{\beta_d}{a^2}\gamma_w \tag{2.47}$$

where β_d is a dimensionless geometrical factor, a is the distance [L] between the center of a fictitious matrix block and the fracture boundary, and γ_w is a dimensionless empirical coefficient. The hydraulic conductivity of the interface between the two domains is given by K_a [L T^{-1}] and its relative permeability is k_{ra}. For dual-porosity systems, the geometry

factor β_d has been shown to be equal to 3 for rectangular slabs and 15 for spheres. *Gerke and van Genuchten* [1993] provide more detail on the evaluation of the exchange term.

2.3.2 Porous Medium - Overland Flow Coupling

When the dual node approach is chosen to represent simultaneous flow in the subsurface porous medium and the overland domain, the exchange term is given by:

$$\Gamma_o = k_{rso} K_{so} (h - h_o) \tag{2.48}$$

where a positive Γ_o represents flow from the subsurface system to the overland system as determined by 2.29, h_o is the overland head, h is the head of the subsurface flow system which now has an additional connection to the overland flow system via K_{so}, the leakance across the ground surface to the modeled subsurface defined as the conductivity of the bottom divided by the thickness of the bottom surface across which flow occurs. The term k_{rso}, which accounts for the rill effects, is zero when the overland flow surface is completely dry.

2.4 Flow Boundary Conditions

2.4.1 Subsurface flow

Boundary conditions for subsurface flow include the following: first-type (Dirichlet) boundaries of prescribed hydraulic head, areal infiltration or recharge, source/sinks, evaporation, and seepage faces. The boundary conditions can also be allowed to vary in time. Details of the implementation of these boundary conditions in the model are given in the next chapter.

The areal recharge boundary is not required when solving for the overland flow domain since the solution to the interacting system determines the subsurface recharge.

2.4.2 Overland flow

Boundary conditions to the overland flow system include the following: first-type (Dirichlet) boundaries of prescribed water elevation, direct rainfall inputs, source/sinks, evaporation, zero-depth gradient and nonlinear critical-depth conditions.

2.5 Solute Transport

2.5.1 Governing Equations

The current formulation of **HydroSphere** assumes that the relevant flow equations are always solved prior to a transport simulation. Similarly to flow, we first present the basic

subsurface transport equation solved by **HydroSphere** which can be expanded to incorporate, among other features, discrete fractures, a second porous continuum, wells, tile drains, and the overland flow domain.

2.5.1.1 Subsurface Porous Matrix Transport

Three-dimensional transport of solutes in a variably-saturated porous matrix is described by the following equation:

$$-\nabla \cdot w_m \left(\mathbf{q}C - \theta_s S_w \mathbf{D} \nabla C\right) + \left[R\lambda C\right]_{par} + \sum \Omega_{\text{ex}} \pm Q_c = w_m \left[\frac{\partial(\theta_s S_w R C)}{\partial t} + \theta_s S_w R \, \lambda \, C\right] \tag{2.49}$$

where C is the solute concentration [M L^{-3}] of the current species amongst possibly multiple species and λ is a first-order decay constant [L^{-1}]. The subscript *par* designates parent species for the case of a decay chain. For the case of a straight decay chain, there is only one parent species, as might be the case for a radioactive decay chain, howver, for degrading organic species, a particular species may have several parent sources through a complex degradation process. Solute exchange with the outside of the simulation domain, as specified from boundary conditions, is represented by Q_c [M L^{-3} T^{-1}] which represents a source (positive) or a sink (negative) to the porous medium system. The assumption of fluid incompressibility is made in (2.49). The dimensionless retardation factor, R, is given by [*Freeze and Cherry*, 1979]:

$$R = 1 + \frac{\rho_b}{\theta_s S_w} K' \tag{2.50}$$

where ρ_b is the bulk density of the porous medium [M L^{-3}] and K' is the equilibrium distribution coefficient describing a linear Freundlich adsorption isotherm [L^{-3} M]. Note that for variably-saturated conditions, the water saturation appears in the definition of R.

The hydrodynamic dispersion tensor $\mathbf{D}$ [L^2 T^{-1}] is given by [*Bear*, 1972]:

$$\theta_s S_w \mathbf{D} = (\alpha_l - \alpha_t) \frac{\mathbf{qq}}{|\mathbf{q}|} + \alpha_t |\mathbf{q}| \mathbf{I} + \theta_s \, S_w \, \tau D_{free} \, \mathbf{I} \tag{2.51}$$

where α_l and α_t are the longitudinal and transverse dispersivities [L], respectively, $|\mathbf{q}|$ is the magnitude of the Darcy flux, τ is the matrix tortuosity [dimensionless], D_{free} is the free-solution diffusion coefficient [L^2 T^{-1}] and $\mathbf{I}$ is the identity tensor. The product τD_{free} represents an effective diffusion coefficient for the matrix. In the unsaturated zone, the tortuosity is allowed to vary with the water saturation, S_w, according to the Millington-Quirk relationship [*Millington and Quirk*, 1961], given by

$$\tau = (S_w \theta_s)^{7/3} / \theta_s^2 \tag{2.52}$$

In equation (2.49), Ω_{ex} represent solute exchange rates [T^{-1}] between the subsurface domain and all other types of domains supported by the model. Currently, these additional domains are overland, wells, tile drains, discrete fractures, immobile second continuum and mobile dual continuum. The definition of Ω_{ex} depends on the conceptualization of solute exchange

between the domains and will be defined later. In the equations shown for the other domains, we will use the notation ex=*f*, ex=*im*, ex=*d*, ex=*w*, ex=*t*, ex=*o*, for the fracture, immobile continuum, dual continuum, well, tile drain and overland domains, respectively.

2.5.1.2 Fractures

The equation for two-dimensional solute transport in a variably-saturated fracture follows from the equation describing solute transport in a fully-saturated fracture [e.g. *Tang et al.*, 1981; *Sudicky and McLaren*, 1992, *Therrien and Sudicky*, 1996]. Its form is:

$$-\nabla \cdot (w_f \mathbf{q_f} C_f - w_f S_{wf} \mathbf{D_f} \nabla C_f) + [R_f \lambda_f C_f]_{par} - \Omega_f = w_f \left[\frac{\partial (S_{wf} R_f C_f)}{\partial t} + S_{wf} R_f \lambda_f C_f \right] \tag{2.53}$$

where C_f is the concentration in a fracture [M L^{-3}], λ_f is a first-order decay constant [L^{-1}] and $\mathbf{D_f}$ is the hydrodynamic dispersion tensor of the fracture [L^2 T^{-1}]. An expression similar to (2.51) can be used to represent $\mathbf{D_f}$, where dispersivities and fluxes correspond to those of the fracture and the fracture porosity is assumed to be unity. The dimensionless retardation factor R_f is defined according to [*Freeze and Cherry*, 1979]:

$$R_f = 1 + \frac{2K'_f}{w_f} \tag{2.54}$$

where K'_f is a fracture-surface distribution coefficient [L^{-1}].

2.5.1.3 Double-porosity

The model can simulate double-porosity transport with the classical first-order theory (see *Sudicky* [1990]), which divides the subsurface domain into a mobile and an immobile region. The formulation used here for double-porosity is restricted to steady-state flow conditions. It is assumed that the porous medium represents the mobile domain, where flow is described by equation (2.1) and solute transport is described by equation (2.49). It is also assumed that there is no flow in the immobile domain, therefore no equation is required for immobile flow, and solute mass balance is given by

$$\frac{\partial (\theta_{\mathrm{Imm}} C_{\mathrm{Imm}})}{\partial t} - \Omega_{\mathrm{Imm}} = 0 \tag{2.55}$$

where C_{Imm} is the solute concentration in the immobile region [M L^{-3}], θ_{Imm} is the porosity of the immobile region [dimensionless] and Ω_{Imm} represents the solute exchange flux between the mobile and immobile zones [M L^{-3} T^{-1}].

2.5.1.4 Isotopic fractionation

The model can simulate isotope fractionation, using a first-order kinetic formulation that is mathematically similar to double-porosity transport. In analogy to double-porosity, we assume here that the mobile domain represent the water phase and the immobile domain is associated to the solid, or rock, phase.

Similarly to double-porosity transport, the formulation used here for isotope fractionation is restricted to steady-state flow conditions. Fractionation from the water phase is also restricted to a single solid (or rock) phase. It is assumed that the porous medium represents the water domain, where flow is described by equation (2.1) and isotope transport is described by equation (2.49). It is also assumed that there is no isotope flow in the solid phase, therefore no equation is required for flow in the solid phase.

Mass balance for the isotope in the solid phase is given by:

$$\frac{\partial C_{\mathrm{Imm}}}{\partial t} - \frac{\Omega_{\mathrm{Imm}}}{x_r} = 0 \tag{2.56}$$

where C_{Imm} is the isotope concentration in the solid, or immobile region [M L^{-3}], Ω_{Imm} represents here the isotope exchange flux between the mobile (water) and immobile (solid) zones [M L^{-3} T^{-1}], and x_r is the dimensionless mass ratio of the isotope in the solid phase to that in the water phase, for a unit volume of water-saturated rock of constant porosity.

2.5.1.5 Dual Continuum

When the dual continuum option is used, advective-dispersive solute transport can be simulated in a second continuum based on the formulation presented by *Gerke and van Genuchten* (1993). Note that, as opposed to the double-porosity option, the dual continuum option is not restricted to steady-state flow conditions because it allows transient fluid exchange between the two interacting continuua. Also, fluid flow and advective-dispersive solute transport can be simultaneously solved in the porous medium and the second continuum. Three-dimensional transport of solutes in a variably-saturated dual continuum is described by:

$$-\nabla \cdot w_d \left(\mathbf{q_d} C_d - \theta_{sd} S_{wd} \mathbf{D_d} \nabla C_d\right) + \left[R_d \lambda_d C_d\right]_{par} - \Omega_d \pm Q_{cd} =$$

$$w_d \left[\frac{\partial(\theta_{sd} S_{wd} R_d C_d)}{\partial t} + \theta_{sd} S_{wd} R_d \lambda_d C_d\right] \tag{2.57}$$

where C_d is the solute concentration in the dual continuum [M L^{-3}] and λ_d is a first-order decay constant [L^{-1}]. Solute exchange with the outside of the simulation domain, as specified from boundary conditions, is represented by Q_{cd} [M L^{-3} T^{-1}] which represents a source (positive) or a sink (negative) to the dual continuum. The dimensionless retardation factor, R_d, is given by [*Freeze and Cherry*, 1979]:

$$R_d = 1 + \frac{\rho_{bd}}{\theta_{sd} S_{wd} K'_d} \tag{2.58}$$

where ρ_{bd} is the bulk density of the dual continuum [M L^{-3}] and K'_d is the equilibrium distribution coefficient describing a linear Freundlich adsorption isotherm [L^{-3} M]. Note that for variably-saturated conditions, the water saturation appears in the definition of R_d.

The hydrodynamic dispersion tensor $\mathbf{D_d}$ [L^2 T^{-1}] is given by [*Bear*, 1972]:

$$\theta_{sd} S_{wd} \mathbf{D_d} = (\alpha_{ld} - \alpha_{td}) \frac{\mathbf{q_d q_d}}{|\mathbf{q_d}|} + \alpha_{td} |\mathbf{q_d}| \mathbf{I} + \theta_{sd}\, S_{wd}\, \tau_d D_{free}\, \mathbf{I} \tag{2.59}$$

where α_{ld} and α_{td} are the longitudinal and transverse dispersivities [L], respectively, $|\mathbf{q_d}|$ is the magnitude of the Darcy flux, τ_d is the dual continuum tortuosity [dimensionless], D_{free} is the free-solution diffusion coefficient [L^2 T^{-1}] and $\mathbf{I}$ is the identity tensor. The product $\tau_d D_{free}$ represents an effective diffusion coefficient for the dual continuum. Recall from the previous discussion on flow in the unsaturated zone that the tortuosity is a function of the degree of saturation according to equation 2.52.

2.5.1.6 Wells

One-dimensional solute transport along the axis of a well is described by:

$$-\overline{\nabla} \cdot \pi r_s^2 \left(\mathbf{q_w} C_w - S_{ww} D_w \overline{\nabla} C_w \right) + \pi r_s^2 \left[\lambda C_w \right]_{par} - Q_w (C_w - C_{w_{\text{Inj}}}) \delta(l - l') - \Omega_w = \pi r_s^2 \frac{\partial C_w}{\partial t} \tag{2.60}$$

where C_w is the solute concentration in the well [M L^{-3}], $\mathbf{q_w}$ is the fluid flux along the well axis [L T^{-1}], and $C_{w_{\text{Inj}}}$ is the concentration of injected water.

The dispersion coefficient for the well, D_w [L^2 T^{-1}] is equal to [*Lacombe et al.*, 1995]:

$$D_w = \frac{r_s^2 \mathbf{q_w}^2}{48 D_{free}} + D_{free} \tag{2.61}$$

2.5.1.7 Tile Drains

One-dimensional solute transport along the axis of a tile drain is described by:

$$-\overline{\nabla} \cdot A \left(\mathbf{q_t} C_t - S_{wt} D_t \overline{\nabla} C_t \right) + A \left[\lambda C_t \right]_{par} - Q_t (C_t - C_{t_{\text{Inj}}}) \delta(l - l') - \Omega_t = A \frac{\partial C_t}{\partial t} \tag{2.62}$$

where C_t is the solute concentration in the tile drain [M L^{-3}], $\mathbf{q_t}$ is the fluid flux along the drain axis [L T^{-1}], and $C_{t_{\text{Inj}}}$ is the concentration of injected water.

The dispersion coefficient for the drain, D_t [L^2 T^{-1}] has a form similar to that for a one-dimensional well and is equal to:

$$D_t = \frac{r_t^2 \mathbf{q_t}^2}{48 D_{free}} + D_{free} \tag{2.63}$$

2.5.1.8 Overland Surface

The equation for two-dimensional transport of solutes along the overland surface is written as

$$-\overline{\overline{\nabla}}(q_o C_o - \mathbf{D_o}\phi_o h_o \overline{\nabla} C_o) + [\phi_o h_o R_o \lambda C_o]_{par} - \Omega_o = \frac{\partial}{\partial t}(\phi_o h_o R_o C_o) + \phi_o h_o R_o \lambda C_o \quad (2.64)$$

where C_o is the concentration in water on the overland surface [M L^{-3}], $\mathbf{D_o}$ is the hydrodynamic dispersion tensor of the overland flow surface [L^2 T^{-1}] and $-\overline{\overline{\nabla}}$ is the vertically integrated two-dimensional gradient operator. An expression similar to (2.51) is used to represent D_o and one similar to (2.54) to represent R_o.

2.6 Solute Transport Coupling

Similarly to fluid flow, two different approaches are used to define the solute exchange terms Ω_{ex} between two different domains. The first approach is based on a numerical superposition principle (see *Therrien and Sudicky*, 1996), where continuity of solute concentration is assumed between the two domains concerned, which corresponds to instantaneous equilibrium between the two domains. In that case, the Ω_{ex} term does not need to be evaluated explicitly in the model and we do not present its definition. However, the solute exchange flux between domains can be computed after the numerical solution at a given time step. This approach corresponds to the common node scheme.

The second method is more general because it does not assume continuity of concentration between two domains, but uses a first-order expression to approximate Fickian transport to transfer solute from one domain to the other. This second approach corresponds to the dual-node scheme mentioned later in the manual.

The common node approach is currently the only one available in **HydroSphere** to simulate solute exchange between the subsurface porous medium and the fractures or macropores, between the porous medium and wells, and between the porous medium and tiles drains. Solute exchange between the subsurface porous medium and the immobile region (double-porosity option), and between the subsurface porous medium and a dual continuum is only simulated by the dual-node approach. For solute exchange between the overland and subsurface domains, both the common-node approach as well as the dual-node approach are available options for coupling. We present here the definition of the exchange term for the dual-node approach.

2.6.1 Mobile - Immobile Region Coupling

Solute exchange between the mobile and immobile region of a porous medium (double-porosity approach) is given by:

$$\Omega_{\mathrm{Imm}} = \alpha_{\mathrm{Imm}}(C - C_{\mathrm{Imm}}) \quad (2.65)$$

where α_{Imm} is a first-order mass transfer coefficient between the mobile and immobile regions [L^{-1}].

For fractured porous media, *Sudicky* [1990] presents relationships for the mass transfer coefficient as a function of fracture geometry. For example, if a porous medium is highly fractured and the shape of the porous medium block delineated by the fracture network can be approximated as spheres, the mass transfer coefficient can be approximated by:

$$\alpha = \frac{15\theta_{\text{Imm}} D^*_{\text{Imm}}}{r_0^2} \tag{2.66}$$

where D^*_{Imm} is the effective diffusion coefficient in the immobile region and r_0 is the radius of a representative sphere [L]. Another expression can be given for the case of a system of parallel fractures, with a uniform fracture spacing equal to w_f and where the porous matrix blocks are prismatic slabs:

$$\alpha = \frac{3\theta_{\text{Imm}} D^*_{\text{Imm}}}{(w_f/2)^2} \tag{2.67}$$

2.6.2 Isotopic Fractionation Coupling

Isotopic exchange between the mobile (water) and immobile (solid) region of a porous medium is similar to the double-porosity approach and is given by:

$$\Omega_{\text{Imm}} = x_r k_r (\alpha_r C - C_{\text{Imm}}) \tag{2.68}$$

where k_r is a forward fractionation rate [L^{-1}] and α_r is the isotope fractionation factor. Concentration C_{Imm} describes here the isotopic concentration in the solid phase.

2.6.3 Porous Medium - Dual Continuum Coupling

When the dual-node approach is chosen to represent simultaneous transport in the subsurface porous medium and a dual continuum (representing fractures), the solute exchange term can be defined as (*Gerke and van Genuchten*, 1993): **rgm Sorab asks 'should this just be the darcy flux between the two nodes? Our equation is written in terms of flux, not velocity'**

$$\Omega_d = -u_m C - u^* C^* \tag{2.69}$$

where **rgm Sorab asks 'what is** α_s **'**

$$u_m = d^* \Gamma_d \phi^* - \alpha_s w_m \theta_s S_w \tag{2.70}$$

and

$$u^* = (1 - d^*)\Gamma_d \phi + \alpha_s w_m \theta_s S_w \tag{2.71}$$

In (2.70) and (2.71), we define the following variables:

$$d^* = 0.5 \left(1 - \frac{\Gamma_d}{|\Gamma_d|}\right) \tag{2.72}$$

and

$$\phi = w_m \frac{\theta_s S_w}{\theta_{tot}}; \quad \phi^* = (1 - w_m) \frac{\theta_s^* S_w^*}{\theta_{tot}}; \quad \theta_{tot} = w_m \theta_s S_w + (1 - w_m) \theta_s^* S_w^* \tag{2.73}$$

2.6.4 Porous Medium - Overland Coupling

Solute exchange between the surface and subsurface domains is calculated purely by advection, for the dual-node approach of representing the two systems. Advection occurs with the fluid flow rate between the two domains, Γ_o, as solved in the flow equation.

Chapter 3

Numerical Implementation

3.1 General

HydroSphere uses the control volume finite element method to solve the overland and subsurface flow equations, and it uses either the standard Galerkin finite element method or the control volume finite element method to solve the transport equation. Elements available to solve the 3D porous medium and dual continuum equations are rectangular prisms (8-node elements), 3D triangular prisms (6-node elements), and 3D tetrahedra (4-node elements). The 2D fracture and overland equations are solved for using either rectangular (4-node elements) or triangular elements (3-node elements) and the 1D well and tile drain equation are solved for 1-D linear elements (2-node elements). For the 3D and 2D elements, a finite difference approximation is also available, according to the method presented by *Panday et al.* (1993).

The model solves either linear equations (for fully-saturated flow or solute transport) or non-linear equations (for variably-saturated subsurface flow, overland flow, solute transport with a flux-limiter, including density-dependent flow and transport). To solve the non-linear equations, **HydroSphere** uses the robust Newton-Raphson linearization method, except for the weakly nonlinear density-dependent problem, which is solved by the Picard method. Although the Newton-Raphson technique requires a larger amount of work for each solution step compared to other linearization methods such as Picard iteration, the robustness and higher order of convergence of the Newton method make it attractive.

The matrix equation arising from the discretization is solved by a preconditioned iterative solver, using either the ORTHOMIN, GMRES or BiCGSTAB acceleration.

In this chapter, we present the discretized equations and provide details on the various solution schemes used by the model. We first present a general description of the control volume finite element method used to discretize the governing equations. We then present the discretized equations for subsurface and overland flow and for transport. We finally present the solution method for non-linear equations.

rgm Sorab says Not to be too picky, but transport equations written in section 2 are all in reduced form, but when you go to discretized equations in section 3, they are all in primitive form. Equations in section 2 should be in primitive form, and the code should also be such.

3.2 Control Volume Finite Element Method

The proposed method of solution for the flow problem is based on the control volume finite element approach [e.g. *Forsyth*, 1991; *Kropinski*, 1990] which has been shown to be particularly well-suited for a fast and efficient implementation of the Newton-Raphson linearization technique [*Forsyth and Simpson*, 1991].

The basic idea of the control volume finite element approach is to obtain a discretized equation that mimics the governing mass conservation equation locally. A volume of influence, referred to as a control volume, is assigned to each node. The discretized equation for a given node then consists of a term describing the change in fluid mass storage for that volume which is balanced by the term representing the divergence of the fluid mass flux in the volume. The fluid mass flux will depend on the physical properties associated with the volume and the difference in the value of the primary variable between the node in question and its neighbors.

Discretization of the subsurface and the overland flow equations is identical except for the difference in dimensionality. For the sake of clarity, we present here a detailed description of the control volume finite element method applied to discretize a simplified prototype continuity equation. The final discretized equations for all subsurface domains and for overland flow are then presented without providing the details of the derivation.

Let us assume the following prototype flow equation

$$\frac{\partial}{\partial t}(\theta_s S_w) - \nabla \cdot (\mathbf{K} \cdot k_r \nabla h) + Q = 0 \tag{3.1}$$

where h is the hydraulic head, equal to $\psi + z$.

Let N_i be the standard finite element basis functions such that

$$\begin{aligned} N_i &= 1 \text{ at node } i \\ &= 0 \text{ at all other nodes} \\ \sum_j N_j &= 1 \text{ everywhere in the solution domain} \end{aligned} \tag{3.2}$$

Using the standard basis function, an approximating function is defined in the usual way for the spatially and temporally variable h and S_w:

$$h \simeq \hat{h} = \sum_j N_j h_j(t)$$

$$S_w \simeq \hat{S_w} = \sum_j N_j S_{wj}(t) \tag{3.3}$$

where j is a nodal index ranging from 1 to n, where n is the total number of nodes.

The standard Galerkin technique (see e.g. *Huyakorn and Pinder* [1983]) is used to discretize equation (3.1) over the domain of interest, V, leading to:

$$\int_V \left[\frac{\partial}{\partial t}(\theta_s \hat{S_w}) - \nabla \cdot \left(\mathbf{K} \cdot k_r \nabla \hat{h}\right) + Q\right] N_i \, dV = 0 \tag{3.4}$$

where i is the nodal index ranging from 1 to total number of nodes. We now only consider the equation applying to node i. Upon approximating the time derivative by a finite difference representation and using a lumped mass approach to treat the storage terms in (3.4), we can write:

$$\int_v \frac{\partial}{\partial t}(\theta_s \hat{S_w}) N_i dv = \theta_s \frac{\left(\hat{S}_w^{L+1} - \hat{S}_w^{L}\right)}{\Delta t} \int_v N_i \, dv \tag{3.5}$$

where v is the region or control volume associated with node i, L is the time level and Δt is the time-step size. Here, a fully implicit discretization in time is used. Likewise, for the region v associated with node i, the source/sink term in (3.4) becomes:

$$Q_i = \int_v Q N_i \, dv \tag{3.6}$$

It is now desired to express the flux term in equation (3.4) as a function of the total head difference between node i and each of its neighbors. By applying the divergence theorem to this term in equation (3.4), one obtains:

$$\int_v -\nabla \cdot \left(\mathbf{K} \cdot k_r \nabla \hat{h}\right) N_i \, dv = \int_v \nabla N_i \cdot \mathbf{K} \cdot k_r \nabla \hat{h} \, dv - \int_B q^* N_i \, dB \tag{3.7}$$

rgm Rene, please define q^s or get it out of my sight :) The last term on the right-hand side represents the fluid flux normal to the boundary, B, of volume v. Let us assume for clarity that this quantity is zero. Use can be made of (3.3) to get:

$$\nabla N_i \cdot \nabla \hat{h} = \nabla N_i \cdot \nabla \left(\sum_j h_j N_j\right) \tag{3.8}$$

where the summation is carried over all the nodes. Using the relation $\sum N_j = 1$. we have:

$$N_i = 1 - \sum_{j \neq i} N_j \tag{3.9}$$

such that:

$$\nabla N_i = -\nabla \sum_{j \neq i} N_j \tag{3.10}$$

Using equations (3.9) and (3.10), the right-hand side of equation (3.8) can now be rewritten in the following way:

$$\begin{aligned} \nabla N_i \cdot \nabla \left(\sum_j h_j N_j\right) &= \nabla N_i \cdot \nabla \left(\sum_{j \neq i} h_j N_j\right) + \nabla \left(h_j N_j\right) \\ &= \nabla N_i \cdot \nabla \left(\sum_{j \neq i} N_j\right) (h_j - h_i) \end{aligned} \tag{3.11}$$

Relationship (3.11) is used to obtain:

$$\int_v \nabla N_i \cdot \mathbf{K} \cdot k_r \nabla \hat{h} \; dv = \int_v \nabla N_i \cdot \mathbf{K} \cdot k_r \nabla \left(\sum_{j \neq i} N_j \right) (h_j - h_i) \; dv \tag{3.12}$$

Discretization of the domain into finite elements will create a nodal connectivity (*i.e.* a table of nodal incidences for the elements). Defining η_i as being the set of nodes connected to node i, it is obvious that the nodes not included in η_i will not contribute to the change in storage or fluid flow at node i. Using this result from discretization and the fact that the $h_j - h_i$ are nodal quantities and that the summation and integration operations are interchangeable, the right-hand side of (3.12) becomes:

$$\int_v \nabla N_i \cdot \mathbf{K} \cdot k_r \nabla \left(\sum_{j \neq i} N_j \right) (h_j - h_i) \; dv = \sum_{j \in \eta_i} \int_v \nabla N_i \cdot \mathbf{K} \cdot \nabla N_j \, (h_j - h_i) \; dv \tag{3.13}$$

Using the following relation

$$\gamma_{ij} = \int_v \nabla N_i \cdot \mathbf{K} \cdot \nabla N_j \; dv \tag{3.14}$$

the right-hand side of (3.13) becomes:

$$\sum_{j \in \eta_i} \int_v \nabla N_i \cdot \mathbf{K} \cdot \nabla N_j \, (h_j - h_i) \; dv = \sum_{j \in \eta_i} \lambda_{ij+1/2} \gamma_{ij} \, (h_j - h_i) \tag{3.15}$$

where $\lambda_{ij+1/2}$ represent a weighted value of the relative permeabilities for nodes i and j, evaluated at the interface between nodal volumes i and j.

Combining (3.5), (3.6) and (3.15), and using fully implicit time weighting, the final form of the discretized equation for node i becomes:

$$\left[(\theta_s S_w)_i^{L+1} - (\theta_s S_w)_i^{L} \right] \frac{v_i}{\Delta t} = \sum_{j \in \eta_i} (\lambda)_{(ij+1/2)}^{L+1} \gamma_{ij} (h_j^{L+1} - h_i^{L+1}) + Q_i^{L+1} \tag{3.16}$$

where superscript L denotes the time level and where the volume of influence for node i is given by

$$v_i = \int_v N_i \, dv \tag{3.17}$$

The discretized equation presented above is independent of the choice of element type. Of the numerous types of three-dimensional elements that can be used to discretize the porous blocks, both 8-node rectangular block elements [*Huyakorn et al.*, 1986] and 6-node prism elements are implemented here. The user also has the option of subdividing rectangular block or prism elements into 4-node tetrahedral elements, which permits the discretization of highly irregular domains. The two-dimensional fracture planes and the overland flow are discretized using either rectangular or triangular elements [*Huyakorn et al.*, 1984]. This choice of simple elements allows use of the influence coefficient technique [*Frind*, 1982;

Huyakorn et al., 1984] to analytically evaluate the integrals appearing in (3.14) in an efficient manner.

An option that has been implemented in the numerical formulation is a choice between a finite element or a finite difference representation using a methodology identical to that described by *Panday et al.* [1993] and *Therrien* [1992]. The problem and boundary conditions are defined in terms of finite elements upon input and new nodal connectivities are established when a finite difference formulation is chosen. **rgm Ed says we should mention FD prism option** The influence coefficient matrices for the finite element method, presented by *Huyakorn et al.* [1986], are manipulated in order to mimic a finite difference discretization. The reader is referred to *Panday et al.* [1993] and *Therrien* [1992] for details on the implementation. The finite difference form of the needed influence coefficient matrices are provided in the appendix **rgm Ed says these are not in appendix**. Because of the different number of nodal connections (*i.e.* 7 for finite difference, 27 for finite element), the finite element method requires nearly four times as much memory to store the coefficient matrix compared to that for the finite difference method. The size of the assembled coefficient matrix is $n \times 27$ for finite elements and $n \times 7$ for finite differences, where n is the number of nodes in the domain. A variety of subsurface flow simulations that we have performed for fractured porous media under either fully- or variably-saturated conditions, have indicated that the finite difference and finite element representations yield similar results. Also, experience has indicated that the CPU time required when using the finite difference method is also nearly a factor of four less.

In the following section, we present the discretized flow equation for the porous medium, fractures, dual continuum, wells, tile drains and overland. The discretized equations are similar to the discretized form of the prototype equation (3.16) and their derivation, not shown here, follows the steps highlighted in this section.

3.3 Discretized Subsurface Flow Equations

3.3.1 Porous Medium

Using the control volume finite element method, with fully implicit time weighting, the following discretized porous medium flow equation is obtained

$$\left[(S_s S_w \psi + \theta_s S_w)_i^{L+1} - (S_s S_w \psi + \theta_s S_w)_i^{L}\right] \frac{w_m v_i}{\Delta t} =$$

$$\sum_{j \in \eta_i} (\lambda)_{(ij+1/2)}^{L+1} \gamma_{ij} (h_j^{L+1} - h_i^{L+1}) - \left(\sum \Gamma_{\text{ex}}^{L+1}\right) v_i \pm Q_i^{L+1} \tag{3.18}$$

where

$$h_i = \psi_i + z_i \tag{3.19}$$

and

$$\gamma_{ij} = \int_v \nabla N_i \cdot w_m \mathbf{K} \cdot \nabla N_j \, dv \tag{3.20}$$

for interpolation functions N defined for the 3D porous medium elements and where the 3D volume associated with a given node is given by

$$v_i = \int_v N_i \, dv \tag{3.21}$$

For upstream weighting, the $\lambda_{ij+1/2}$ values are given by:

$$\begin{aligned} \lambda_{ij+1/2} &= k_{rj} \quad \text{if} \quad \gamma_{ij} \left(h_j - h_i \right) > 0 \\ \lambda_{ij+1/2} &= k_{ri} \quad \text{if} \quad \gamma_{ij} \left(h_j - h_i \right) < 0 \end{aligned} \tag{3.22}$$

Upstream weighting of the relative permeability is highly recommended in order to ensure monotonicity of the solution [*Forsyth*, 1991]. A monotone solution will yield saturations that always remain in the physical range, (*i.e.* between 0.0 and 1.0). *Kropinski* [1990] provides a vivid illustration of the importance of the type of relative permeability weighting on the quality and stability of the solution of Richards' equation. She considered a two-dimensional unsaturated flow problem in which the air entry pressure was low which makes the governing equation more hyperbolic in nature. Results showed that the use of central weighting produced significant negative saturations as the wetting front advanced in a relatively dry soil; however, the solution was stable and remained in the physical range when upstream weighting was used. Although it is well-known that the use of upstream weighting for advection-dispersion problems can lead to excessive smearing of a concentration front, its use in conjunction with hyperbolic-type equations, such as the one for variably-saturated flow, does not smear as much because the solution is self-sharpening. It has also been shown that, for purely hyperbolic equations, the use of central weighting can cause complete failure of the solution [*Sammon*, 1988].

The reduction of area available for flow across a fracture-matrix interface can be easily incorporated in equation (3.18) in a manner suggested by *Wang and Narasimhan* [1985]. The area between nodes i and j, which are both located in the matrix, is imbedded in the γ_{ij} term, defined by equation (3.20). For cases where i or j also coincide with a fracture node, this area between the two nodes is multiplied by a factor representing the new effective area, accounting for a matrix-matrix flow area reduction when the fracture desaturates.

It should also be noted that in the discretized equation (3.18), the saturation term is represented exactly and no use is made of the water capacity term which is known to induce severe mass balance errors [*Celia et al.*, 1990, *Milly*, 1985]. Various schemes ranging in complexity have therefore been derived to resolve this problematic mass balance error [e.g. *Cooley*, 1983; *Milly*, 1985; *Celia et al.*, 1990]. In this work, the use of the Newton-Raphson procedure allows a direct representation of the saturation term, thereby completely avoiding the difficulties arising from the use of the water capacity term.

3.3.2 Discrete Fractures

The discretized 2D equation for flow in fractures is

$$\left[(S_{wf})_i^{L+1} - (S_{wf})_i^L\right] \frac{w_f a_i}{\Delta t} = \sum_{j \in \eta_{f_i}} (\lambda_f)_{(ij+1/2)}^{L+1} \gamma_{f_{ij}} (h_{f_j}{}^{L+1} - h_{f_i}{}^{L+1}) + \Gamma_f^{L+1} a_i \tag{3.23}$$

where η_{f_i} is the set of fracture nodes connected to fracture node i through the 2D fracture elements and where

$$h_{f_i} = \psi_{f_i} + z_{f_i} \tag{3.24}$$

and

$$\gamma_{f_{ij}} = \int_a \nabla N_i \cdot w_f \mathbf{K_f} \cdot \nabla N_j \; da \tag{3.25}$$

with interpolation functions N defined for the 2D fracture elements.

The 2D area associated with a given fracture node is given by

$$a_i = \int_a N_i \, da \tag{3.26}$$

For upstream weighting of relative permeabilities, the $(\lambda_f)_{ij+1/2}$ values are given by:

$$\begin{aligned} (\lambda_f)_{ij+1/2} &= k_{rfj} \quad \text{if} \;\; \gamma_{f_{ij}} \left(h_{f_j} - h_{f_i}\right) > 0 \\ (\lambda_f)_{ij+1/2} &= k_{rfi} \quad \text{if} \;\; \gamma_{f_{ij}} \left(h_{f_j} - h_{f_i}\right) < 0 \end{aligned} \tag{3.27}$$

3.3.3 Dual Continuum

Using the control volume finite element method, the discretized equation for flow in a dual continuum is

$$\begin{aligned} &\left[(S_{sd} S_{wd} \psi_d + \theta_{sd} S_{wd})_i^{L+1} - (S_{sd} S_{wd} \psi_d + \theta_{sd} S_{wd})_i^L\right] \frac{w_m^* v_i}{\Delta t} = \\ &\sum_{j \in \eta_{id}} (\lambda_d)_{(ij+1/2)}^{L+1} \gamma_{dij} (h_{dj}{}^{L+1} - h_{di}{}^{L+1}) - \left[\Gamma_d^{L+1} + Q_{di}{}^{L+1}\right] v_i \end{aligned} \tag{3.28}$$

where volume v_i is defined for the dual continuum nodes, with an expression similar to (3.21) and where

$$h_{di} = \psi_{di} + z_{di} \tag{3.29}$$

and

$$\gamma_{dij} = \int_v \nabla N_i \cdot w_d \mathbf{K_d} \cdot \nabla N_j \; dv \tag{3.30}$$

where the interpolation functions N are defined for the 3D dual continuum elements.

For upstream weighting of relative permeabilities, the $\lambda_{dij+1/2}$ values are given by:

$$\begin{aligned} \lambda_{dij+1/2} &= k_{drj} \quad \text{if} \;\; \gamma_{dij} \left(h_{dj} - h_{di}\right) > 0 \\ \lambda_{dij+1/2} &= k_{dri} \quad \text{if} \;\; \gamma_{dij} \left(h_{dj} - h_{di}\right) < 0 \end{aligned} \tag{3.31}$$

3.3.4 Wells

Using the control volume finite element method, the discretized equation for flow along the axis of a well is

$$\pi \left\{ \left[(r_c^2/L_s + r_s^2 S_{ww})_i \, \psi_{wi} \right]^{L+1} - \left[(r_c^2/L_s + r_s^2 S_{ww})_i \, \psi_{wi} \right]^{L} \right\} \frac{l_i}{\Delta t} =$$

$$\sum_{j \in \eta_{wi}} (\lambda_w)^{L+1}_{(ij+1/2)} \gamma_{wij} ({h_{wj}}^{L+1} - {h_{wi}}^{L+1}) - \Gamma_w^{L+1} l_i + {Q_{wi}}^{L+1} \tag{3.32}$$

where η_{wi} is the set of well nodes connected to well node i through the 1D well elements and where

$$h_{wi} = \psi_{wi} + z_{wi} \tag{3.33}$$

and

$$\gamma_{wij} = \int_l K_w \nabla N_i \cdot \nabla N_j \; dl \tag{3.34}$$

where the interpolation functions N are defined for the 1D well elements.

The length associated with a given well node is given by

$$l_i = \int_l N_i \, dl \tag{3.35}$$

For upstream weighting, the $(\lambda_w)_{(ij+1/2)}$ values are given by:

$$\begin{aligned} (\lambda_w)_{(ij+1/2)} &= k_{rwj} \quad \text{if} \;\; \gamma_{wij}(h_{wj} - h_{wi}) > 0 \\ (\lambda_w)_{(ij+1/2)} &= k_{rwi} \quad \text{if} \;\; \gamma_{wij}(h_{wj} - h_{wi}) < 0 \end{aligned} \tag{3.36}$$

3.3.5 Tile Drains

Using the control volume finite element method, the discretized equation for flow along a tile drain is

$$(A_i^{L+1} - A_i^L) \frac{l_i}{\Delta t} = \sum_{j \in \eta_{ti}} (\lambda_t)^{L+1}_{(ij+1/2)} \gamma_{tij} (h_{tj}^{L+1} - h_{ti}^{L+1}) - \Gamma_t^{L+1} l_i + {Q'}_{ti}^{L+1} \tag{3.37}$$

where η_{ti} is the set of tile drain nodes connected to tile drain node i through the 1D tile drain elements and where

$$h_{ti} = \psi_{ti} + z_{ti} \tag{3.38}$$

and

$$\gamma_{tij} = \int_l K_t \nabla N_i \cdot \nabla N_j \; dl \tag{3.39}$$

where the interpolation functions N are defined for the 1D tile drain elements.

The length associated with a given tile drain node is given by

$$l_i = \int_l N_i \, dl \tag{3.40}$$

For upstream weighting, the $(\lambda_t)_{(ij+1/2)}$ values are given by:

$$\begin{aligned} (\lambda_t)_{(ij+1/2)} &= k_{rtj} \quad \text{if} \quad \gamma_{tij}(h_{tj} - h_{ti}) > 0 \\ (\lambda_t)_{(ij+1/2)} &= k_{rti} \quad \text{if} \quad \gamma_{tij}(h_{tj} - h_{ti}) < 0 \end{aligned} \tag{3.41}$$

3.4 Discretized Overland Flow Equation

The discretized 2D overland flow equation is

$$\left[(h_o)_i^{L+1} - (h_o)_i^L\right] \frac{a_i}{\Delta t} = \sum_{j \in \eta_{oi}} (\lambda_o)_{(ij+1/2)}^{L+1} \gamma_{oij} (h_o{}_j^{L+1} - h_o{}_i^{L+1}) + \Gamma_o^{L+1} a_i \pm q_{oi} \tag{3.42}$$

where η_{oi} is the set of overland nodes connected to overland node i through the 2D overland elements and where

$$h_{oi} = d_{oi} + z_{oi} \tag{3.43}$$

and

$$\gamma_{oij} = \int_a \nabla N_i \cdot \mathbf{K_o} \cdot \nabla N_j \; da \tag{3.44}$$

with interpolation functions N defined for the 2D overland elements.

The 2D area associated with a given overland node is given by

$$a_i = \int_a \phi_o N_i \, da \tag{3.45}$$

For upstream weighting of overland pseudo relative permeabilities, the $(\lambda_o)_{ij+1/2}$ values are given by:

$$\begin{aligned} (\lambda_o)_{ij+1/2} &= k_{roj} \quad \text{if} \quad \gamma_{oij} \, (h_{oj} - h_{oi}) > 0 \\ (\lambda_o)_{ij+1/2} &= k_{roi} \quad \text{if} \quad \gamma_{oij} \, (h_{oj} - h_{oi}) < 0 \end{aligned} \tag{3.46}$$

Careful consideration should be provided to implementing the conductances of the flow terms of equation 3.44. The conductance tensor $\mathbf{K_o}$ has two components in 2 dimensions, K_{ox} and K_{ox}, which are combinations of the properties of the two nodes involved in the respective flow connection. The constant part in equations 2.39 and 2.40, constitutes the frictional resistance term ($1/n_x$ and $1/n_y$ for Manning, C_x and C_y for Chezy, or $\sqrt{8g/f_x}$ and $\sqrt{8g/f_y}$ for Darcy-Weisbach, all referred to as H_x and H_y) and is an input parameter for each element. The gradient part of equations 2.39 and 2.40, $[\partial h_o/\partial s]^{-1/2}$, is calculated from the average x- and y-direction gradients of the connecting cells, to determine the gradient in

the direction of maximum slope. The remaining terms in equations 2.39 and 2.40, along with the depth within the first gradient operator of equation 2.42, combine to $d_o^{5/3}$ for Manning, $d_o^{3/2}$ for Chezy and $d_o^{3/2}$ for Darcy-Weisbach. Full upstream weighting of this term between the two connecting nodes ensures a monotonic solution, without unphysical oscillations. Further, upstream weighting ensures that flow from a dry node is zero, maintaining the physical reality to the set of governing equations.

3.5 Flow Coupling

The porous medium (and the dual continuum if present) is discretized in three dimensions with either rectangular prisms, triangular prisms or tetrahedra. Two-dimensional rectangular or triangular elements represent the discrete fracture and the overland domains, and one-dimensional line elements represent the wells and the tile drains. When discretizing the domain, nodes forming the fracture or overland elements or nodes forming the well and tile drain elements have to coincide with those on the adjacent porous medium finite volumes, similarly to *Sudicky et al.* [1995]. Because the fracture elements are generated such that they correspond to planes that intersect three or 4 nodes in the three-dimensional rectangular blocks or prisms, the nodes comprising the fracture elements are therefore common to nodes comprising the porous matrix elements. The commonality of these nodes thus ensures the continuity of hydraulic head at the fracture matrix interface. Also, by superimposing the contributions at each node from both element types, there is no need to explicitly calculate the fluid leakage terms appearing in the discretized equations. For the overland flow nodes, a choice of common or dual nodes is offered, but the overland flow nodes nevertheless have to coincide with porous medium nodes. Additionally, nodes forming the 3D dual continuum domain coincide with the porous medium nodes, but they form duplicate nodes since the dual node approach is used.

With the common node approach, the matrix contributions arising from the discretization of the discrete fracture or the overland nodes, as well as matrix contributions from the well nodes and tile drain nodes are superimposed onto those stemming from the discrete form of porous medium equation. Continuity in pressure head is therefore ensured between the different domains, which avoids the need for a direct evaluation of the exchange fluxes between the porous medium elements and the other domains.

As an example, we write below the final discretized matrix equation when discrete fractures are located in the porous domain. Using discretized equation (3.23), we write the coupling term Γ_f as

$$\Gamma_f^{L+1} a_i = \left[(S_{wf})_i^{L+1} - (S_{wf})_i^L \right] \frac{w_f a_i}{\Delta t} - w_f \sum_{j \in \eta_{f_i}} (\lambda_f)_{(ij+1/2)}^{L+1} \gamma_{f_{ij}} (h_{f_j}^{L+1} - h_{f_i}^{L+1}) \qquad (3.47)$$

Replacing the expression for the coupling term into porous medium equation 3.18 gives the following global equation

$$\left[(S_s S_w \psi + \theta_s S_w)_i^{L+1} - (S_s S_w \psi + \theta_s S_w)_i^L \right] \frac{w_m v_i}{\Delta t} + \left[(S_{wf})_i^{L+1} - (S_{wf})_i^L \right] \frac{w_f a_i}{\Delta t} =$$

$$\sum_{j\in\eta_i}(\lambda)^{L+1}_{(ij+1/2)}\gamma_{ij}(h_j^{L+1}-h_i^{L+1})-\left(\sum\Gamma_{\text{ex}}^{L+1}\right)v_i\pm Q_i^{L+1}+$$
$$w_f\sum_{j\in\eta_{f_i}}(\lambda_f)^{L+1}_{(ij+1/2)}\gamma_{f_{ij}}(h_{f_j}^{L+1}-h_{f_i}^{L+1}) \tag{3.48}$$

In equation 3.48, we assume that $h_i = h_{f_i}$ for node that are common to the porous medium and fracture domains, which ensures continuity. The exact value of the fluid exchange between the domains, Γ_f, is therefore not computed explicitly prior to solution, but the exchange can be back-calculated during post-processing of results by evaluating equation (3.47) at the desired nodes.

Similarly to the approach shown here for superposition of 2 domains, the model allows superposition of all flow domains by adding the relevant discretized equations and assuming continuity of head.

In the next section, we present coupling for the dual node approach, which is the only approach currently available for coupling the porous medium and second subsurface continuum, and is one of the two options available for coupling the porous medium and overland flow.

3.5.1 Porous Medium - Overland Coupling

The overland flow equation is solved on a 2-D topological connectivity of grids. The finite-element mesh for 2-D areal overland flow varies spatially in the vertical direction to conform with land surface topography. The overland flow mesh is stacked upon a subsurface grid when solving for both domains (i.e. the x- and y-locations of nodes are the same for each layer of nodes), as shown in Figure 3.1. **rgm Ed says should modify figure so to be FE instead of block-centred FD** The equations applied to the 2-D areal overland flow domain are the vertically-averaged Saint Venant equations for overland flow. The diffusion-wave approximation to the Saint Venant equation is solved in two-dimensions by this module.

These equations are coupled with the 3-D variably saturated subsurface flow equation via superposition or via leakage through a surficial skin layer. Fully implicit coupling of the surface and subsurface flow regimes provides an integral view of the movement of water, as opposed to the traditional division of overland and subsurface regimes. Flux across the land surface is, therefore, a natural internal process allowing water to move between the surface and subsurface flow systems as governed by local flow hydrodynamics, instead of using physically artificial boundary conditions at the interface.

When the subsurface connection is provided via superposition, **HydroSphere** adds the overland flow equation terms to those of the top layer of subsurface nodes. When the subsurface connection is via skin leakance, the overland flow module of **HydroSphere** generates an additional layer of nodes on top of the subsurface domain, and each overland flow node communicates with the first active subsurface flow node directly beneath it to form the subsurface connection. After the subsurface flow equations have been assembled into an implicit system of matrix equations, the 2-D areal overland flow equations are added

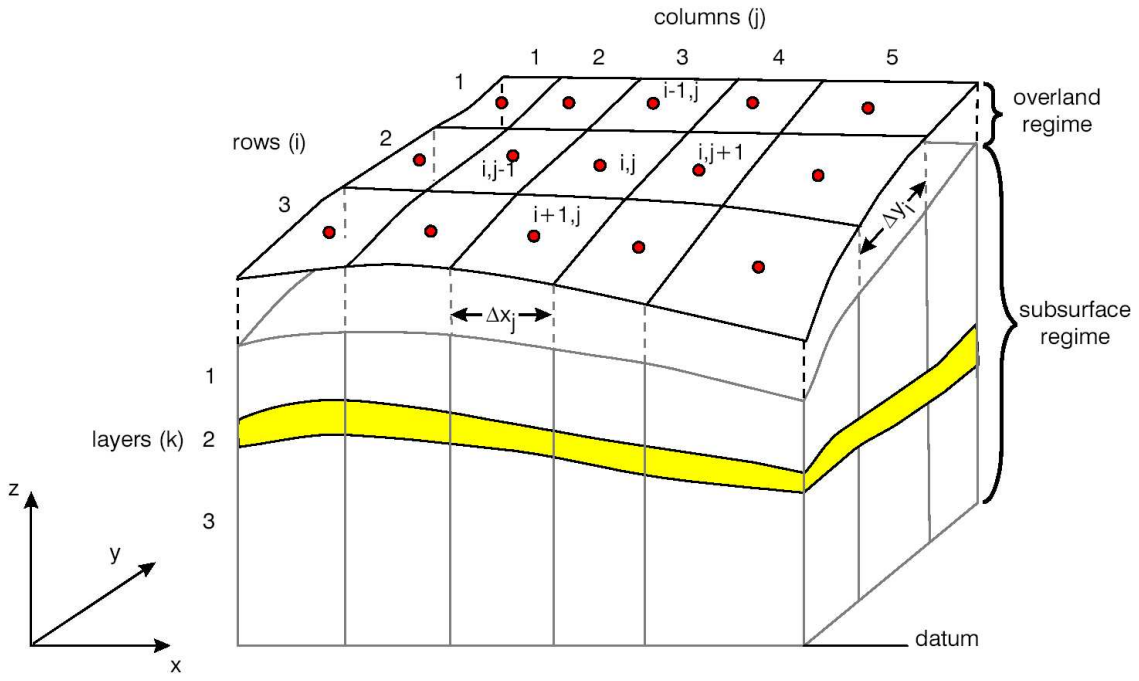


Figure 3.1: Spatial Discretization of the Overland Flow System and its Connection to the Subsurface.

to the matrix along with their subsurface interactions, and the fully-integrated flow system is solved at each time step. This provides significant robustness and accuracy over flux linkage techniques, often used when conjunctive modeling is required for the surface and subsurface regimes.

The 2-D areal overland flow modules of **HydroSphere** follow the same conventions for spatial and temporal discretizations as those used by the subsurface modules. For the superposition approach to linking the surface and subsurface regimes, the topmost layer of subsurface nodes also represent overland flow and additional spatial discretization is not required. When a skin-layer approach is used, Figure 3.1 shows the spatial discretization conventions of the discrete overland flow system. The grid generated for the subsurface domain is mirrored areally for the overland flow nodes, with overland flow node elevations corresponding to the top elevation of the topmost active layer of the subsurface grid. Note that overland flow node elevations may vary substantially to conform with topography. However, the assumptions of small slope inherent in the diffusion-wave equation will not allow for modeling of inertial effects.

Temporal discretization is fully implicit to be consistent with the subsurface modules. Additional computational speed is achieved by following an adaptive explicit/implicit solution scheme.

The set of governing equations (3.42) are assembled into the topmost layer of nodes of the global matrix for subsurface flow in **HydroSphere** for the superposition alternative or into

an additional layer of nodes generated to represent the overland flow surface for the skin-layer case, which further requires the subsurface flow connection term K_G to be added to the left hand side of the respective row G, in the column that connects it to the overland flow node.

Note that the interaction flux term q_{go} is not needed for the superposition scheme, but is required when the surface and subsurface are coupled via a skin leakance. For this case, the interaction flux q_{go} will also be added to the right hand side of the subsurface flow equation 3.18.

3.6 Flow Boundary Conditions

3.6.1 Subsurface flow

Boundary conditions for subsurface flow include the following: first-type (Dirichlet) boundaries of prescribed hydraulic head, areal infiltration or recharge, source/sinks, evapotranspiration and seepage faces.

The source/sink term in (3.18) can be manipulated in order to impose prescribed head boundary conditions [*Forsyth*, 1988; *Kropinski*, 1990]. To assign a prescribed pressure head, ψ_b, to a portion of the domain, the source/sink term becomes

$$Q_i = K_{ij} k_{rw} W_i (\psi_b - \psi_i) \tag{3.49}$$

where W_i is a number large enough (e.g. 10^{20}) to ensure that $\psi_i = \psi_b$ when the assembled system of equations for all nodes is solved. This can be viewed as injecting or withdrawing sufficient fluid at node i to maintain the prescribed pressure head.

Seepage faces represent boundaries that require special treatment and a variety of methods have been suggested to implement such boundaries [*Neuman*, 1973; *Cooley*, 1983]. The method used here is from *Forsyth* [1988] and requires that the approximate location of the seepage face be known a priori. The appropriate form of the source/sink term at the nodes forming the seepage face will be:

$$\begin{aligned} Q_i = &\quad K^* k_{rw} W_i \left(\psi_{atm} - \psi_i\right) && \psi_i > \psi_{atm} \\ = &\quad 0 && \psi_i < \psi_{atm} \end{aligned} \tag{3.50}$$

where K^* is the component of the hydraulic conductivity tensor normal to the seepage face. The above expression allows seepage only when the pressure in the medium is greater than the atmospheric pressure, ψ_{atm}.

3.6.2 Overland flow

Boundary conditions to the overland flow system include the following: first-type (Dirichlet) boundaries of prescribed water elevation, rainfall rate, source/sinks, evaporation, zero-depth gradient and critical-depth conditions.

First-type boundary conditions are implemented in an identical manner to what is described for subsurface flow in section 3.6.1 above.

Areal rainfall (volumetric inflow over an area) is implemented as an input water flux multiplied by the contributing area. **rgm Ed feels the following is a bit out of place here** Note that a mix-and-match domain can be created for modeling, if the user does not wish to solve the overland flow part of the hydrologic system in certain regions of the domain. Such overland flow elements may be inactivated and recharge is applied to the underlying active subsurface nodes, when this occurs.

Sources/Sinks are applied as net fluxes to the overland flow nodes that receive them. Sinks are constrained by the physical property that water depth cannot be negative (i.e. water cannot be extracted when the water level is below bed elevation). If this condition occurs at any solution iteration, only as much water is withdrawn as to not violate this constraint. A further constraint is that injection should also be restricted at sink nodes. Thus, the sink strength should only reduce to zero under limited supply conditions and not become a negative sink (i.e. a source).

Evaporation is applied as an areal sink to an overland flow node, subject to similar non-negative depth constraints as discussed for sinks, above. The overland-flow-evaporation module is a basic package which needs to be used in conjunction with an evapotranspiration module for the subsurface. **rgm Ed questions this bit** Thus, the two processes of evaporation and subsurface evapotranspiration can be individually modeled as separate components, without resorting to complicated models for evapotranspiration.

Zero-depth gradient and critical depth boundary conditions are implemented to simulate conditions at the lower boundaries of a hill slope. Zero-depth gradient (ZDG) condition forces the slope of the water level to equal to the bed slope which is provided by the user at this boundary. The discharge, Q_o, at the zero-depth gradient boundary is given for the Manning equation by:

$$Q_o = \frac{1}{n_i} d_o^{5/3} \sqrt{S_o} \tag{3.51}$$

for the Chezy Equation by

$$Q_o = C_i d_o^{3/2} \sqrt{S_o} \tag{3.51}$$

and for the Darcy-Weisbach relation by

$$Q_o = \sqrt{\frac{8g}{f_i}} d_o^{3/2} \sqrt{S_o} \tag{3.51}$$

where Q_o is the flux per unit width, i is the direction of the zero-depth gradient discharge ($i = x$ in the x-direction and $i = y$ in the y-direction). n_i is Manning roughness in the

direction i, C_i is the Chezy coefficient in direction i, f_i is the friction factor along direction i, and S_o is the bed slope at the zero-depth gradient boundary.

Critical depth (CD) condition forces the depth at the boundary to be equal to the critical depth. The discharge Q_o per unit width at the critical depth boundary is given by:

$$Q_o = \sqrt{g d_o^3} \tag{3.52}$$

3.7 Discretized Subsurface Transport Equations

3.7.1 Porous Medium

When the transport equation is linear, which is the case except when a flux limiter is used for the advective term in order to minimize adverse numerical dispersion, its solution is not as involved as that for the non-linear cases. We present here a discretization scheme for the transport equations based on the control volume finite element method presented in section (3.2). The type of elements used for transport are identical to those used for the flow problem and the choice of superposition of several domains (common node approach) is also given, where elements representing one domain are superimposed onto the element representing a second domain, as performed for the flow equation. This ensures the continuity of concentration at the domain-to-domain interface and avoids the need to explicitly determine the solute mass exchange terms involving Ω_{ex} in the governing transport equations. When a dual continuum is simulated, the dual node approach is used to represent the interaction between the porous medium and the dual continuum.

We present here the standard discretized equation for the 3D porous medium, obtained from the application of the control volume finite element method **rgm Sorab says is unnecessary to expand by Darcy's law in transport equation - just write v. Same comment for equations of the other domains. This is not a compositional simulation where you want the term h and C to be expanded by Newton, so it is just complicating to write the equation that way.**

$$\left[(\theta_s S_w RC)_i^{L+1} - (\theta_s S_w RC)_i^L\right] \frac{w_m v_i}{\Delta t} = \sum_{j\in\eta_i} (C)_{(ij+1/2)}^{L+1} (\lambda)_{(ij+1/2)}^{L+1} \gamma_{ij} (h_j^{L+1} - h_i^{L+1}) +$$

$$\sum_{j\in\eta_i} \chi_{ij} (C_j^{L+1} - C_i^{L+1}) + (Q_i C_{ups})_i^{L+1} + \left[(R\lambda C_i)_{par} - (\theta_s S_w R\,\lambda\, C)_i + \sum \Omega_{\text{ex}}^{L+1}\right] v_i \tag{3.53}$$

where

$$\begin{aligned} C_{ups} &= C_i \quad \text{if} \quad Q_i < 0 \\ &= C_{\text{inflow}} \quad \text{if} \quad Q_i > 0 \end{aligned} \tag{3.54}$$

and where C_{inflow} is the specified source inflow concentration (recall that Q_i is the source/sink term used to represent boundary conditions).

The term $C_{(ij+1/2)}$ in equation (3.53) depends on the type of advective weighting used. For central weighting

$$C_{(ij+1/2)} = \frac{C_i + C_j}{2} \tag{3.55}$$

Upstream weighting gives

$$\begin{aligned} C_{(ij+1/2)} = C_{ups} = C_j \quad \text{if} \quad \gamma_{ij}\,(h_j - h_i) > 0 \\ C_{(ij+1/2)} = C_{ups} = C_i \quad \text{if} \quad \gamma_{ij}\,(h_j - h_i) < 0 \end{aligned} \tag{3.56}$$

A TVD type flux limiter is also available to evaluate $C_{(ij+1/2)}$ according to [*Van Leer*, 1974; *Unger et. al.*, 1996]

$$C_{(ij+1/2)} = C_{ups} + \sigma(r_{ij}) \left(\frac{C_{dwn} - C_{ups}}{2} \right) \tag{3.57}$$

where C_{dwn} is the concentration of the downstream node between i and j. The smoothness sensor r_{ij} is given by

$$r_{ij} = \left(\frac{C_{ups} - C_{i2ups}}{\|P_{ups} - P_{i2ups}\|} \right) \left(\frac{C_{dwn} - C_{ups}}{\|P_{dwn} - P_{ups}\|} \right)^{-1} \tag{3.58}$$

where C_{i2ups} is the second upstream node between i and j, and P_{ups}, P_{i2ups}, P_{dwn} are the position vectors of the upstream, second upstream and downstream nodes, respectively.

A van Leer flux limiter is used in equation (3.57) such that

$$\begin{aligned} \sigma(r) = 0 \qquad \text{if} \quad r \leq 0 \\ \sigma(r) = (2r)/(1+r) \quad \text{if} \quad r > 0 \end{aligned} \tag{3.59}$$

We further define

$$\chi_{ij} = -\int_v \nabla N_i \cdot \theta_s S_w^{N+1} \mathbf{D} \cdot \nabla N_j \, dv \tag{3.60}$$

Other terms in equation (3.53) are as defined for the discrete flow equation (3.18).

3.7.2 Discrete Fractures

The discretized 2D transport equation in fractures is

$$\begin{aligned} \left[(S_{wf} R_f C_f)_i^{L+1} - (S_{wf} R_f C_f)_i^{L} \right] \frac{w_f a_i}{\Delta t} = w_f \sum_{j \in \eta_{f\,i}} C_{f\,(ij+1/2)}^{L+1} \lambda_{f\,(ij+1/2)}^{L+1} \gamma_{f\,ij} (h_{f\,j}^{L+1} - h_{f\,i}^{L+1}) + \\ \sum_{j \in \eta_{f\,i}} \chi_{f\,ij} (C_{f\,j}^{L+1} - C_{f\,i}^{L+1}) + \left[(R_f \lambda C_{f\,i})_{par} - (S_{wf} R_f \, \lambda \, C_f)_i + \Omega_f^{L+1} \right] a_i \end{aligned} \tag{3.61}$$

where

$$\chi_{f\,ij} = -\int_a \nabla N_i \cdot S_{wf}^{N+1} \mathbf{D_f} \cdot \nabla N_j \, da \tag{3.62}$$

and where the interface permeability can be either central or upstream weighted or given by a TVD flux limiter, as shown for the porous medium equation in section (3.7.1). Other terms in equation (3.61) are as defined for the discrete fracture flow equation (3.23).

3.7.3 Double-porosity

The discrete equation for transport into the immobile region of a double-porosity domain is given by:

$$\left[(\theta_{\text{Imm}}C_{\text{Imm}})_i^{L+1} - (\theta_{\text{Imm}}C_{\text{Imm}})_i^{L}\right]\frac{v_i}{\Delta t} = \Omega_{\text{Imm}}^{L+1}v_i \tag{3.63}$$

3.7.4 Isotope fractionation

The discrete equation for isotopic fractionation into the immobile (solid) region is given by:

$$\left[(C_{\text{Imm}})_i^{L+1} - (C_{\text{Imm}})_i^{L}\right]\frac{v_i}{\Delta t} = \frac{\Omega_{\text{Imm}}^{L+1}}{x_r}v_i \tag{3.64}$$

3.7.5 Dual Continuum

For the dual continuum, the discretized 3D transport obtained after using the control volume finite element method is

$$\left[(\theta_{sd}S_{wd}R_dC_d)_i^{L+1} - (\theta_{sd}S_{wd}R_dC_d)_i^{L}\right]\frac{w_dv_i}{\Delta t} = \sum_{j\in\eta_{di}}(C_d)_{(ij+1/2)}^{L+1}(\lambda_d)_{(ij+1/2)}^{L+1}\gamma_{dij}(h_{dj}^{L+1}-h_{di}^{L+1})+$$

$$\sum_{j\in\eta_{di}}\chi_{dij}(C_{dj}^{L+1}-C_{di}^{L+1}) + (Q_dC_{ups})_i^{L+1} + \left[(R_d\lambda_dC_{di})_{par} - (\theta_{sd}S_{wd}R_d\,\lambda_d\,C_d)_i + \Omega_d^{L+1}\right]v_i \tag{3.65}$$

All terms in equation 3.65 are defined in a manner analogous to the discretized porous medium transport equation shown in section (3.7.1).

3.7.6 Wells

One-dimensional transport along a well is described by the following discretized equation

$$\left[(C_w)_i^{L+1} - (C_w)_i^{L}\right]\frac{\pi r_s^2 l_i}{\Delta t} = \sum_{j\in\eta_{wi}}(C_w)_{(ij+1/2)}^{L+1}(\lambda_w)_{(ij+1/2)}^{L+1}\gamma_{wij}(h_{wj}^{L+1} - h_{wi}^{L+1})+$$

$$\sum_{j\in\eta_{wi}}\chi_{wij}(C_{wj}^{L+1} - C_{wi}^{L+1}) + (Q_wC_{ups})_i^{L+1} + \left[\pi r_s^2\,(\lambda C_{wi})_{par} + \Omega_w^{L+1}\right]l_i \tag{3.66}$$

All terms in equation 3.66 are defined in a manner analogous to the discretized porous medium transport equation shown in section (3.7.1).

3.7.7 Tile Drains

One-dimensional transport in a tile drain is described by the following discretized equation

$$\left[(C_t)_i^{L+1} - (C_t)_i^L\right] \frac{Al_i}{\Delta t} = \sum_{j \in \eta_{t_i}} (C_t)_{(ij+1/2)}^{L+1} (\lambda_t)_{(ij+1/2)}^{L+1} \gamma_{t_{ij}} (h_{t_j}^{L+1} - h_{t_i}^{L+1}) +$$

$$\sum_{j \in \eta_{t_i}} \chi_{t_{ij}} (C_{t_j}^{L+1} - C_{t_i}^{L+1}) + (Q_t C_{ups})_i^{L+1} + \left[A \left(\lambda C_{ti}\right)_{par} + \Omega_t^{L+1}\right] l_i \tag{3.67}$$

rgm Rene, should all lambda above have subscript t? All terms in equation 3.67 are defined in a manner analogous to the discretized porous medium transport equation shown in section (3.7.1).

3.7.8 Overland Domain

For the overland flow domain, the governing transport equation is discretized as:

$$\left[(\phi_o h_o R_o C_o)_i^{L+1} - (\phi_o h_o R_o C_o)_i^L\right] \frac{a_i}{\Delta t} = \sum_{j \in \eta_{oi}} C_{o_{(ij+1/2)}}^{L+1} \cdot q_{o_{(ij+1/2)}}^{L+1}$$

$$+ \sum_{j \in \eta_{oi}} \chi_{oij} (C_{oj}^{L+1} - C_{oi}^{L+1}) + \left[(\phi_o h_o R_o \lambda C_{oi})_{par} - (\phi_o h_o R_o \lambda C_o)_i + \Omega_o^{L+1}\right] a_i \tag{3.68}$$

where

$$\chi_{oij} = -\int_a \nabla N_i \cdot (\phi_o h_o)^{N+1} D_o \cdot \nabla N_j da \tag{3.69}$$

and where the interface flux is obtained from the solution to the associated flow equation. The term $C_{o_{(ij+1/2)}}^{L+1}$ may be treated in a mid-point or upstream weighted manner, or by using the TVD flux limiter as discussed earlier in section 3.7.1.

3.8 Transport Coupling

Coupling of the various domains for solute transport is done in a manner similar to the coupling used for fluid flow. When the common node approach is used, the discretized equations for the coupled domains are added, and continuity of concentration is assumed at the nodes shared by the coupled domains. The exchange term Ω_{ex} does not need to be explicitly evaluated but can be back-calculated after solution.

For the dual node approach, currently available for coupling between the porous medium and dual continuum domains, the exchange term is explicitly evaluated and there is no assumption of equilibrium or continuity between the concentrations of the two domains. For the dual continuum approach, the coupling term is evaluated according to

$$\Omega_d^{L+1} v_i = \left[-(u_m C)_i^{L+1} - (u^* C^*)_i^{L+1}\right] v_i \tag{3.70}$$

When the dual-node approach is used to couple the overland and subsurface domains, the coupling term may be expressed as

$$\Omega_o^{L+1} = C_{ups}^{L+1}\Gamma_o \tag{3.71}$$

where $C_{ups}^{L+1} = C_o^{L+1}$ when the flux is from the overland to the subsurface system and $C_{ups}^{L+1} = C^{L+1}$ when the flux is from the subsurface to the overland system.

3.9 Transport Boundary Conditions

3.9.1 Subsurface Transport

Boundary conditions for transport include the following: first-type (Dirichlet) boundaries, second type (mass flux), third type (total flux) and each can be input as time-dependent quantities.

3.10 Numerical Techniques

3.10.1 Matrix Solution

Discretized flow equations, such as equations (3.18), (3.23), (3.28), (3.32), (3.37), (3.42), or any combination of these equations, forms a matrix system of the form

$$Ax = b \tag{3.72}$$

where x is the unknown, A is a matrix of coefficients and b is a force vector. When flow is fully-saturated, the discretized equations are linear and the matrix of coefficient A is also linear. A direct solution of the matrix equation is then possible. The resulting system of equations can be very large for a fully three-dimensional field-scale problem. A fast and efficient matrix solver is therefore critical in order to reduce core memory requirements and CPU time. The preconditioned ORTHOMIN solver has been shown to be very efficient and robust for solving large systems of equations [*Behie and Forsyth*, 1984] and it has therefore been implemented to solve the system of equations. The preconditioning chosen consists of performing an ILU decomposition of the assembled coefficient matrix without altering its original sparsity pattern [*Behie and Forsyth*, 1984]. The solver has the capability, however, to perform a higher-order ILU decomposition of the coefficient matrix, where additional steps of a Gaussian elimination are performed, resulting in the addition of extra bands to the original matrix. Performing a higher-order decomposition results in a better-conditioned decomposed matrix, which can improve the convergence rate of the solver, but at the expense of increased storage requirements and additional computation time.

An option to use either a finite difference or finite element discretization, as described earlier, has also been implemented for the transport solution. Experience has indicated that for discretely-fractured porous media in which the matrix has low permeability such

that mechanical dispersion in the matrix is weak relative to molecular diffusion, the finite element and finite difference representations give essentially identical results. This suggests that the cross-derivative terms in the transport equation are small compared to the terms that are retained in the finite difference approach.

As in the case of the flow problem, a mass balance is performed to assess the accuracy of the solution. A procedure similar to that described by *Huyakorn and Pinder* [1983] is used. The transport matrix equations are solved using the same ILU-preconditioned ORTHOMIN solver [*Behie and Forsyth*, 1984] as is used for the flow problem.

3.10.2 Newton-Raphson Method

The Newton-Raphson technique is used to linearize the non-linear equations arising from discretization of variably-saturated subsurface or overland flow equations. The method is described in *Huyakorn and Pinder* [1983] and is reproduced here to demonstrate the advantage of using the control volume finite element approach over a conventional Galerkin method. To illustrate the method, we apply it to equation (3.16), which is rewritten in the following way:

$$f_i^r = \left\{[\theta_s S_w]_i^{L+1} - [\theta_s S_w]_i^L\right\} \frac{v_i}{\Delta t} - \sum_{j \in \eta_i} (\lambda)_{(ij+1/2)}^{L+1} \gamma_{ij} (h_j^{L+1} - h_i^{L+1}) - Q_i^{L+1} \tag{3.73}$$

where r represents the iteration level. Application of the Newton method to (3.73) produces the following matrix equation:

$$F_{ij}^r \Delta\psi_j^{r+1} = -f_i^r \tag{3.74}$$

which can be solved with the same preconditioned iterative solver used for linear matrix equations, since the Jacobian matrix F_{ij} is linear.

In (3.74), the Jacobian matrix, F_{ij}^r, is defined as:

$$F_{ij}^r = \frac{\partial f_i^r}{\partial \psi_j^r} \tag{3.75}$$

and vector f_i^r represent the residual of the discretized equation. The iteration process is carried out repeatedly until the change in the pressure head, $\Delta\psi_j^{r+1}$, or the residual of the equation, f_i^r, becomes less than a specified tolerance at all the nodes. It is important that the evolution of the residual, f_i^r, be monitored during iteration to ensure proper convergence.

Full Newton iteration can be computationally expensive mainly because of the need to evaluate the Jacobian matrix. It is therefore highly desirable to implement a scheme capable of evaluating it in an efficient manner. One option is to evaluate the Jacobian numerically [*Forsyth and Simpson*, 1991; *Kropinski*, 1990]. Term by term evaluation of the Jacobian can be represented by [*Kropinski*, 1990]:

$$\frac{\partial f_i^r}{\partial \psi_i^r} \approx \frac{f_i^r(\psi_i^r + \epsilon) - f_i^r(\psi_i^r)}{\epsilon} \tag{3.76}$$

where ϵ represents a small numerical shift in the pressure head value.

Obviously, from (3.76), numerical differentiation requires more than one function evaluation for each term; however, it can be shown that not all terms in the Jacobian will require two function evaluations. The form of the discretized equation (3.73) also makes it intuitively easy to evaluate the Jacobian numerically. This is because the fluid flow terms appearing in the summation in (3.73) depend only on nodes i and j. *Forsyth and Simpson* [1991] and *Kropinski* [1990] give a detailed procedure for the Jacobian evaluation and it is reproduced below for completeness.

The diagonal term of the Jacobian for node i can be determined from (3.73) and (3.75) as:

$$\frac{\partial f_i^r}{\partial \psi_i^r} = \frac{\partial}{\partial \psi_I^r}\left\{[\theta_s S_w]_i^r - [\theta_s S_w]_i^L\right\}\frac{v_i}{\Delta t} - \sum_{J\in\eta_I}\frac{\partial}{\partial \psi_I^r}(\lambda)^r_{(ij+1/2)}\gamma_{ij}(h_j^r - h_i^r) - \frac{\partial Q_I^r}{\partial \psi_I^r} \tag{3.77}$$

The entries in column i of the Jacobian will be, excluding the diagonal:

$$F_{ji} = \frac{\partial f_j^r}{\partial \psi_i^r} \tag{3.78}$$

where $j \in \eta_i$. As stated previously, the only term in f_j^r depending on ψ_i^r will be the one representing flow from node i to node j. Therefore:

$$\frac{\partial f_j^r}{\partial \psi_i^r} = -\frac{\partial}{\partial \psi_i^r}(\lambda)^r_{(ji+1/2)}\gamma_{ji}(h_i^r - h_j^r) \tag{3.79}$$

Because of local conservation of mass, the fluid flow term between i and j appearing in the equation for node i will be similar to the one appearing in equation for node j. Therefore we have

$$\begin{aligned}\lambda_{ij} &= \lambda_{ji}\\ \Gamma_{ij} &= \Gamma_{ji}\end{aligned} \tag{3.80}$$

Using (3.80), (3.79) becomes:

$$\frac{\partial f_j^r}{\partial \psi_i^r} = \frac{\partial}{\partial \psi_i^r}(\lambda)^r_{(ij+1/2)}\gamma_{ij}(h_j^r - h_i^r) \tag{3.81}$$

The right hand side of (3.81) is also found in the summation appearing in (3.77). The expression for the diagonal term (3.77) can therefore be expressed as:

$$\frac{\partial f_i^r}{\partial \psi_i^r} = \frac{\partial}{\partial \psi_i^r}\left\{[\theta_s S_w]_i^r - [\theta_s S_w]_i^L\right\}\frac{v_i}{\Delta t} - \sum_{j\in\eta_i}\frac{\partial f_j^r}{\partial \psi_i^r} - \frac{\partial Q_i^r}{\partial \psi_i^r} \tag{3.82}$$

which shows that the evaluation of the diagonal term for node i incorporates all the terms appearing in column i of the Jacobian. The Jacobian can therefore be constructed by only evaluating the diagonal terms and subsequently filling in the off-diagonal terms in a column-wise fashion.

Forsyth and Simpson [1991] and *Kropinski* [1990] show that for n unknowns and with the summation in (3.73) extending from unity to η_i, the building of the Jacobian requires $n(2 + 2\eta_i)$ function evaluations. The Picard iteration scheme, on the other hand, requires at least $n(1 + \eta_i)$ function evaluations to compute the residual, which is seen to be only a factor of two less than the more robust Newton method. Numerical differentiation is also attractive because it allows easy use of tabular data to represent arbitrary constitutive relations should analytical expressions be unavailable.

3.10.3 Primary Variable Substitution

Forsyth et. al. [1994] discuss the use of a saturation based form of equation 2.1 which has good convergence properties in terms of nonlinear iterations when compared with a pressure based method. Because this method cannot be used in the saturated zone they define a new variable which is essentially the saturation in the unsaturated zone and the pressure head in the saturated zone.

They use full Newton iteration to solve the discrete equation everywhere and, in the case of a constant air phase pressure approximation, which applies here, they simply use a different primary variable in different regions. Note that primary variables are regarded as independent when constructing the Jacobian.

The primary variable at a given node may be switched after every Newton iteration using the following method:

$$\begin{array}{lll} \text{IF} & & \\ & S_i > tol_f & \text{Use } \psi_i \text{ as primary variable at node } i \\ \text{ELSE IF} & & \\ & S_i < tol_b & \text{Use } S_i \text{ as primary variable at node } i \\ \text{ELSE} & & \\ & & \text{Do not change primary variable at node } i \\ \text{ENDIF} & & \end{array} \tag{3.83}$$

with the requirements that

$$tol_f < 1 \tag{3.84}$$

and

$$tol_f \neq tol_b \tag{3.85}$$

3.10.4 Time Stepping

A variable time-stepping procedure similar to the one outlined by *Forsyth and Sammon* [1986] and *Kropinski* [1990] has been incorporated in the solution procedure. After obtaining the solution at time level L, the next time-step is selected according to:

$$\Delta t^{L+1} = \frac{S_{wmax}}{max \mid S_{w_i}^{L+1} - S_{w_i}^{L} \mid} \Delta t^{L} \tag{3.86}$$

where S_{wmax} is the maximum change in water saturation allowed during a single time-step. This implementation of variable time-stepping therefore allows the use of increasingly larger time steps if the dependent variable does not experience drastic changes. The time-step selection can also be based on changes in pressure head by using an expression identical to (3.86). It is also recognized that the number of Newton iterations taken to achieve convergence is a good indicator of the suitability of the current time-step size. If the number of iterations exceeds a specified maximum, IT_{max}, at time level $L+1$ in which the time step is currently Δt, the solution is restarted at time level L and the time-step is reduced,

typically by a factor of two. Overall, the procedure can lead to a very significant reduction in computational effort, especially when the solution is desired only at a few widely-spaced target times (see *Therrien and Sudicky* [1996]).

3.10.5 Solution Procedures

The solution methodology for 2-D areal overland flow is embedded into the time looping of the solution methodology for the subsurface calculations of **HydroSphere** (i.e., at each iteration, assembly of the matrix of flow equations for the subsurface is followed by assembly of its overland flow equations). The entire implicit system of matrix equations is then solved at each nonlinear iteration until convergence before proceeding to the next time step. Adaptive time-stepping, Newton-Raphson linearization, and under-relaxation formulas used for the solution are discussed in chapter 3 among the sections that document solution to the subsurface equations.

Chapter 4

Verification Examples

4.1 Subsurface Flow

4.1.1 Drawdown in a Theis Aquifer

In order to verify the accuracy of **HydroSphere** in simulating drawdown due to pumping in a Theis aquifer, it was compared with an exact analytical solution. The example is taken from *Freeze and Cherry*, [1979], to which the reader is referred for detailed information regarding the Theis solution. The input parameters for the analytical solution are shown in the Table 4.1.

In the numerical model, a circular grid of 28596 prism elements which was 10000 m in diameter and 1 m thick was generated. The pumping well was simulated with a single vertical line element which was located at the centre of the grid. The discarge was specifed at the lowermost node in the well. A prescribed head boundary condition was maintained at the outer edge of the domain.

For comparison, the same problem was run using the axisymmetric option with a graded mesh consting of 33 block elements, ranging from 0.01 m near the well to 1000 m near the outer boundary.

Drawdown versus time data for a node located 55 m from the pumping well is shown in

Parameter	Value	
Pumping rate	4.0×10^{-3}	m/s
Hydraulic conductivity	0.0023	m/s
Aquifer thickness	1.0	m
Aquifer storativity	7.5×10^{-4}	
Radial distance to observation point	55	m

Table 4.1: Parameter values for simulation of Theis problem

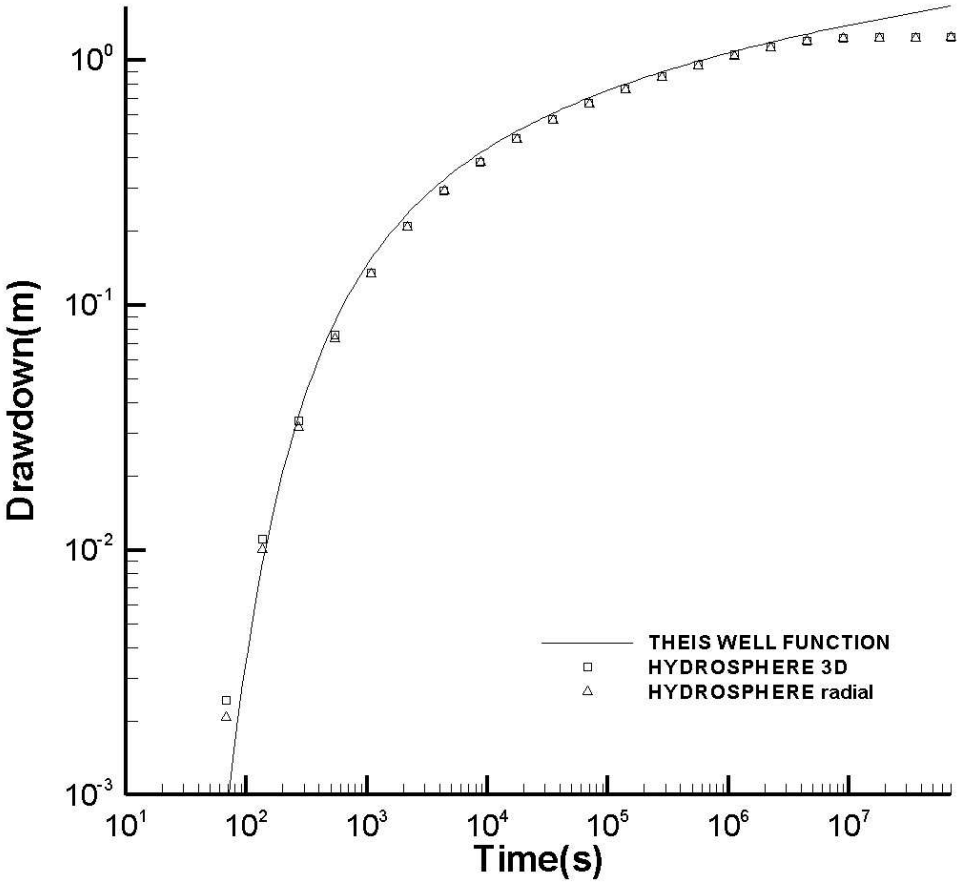


Figure 4.1: Results for pumping in a Theis aquifer

figure 4.1.

The results from both the full 3D grid and axisymmetric grid are very close to those obtained from the analytical solution. Both solutions drop below the Theis solution at late time due to the influence of the constant head boundary condition. Although both approaches give essentially identical results, the axysymmetric option results in a 2 order-of-magnitude decrease in CPU time.

4.1.2 Unsaturated flow through a column

This verification example consists of one-dimensional transient infiltration in an unsaturated vertical column. The specifications of the problem are taken from *Huyakorn et al.* [1986], Example 4. The physical system is 200 cm long in the vertical (z) direction, with the top face corresponding to the soil surface and the bottom face corresponding to the water table.

ψ(cm)	S_w
-0.01	.333
0.0	1.0

Table 4.2: Water saturation versus pressure head relationship for the unsaturated column example

S_w	K_{rw}
.333	0.0
1.0	1.0

Table 4.3: Relative permeability versus water saturation relationship for the unsaturated column example

The column has dimensions of 50 cm in each of the horizontal (x and y) directions. Initially, the pressure head at the water table is zero, it is -90 cm at the soil surface and equals -97 cm in the remainder of the domain. Infiltration at the rate of 5 cm/day is then applied for a period of 10 days. The saturated hydraulic conductivity of the soil is 10 cm/day and its porosity is 0.45. The constitutive relationships for the soil are given by:

$$k_{rw} = \frac{(S_w - S_{wr})}{(1 - S_{wr})}$$

and:

$$\frac{(\psi - \psi_a)}{(-100 - \psi_a)} = \frac{(1 - S_w)}{(1 - S_{wr})}$$

where the residual saturation, S_{wr}, is 0.333 and the air entry pressure , ψ_a, is 0.0 cm. Substituting these values in the equations given above yields simple linear relationships which can be input to the model in tabular form. The input values of water saturation versus pressure head are shown in Table 4.2. The input values of relative permeability versus water saturation are shown in Table 4.3.:

The column is discretized in three dimensions with 2 nodes in each of the x- and y-directions and 41 nodes in the z-direction. The mesh thus consists of a total of 164 nodes and 40 elements. The time steps are identical to *Huyakorn et al.* [1986] with an initial value equal to 0.1 days, which is increased by a factor of 1.2 until a maximum of 1.0 days is attained. The tolerance on pressure head for the Newton-Raphson iteration is set to 0.01 cm.

Figure 4.2 shows pressure head profiles at 4 different times during the infiltration event from *Huyakorn et al.* [1986]. For comparison, results from **HydroSphere** are also presented. It can be seen that the results are almost identical.

4.1.3 Very Dry Initial Conditions

This verification problem is taken from *Forsyth et. al.*[1995], Example 2, which was developed to compare the performance of numerical simulators for very dry initial conditions.

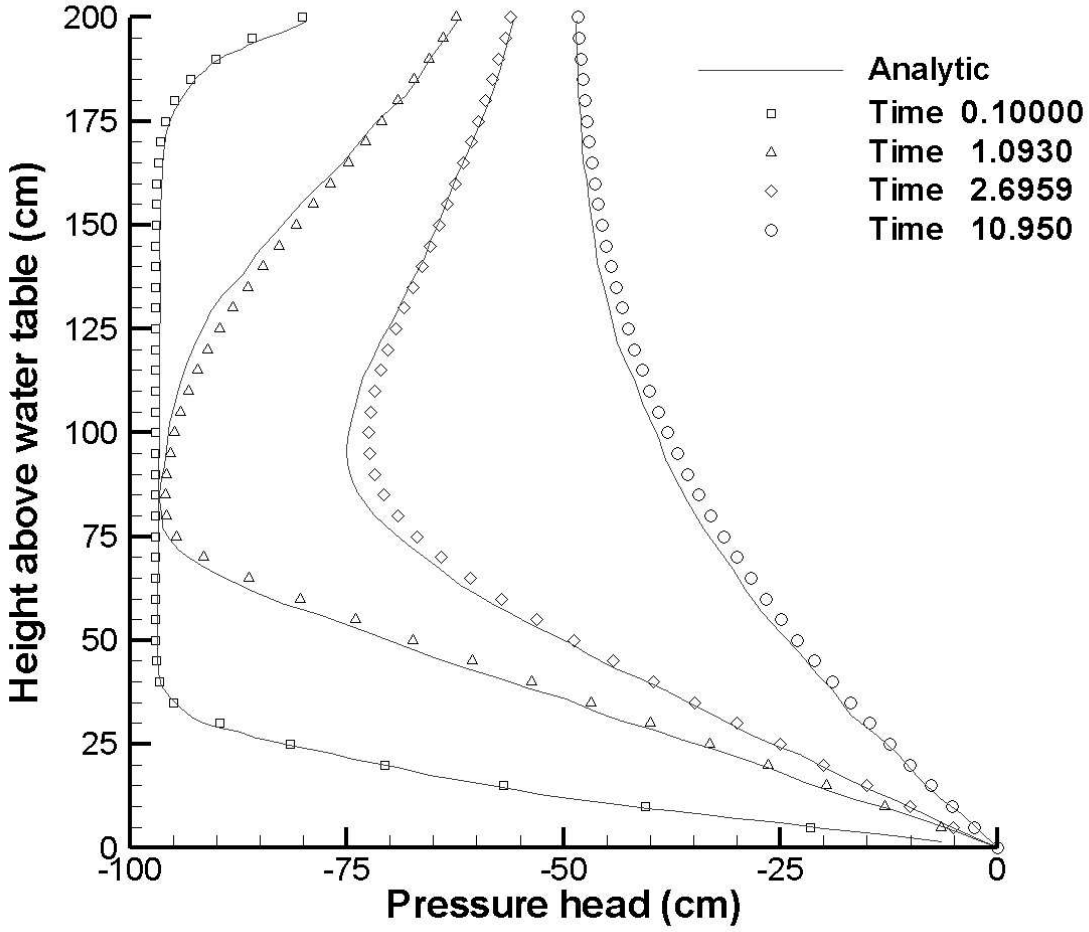


Figure 4.2: Pressure head profiles for the unsaturated flow verification example

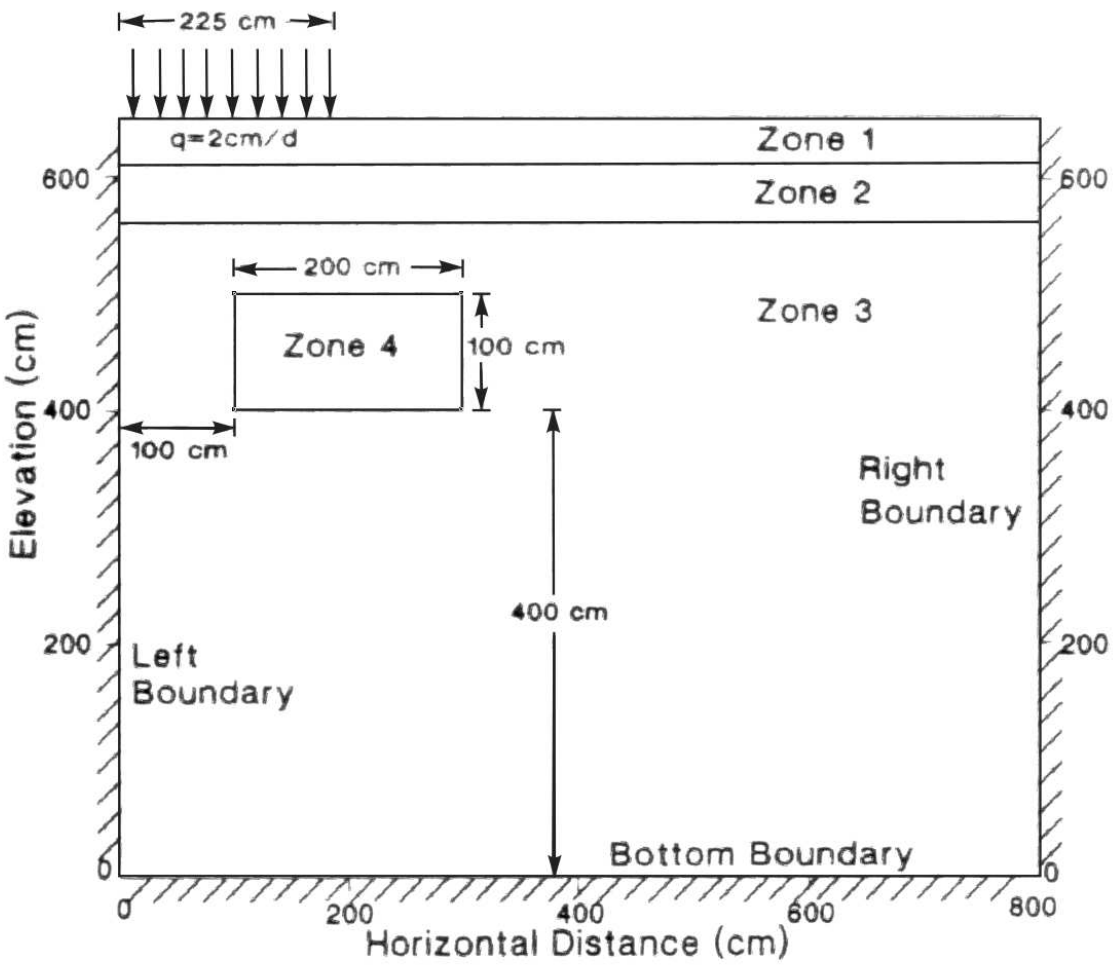


Figure 4.3: Results for very dry initial conditions

The computational domain is shown in Figure 4.3. All boundaries are no flow except for the zone of infiltration at the top left corner. Table 4.4 provides the material properties for the 4 soil zones. They report using a 90 × 21 finite volume grid to discretize the domain, but the exact grid coordinates were unavailable. The initial pressure head was set to -734 cm, and water infiltration occurred for 30 days.

Figure 4.4 compares saturation contours between **HydroSphere** and Forsyth's one phase, central weighting case. The saturation front produced by **HydroSphere** is considerably sharper than that shown by Forsyth.

Parameter	Value	
Soil Zone 1		
porosity, n	0.3680	
permeability, k	9.3×10^{-12}	m^2
van Genuchten parameter, α	0.0334	cm^{-1}
van Genuchten parameter, β	1.982	
residual saturation, S_r	0.2771	
Soil Zone 2		
porosity, n	0.3510	
permeability, k	5.55×10^{-12}	m^2
van Genuchten parameter, α	0.0363	cm^{-1}
van Genuchten parameter, β	1.632	
residual saturation, S_r	0.2806	
Soil Zone 3		
porosity, n	0.3250	
permeability, k	4.898×10^{-12}	m^2
van Genuchten parameter, α	0.0345	cm^{-1}
van Genuchten parameter, β	1.573	
residual saturation, S_r	0.2643	
Soil Zone 4		
porosity, n	0.3250	
permeability, k	4.898×10^{-11}	m^2
van Genuchten parameter, α	0.0345	cm^{-1}
van Genuchten parameter, β	1.573	
residual saturation, S_r	0.2643	

Table 4.4: Material properties for the simulation of *Forsyth et. al.* [1995], example 2

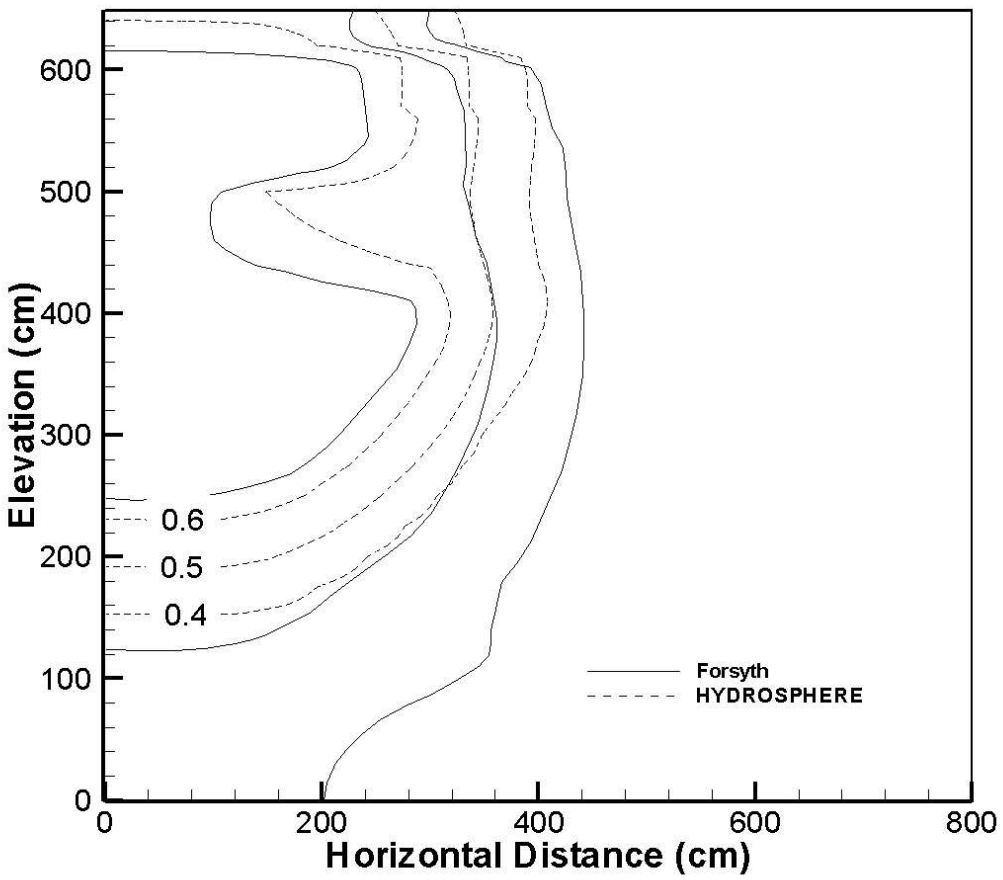


Figure 4.4: Results for very dry initial conditions

4.1.4 Drainage of a fractured tuff column

An example is now presented to verify the variably-saturated flow solution in discretely-fractured porous media. Because the variably-saturated flow equation is nonlinear and analytical solutions are at best approximate, the numerical formulation was verified by comparison to the numerical solution presented by *Wang and Narasimhan* [1985] for an example problem which involves the vertical drainage of a three-dimensional fractured tuff column.

Analytical expressions describing the fracture relative permeability, k_r, saturation, S and effective fracture-matrix flow area, σ, as presented by *Wang and Narasimhan* [1985] have the following forms:

$$k_r(\psi) = \frac{1}{6(4+\beta b_c)}\{[24 - \exp(-\beta b_s)(24 + 24\beta b_s + 12\beta^2 b_s^2 + 4\beta^3 b_s^3 + \beta^4 b_s^4)] + \beta b_c[6 - \exp(-\beta b_s)(6 + 6\beta b_s + 3\beta^2 b_s^2 + \beta^3 b_s^3)]\} \tag{4.1}$$

$$S(\psi) = \frac{1}{2+\beta b_c}\{[2 - \exp(-\beta b_s)(2 + 2\beta b_s + \beta^2 b_s^2)] + \beta b_c[1 - \exp(-\beta b_s)(1 + \beta b_s)]\} \tag{4.2}$$

$$\sigma(\psi) = 1 - \exp(-\beta b_c - \beta b_s)(1 + \beta b_c + \beta b_s) \tag{4.3}$$

where the β values are parameters determined from fracture spacing. The variable b_c is the contact cutoff aperture for the fracture and can be determined by the root of the equation:

$$1 - \exp(-\beta b_c)(1 + \beta b_c) = \omega \tag{4.4}$$

ω being the fraction of contact area for a fracture.

The variable b_s represents the saturation cutoff aperture and is given by:

$$b_s = -\frac{2\gamma \cos\Theta}{\rho g \psi} \tag{4.5}$$

where g is the acceleration due to gravity, ρ is the density of water, γ is the surface tension, Θ represents the angle between the solid and liquid surface and ψ is the pressure head.

The values of the above variables that are used in this simulator are based on the values presented in the *Wang and Narasimhan* [1985] study, and are show in Table 4.5. They examined flow in a fractured porous tuff unit, the Topopah Spring Member at Yucca Mountain, and developed a theory for computing the unsaturated flow properties of fractures which was then applied to this rock unit. Based on observations, they obtained values for all the parameters needed to describe unsaturated flow in the fractured tuff unit. They used an

Parameter	Value	
Fluid density ρ	1000	kg/m^3
Acceleration due to gravity g	9.80665	m/s^2
Surface tension γ	0.07183	kg/s^2
Solid/liquid surface angle Θ	0.0	
Fracture contact area ω	12	%
Horizontal fracture contact cutoff aperture $b_{c(H)}$	0.074	mm
Vertical fracture contact cutoff aperture $b_{c(V)}$	0.057	mm
β_H	0.804×10^4	1/m
β_V	1.04×10^4	1/m

Table 4.5: Parameter values used for *Wang and Narasimhan* [1985] relationships

intrinsic permeability of $1.02 \times 10^{-11} m^2$ for the fractures. Note that they identified two sets of fractures, vertical fractures, for which the subscript V is used, and horizontal fractures denoted by subscript H.

The theoretical expressions developed by *Wang and Narasimhan* [1985] to describe the saturation, relative permeability and effective fracture-matrix flow area for the fractures, as functions of the fluid pressure, were implemented in this model and used for the simulation. The comparison was made for the case where the phase-separation constriction factor, which was used by *Wang and Narasimhan* [1985] to represent the effects of the air phase on the flow of water, was neglected.

Figure 4.5 illustrates the geometry of the physical system. The porous tuff matrix contains three fracture sets, two sets are vertical with a constant fracture aperture equal to 240 μm and one set is horizontal, with a fracture aperture equal to 310 μm. It should be noted that the fractures have not been drawn to scale in Figure 4.5. The fractures partition the matrix into blocks, with each block having dimensions equal to 0.22 m $\times$ 0.22 m $\times$ 0.48 m. A total of 27 such blocks are represented in Figure 4.5. The saturated hydraulic conductivity of the tuff matrix is 3.2×10^{-8} cm/s, its porosity is 0.09 and its specific storage equals 1×10^{-6} m^{-1}. The constitutive relations describing matrix saturation and relative permeability are represented by the van Genuchten relations, (2.5) and (2.6), with $S_{wr} = 9.6 \times 10^{-4}$, $\alpha = 7.027 \times 10^{-3}$ m^{-1}, $m = 0.45$ and $n = 1.818$. The specific storage of the fractures is equal to 4.4×10^{-6} m^{-1}.

Due to the symmetry of the system, *Wang and Narasimhan* [1985] only considered one vertical column bounded by four vertical fractures. For this comparison, only one quarter of this column, i.e. total dimensions equal to 0.11 m $\times$ 0.11 m $\times$ 1.44 m, need be discretized, taking advantage of the horizontal symmetry of the drainage process. The column was discretized using 7 nodes in each of the horizontal directions and 31 nodes in the vertical direction, for a total of 1519 nodes (1080 three-dimensional elements). The nodal spacing used was identical to that reported by *Wang and Narasimhan* [1985]. Each vertical fracture was represented by 180 two-dimensional elements and 36 elements were used to discretize each horizontal fracture.

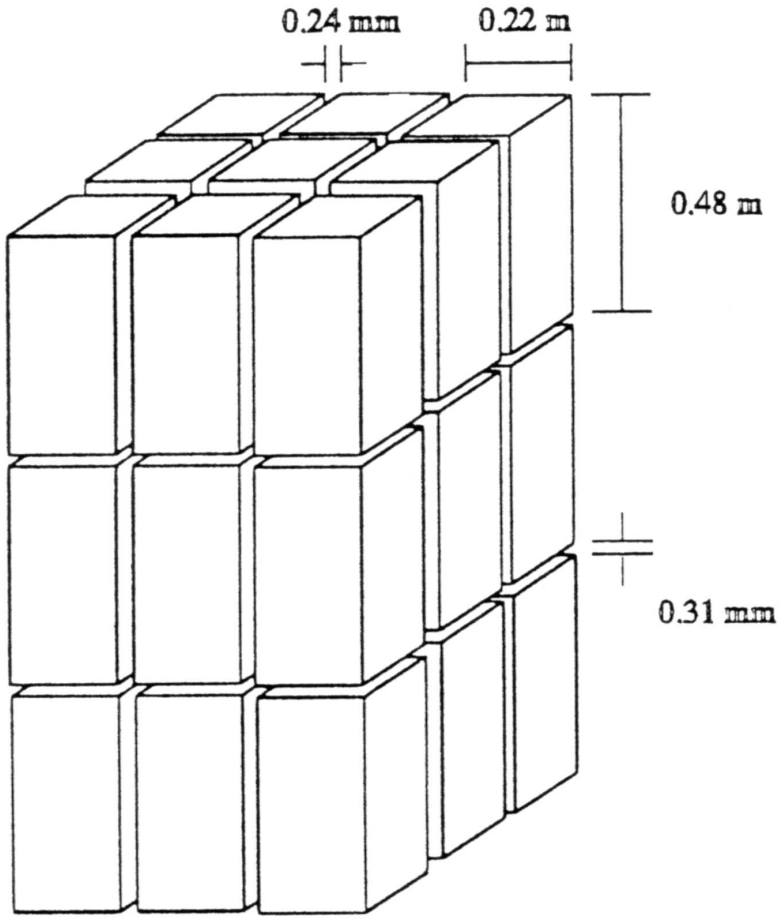


Figure 4.5: Verification example involving fractured porous tuff, from *Wang and Narasimhan*, [1985].

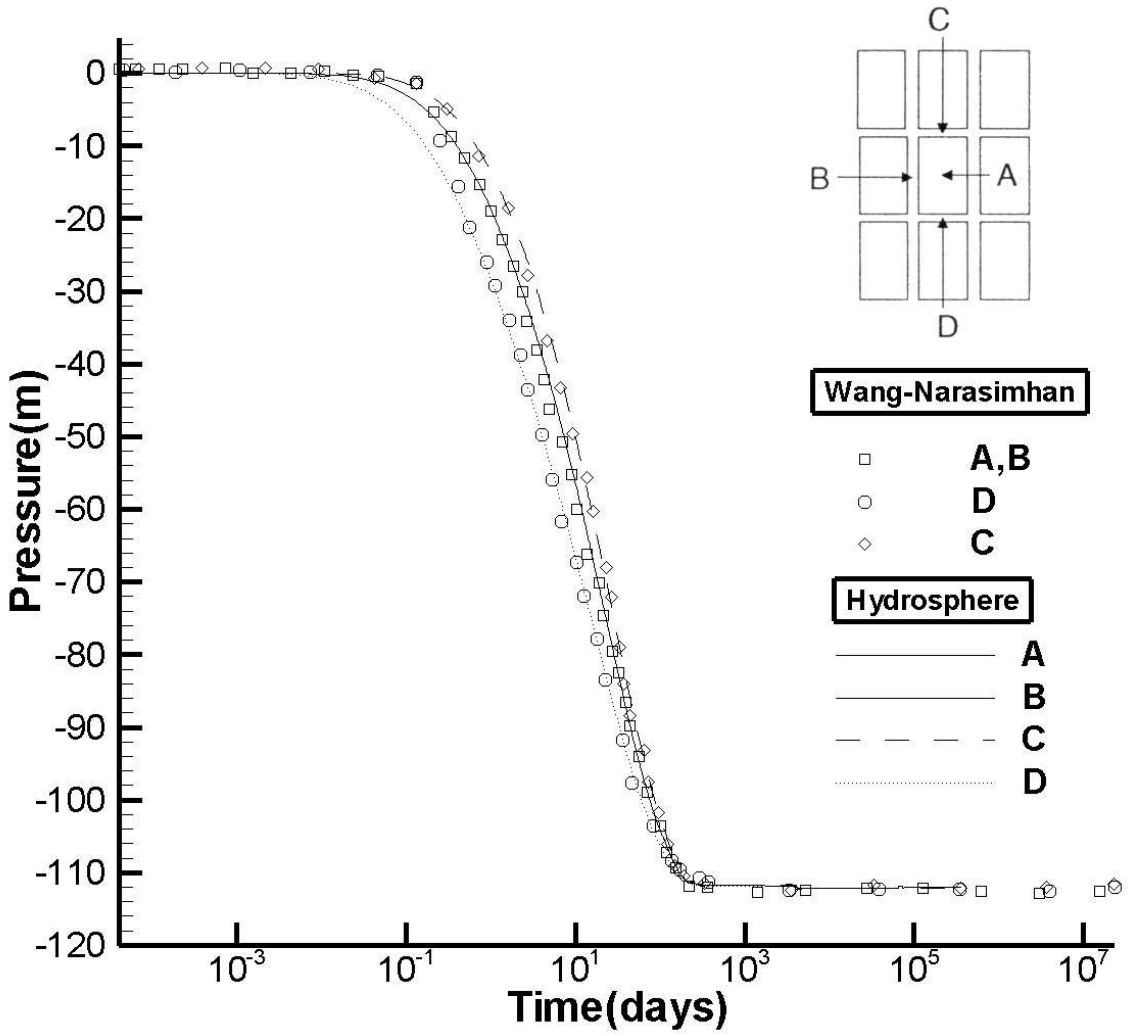


Figure 4.6: Pressure drop at selected points during the drainage of a fractured porous tuff.

The column was initially saturated, the fluid was static and its potential was everywhere zero. Drainage was performed by applying a suction equal to -112.0 m at the bottom of the column, all other boundaries being impermeable. This suction caused the fractures and the matrix to desaturate with time. Time stepping control was used to move the solution through time. A maximum change in pressure head of 1.0 m for each time step was used in conjunction with (3.86). The total CPU time for the flow simulation was 6 minutes on the IBM RS/6000 Model 590 and a total of 511 variable time steps were necessary to reach the final simulation time of 10^5 years. It should be noted that convergence of the Newton procedure occurred typically after the first iteration for most time steps.

A comparison of the results obtained with this model and those of *Wang and Narasimhan* [1985] is presented in Figure 4.6, which shows the change in fluid pressure with time for four different locations in the column. Location A represents the middle of the porous block,

location B is the middle of a vertical fracture bounding the matrix block and points C and D are in the middle of the horizontal fractures. It can be seen from Figure 4.6 that there is a very good agreement between the results obtained and those reported by *Wang and Narasimhan* [1985]. The pressure head is seen to decrease gradually at early times for all observation points. The decrease is more rapid at point D, which is closer to the drainage boundary. The pressure at points A and B is identical, revealing that the drainage process for this case is mainly influenced by the porous matrix when unsaturated conditions prevail.

The drainage simulation was repeated using both the finite element and the finite difference representations. Both representations produced identical results, although the finite difference representation required one quarter of the CPU time compared to the finite element scheme.

4.2 Overland Flow

4.2.1 General

In order to verify the numerical techniques and demonstrate applicability of the OLF modules within **HydroSphere** via simulation examples, three levels of code testing are presented. Level 1 verification is performed by comparison with available analytical solutions. Level 2 verification is performed on practical problems with complexities that preclude analytical solutions, by comparisons with published numerical solutions of simulators with some equivalent features. Finally, Level 3 verification focuses on field applications.

4.2.2 Level 1: 1-D Overland Flow Study of *Govindaraju et al.*, [1988a and 1988b]

The OLF package is verified against analytical and numerical solutions of kinematic wave, diffusive wave and dynamic wave equations, presented by *Govindaraju et al.* [1988a and 1988b]. The problem considered involves one-dimensional overland flow along a plane of one unit width (Figure 4.7). The authors presented numerical and analytical solutions for the different waves under a wide range of flow conditions ranging from highly subcritical flow (Froude number $F_o = u_o/\sqrt{gd_o} < 0.5$) to supercritical flow ($F_o = 1.5$) and at different kinematic wave numbers $K(= S_oL/d_oF_o^2)$ ranging from 3 to 50, where u_o is the uniform velocity and d_o is the uniform depth at the downstream end, S_o is the bed slope, and L is the slope length.

The following dimensions were used in the simulations:

Slope	$S_o = 0.01$
Slope length	$L = 100$ m

The rest of the parameters used for the simulations are shown in Table 4.6 which reproduces some of the of investigations of *Govindaraju et al.* [1988a and 1988b].

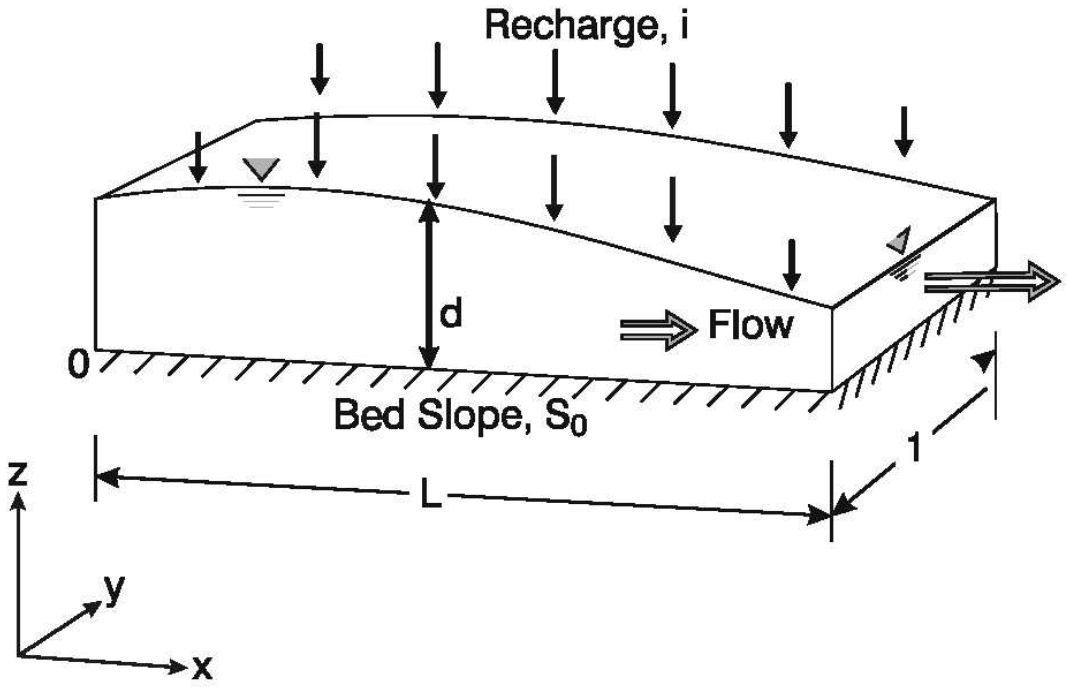


Figure 4.7: Schematic Description of Problem.

Figure	F_o	K	i m/s	u_o m/s	d_o m	n s/m$^{1/3}$	Q_o m^3/s	T s	Boundary Conditions
4.8	0.5	10	0.0040	0.9905	0.4000	0.0548	0.3962	242.3130	ZDG
4.9	0.4	20	0.0022	0.7004	0.3125	0.0658	0.2189	342.6823	ZDG
4.10	1.5	3	0.0027	1.8083	0.1481	0.0155	0.2679	132.7203	ZDG
4.11	1.5	3	0.0027	1.8083	0.1481	0.0155	0.2679	132.7203	ZDG
4.12	0.707	3	0.0121	1.8083	0.6669	0.0422	1.2059	132.7203	ZDG
4.13	0.707	3	0.0121	1.8083	0.6669	0.0422	1.2059	132.7203	ZDG
4.14	0.25	10	0.0158	0.9905	1.6000	0.1381	1.5847	242.3130	ZDG
4.15	0.5	3	0.0241	1.8083	1.3333	0.0670	2.4111	132.7203	ZDG
4.16	0.1	50	0.0089	0.4429	2.0000	0.3584	0.8859	541.8284	ZDG

Table 4.6: Parameters and results of simulation of 1-D flow

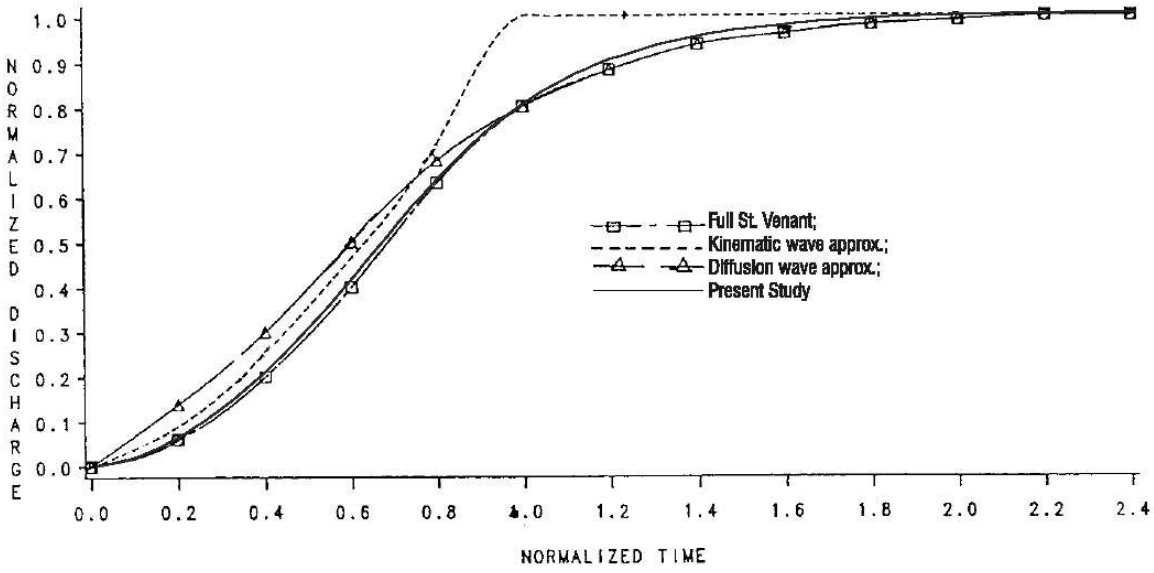


Figure 4.8: Comparison of Normalized Rising Hydrographs for Saint Venant Equations, the Diffusion Wave Approximation [*Govindaraju et al.*, 1988a] and MSVMS for $F_o = 0.5$ and $K = 10$.

The notation of Table 4.6 is as follows: F_o is the Froude number, K is the kinematic wave number ($K > 20$ indicates a kinematic wave), i_o is the uniform recharge to the system, d_o, Q_o and u_o are the uniform depth, discharge and velocity at the down stream end, n is Manning's roughness coefficient, and T is the total time of the simulation in seconds. The time, L/u_o, is used to normalize time in the figures while Q_o and d_o are used to normalize discharge and flow depth respectively. The boundary conditions at the downstream end are either zero-depth gradient (ZDG, equation 3.51) or critical depth (CD, equation 3.52) conditions.

Modeling Approach and Results

The slope was discretized into 100 columns and 1 row of nodes with dimensions 1m × 1m each. Only overland flow resulting from recharge was simulated and no interaction with ground water was considered. Adaptive time stepping was provided in the simulations with an initial time-step size of 1s, a maximum time-step size of 100s and a minimum time-step size of 0.01s. Iteration considerations include 100 and 20 maximum inner and outer iterations respectively, and a convergence tolerance of 10^{-4}. Figures 4.8 through 4.16 show the normalized discharge hydrographs at the downstream end of the slope as well as the steady-state water depth profiles for the various cases studied. Note that in Figures 4.14 through 4.16, the numerical solutions of Govindaraju correspond to the diffusion wave equation. The resulting hydrographs were normalized by dividing the discharge by the normal discharge Q_o and time by $t_o(t_o = L/u_o)$ for each simulation. Both the rising limb (Figures 4.8, 4.9, 4.10, and 4.12) and the falling limb (Figures 4.11 and 4.13) were compared to the results of *Govindaraju et al.* [1988a]. In addition *Govindaraju et al.* [1988b] presented

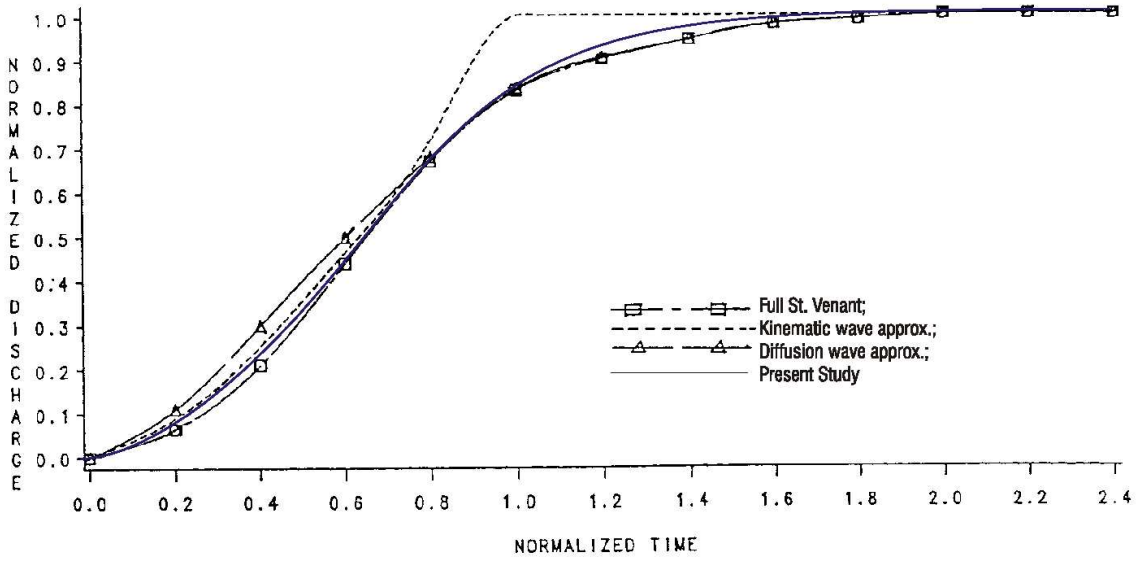


Figure 4.9: Comparison of Normalized Rising Hydrographs for the Saint Venant Equations, the Diffusion wave Approximation, the Kinematic Wave Approximation [*Govindaraju et al.*, 1988a], and MS-VMS for $F_o = 0.4$ and $K = 20$.

results for the steady-state depth profile (specifically Figures 4.14, 4.15 and 4.16). These depth profiles were compared to depth profiles from the **HydroSphere** simulations, at times long enough such that steady-state conditions are achieved.

Our code results are in good agreement with the diffusion wave solution of *Govindaraju et al.* [1988a and 1988b], with Picard and Newton iterations providing almost identical solutions. However, due to the limitations of the diffusive wave approximation, the solution deviates from the dynamic wave (i.e., the full Saint Venant solution) for small values of the kinematic wave number K. The *Govindaraju et al.* [1988a and 1988b] results also show that the kinematic wave approximation deviates from the diffusive and dynamic wave solutions for highly subcritical flow ($F_o < 0.5$). The steady-state depth profiles from **HydroSphere** are also in good agreement with the numerical results of *Govindaraju et al.* [1988b]. Similar to the *Govindaraju et al.* [1988b] conclusion, zero-depth gradient boundary conditions at the downstream end result in hydrographs similar to those of critical depth at the downstream end. However, the critical depth condition affects the shape of the water profile near the downstream end.

4.2.3 Level 2: Conjunctive Oveland-Subsurface Flow Study of *Smith and Woolhiser* [1971]

Smith and Woolhiser [1971] present an experimental study of combined overland and subsurface flow which may be used to validate the overland and subsurface flow routines and

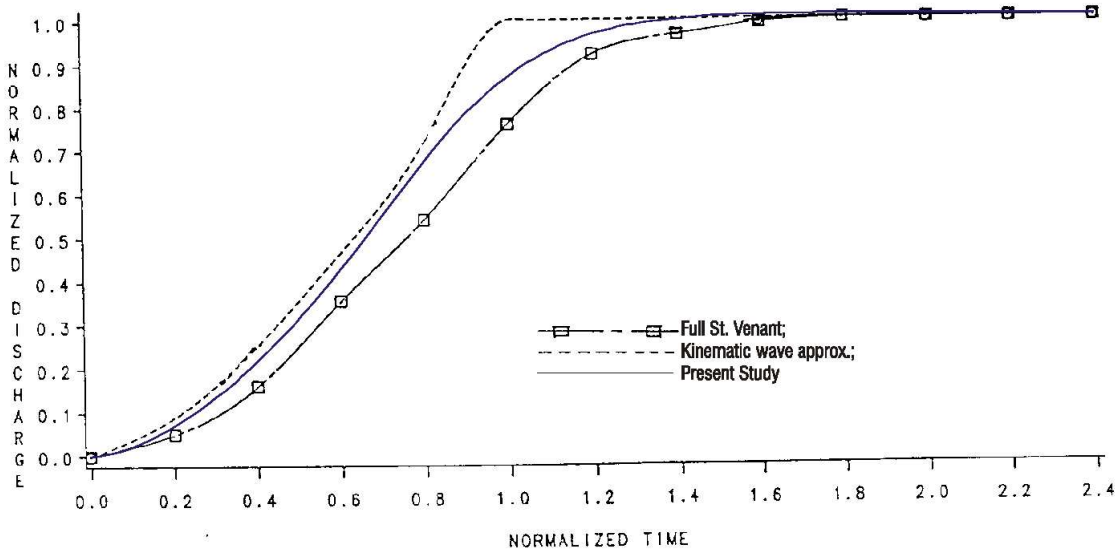


Figure 4.10: Comparison of Normalized Rising Hydrographs for the Saint Venant Equations, the Kinematic Wave Approximation [*Govindaraju et al.*, 1988a], and MS-VMS for $F_o = 1.5$ and $K = 3$.

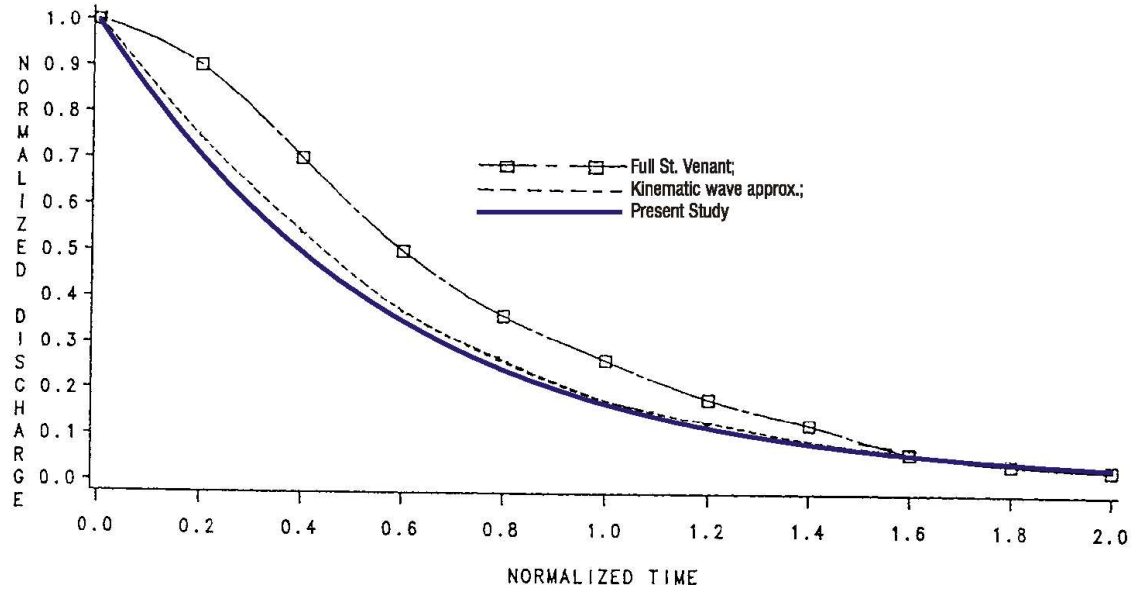


Figure 4.11: Comparison of Normalized Recession Hydrographs for Saint Venant Equations, the Kinematic Wave Approximation [*Govindaraju et al.*, 1988a], and MS-VMS for $F_o = 1.5$ and $K = 3$.

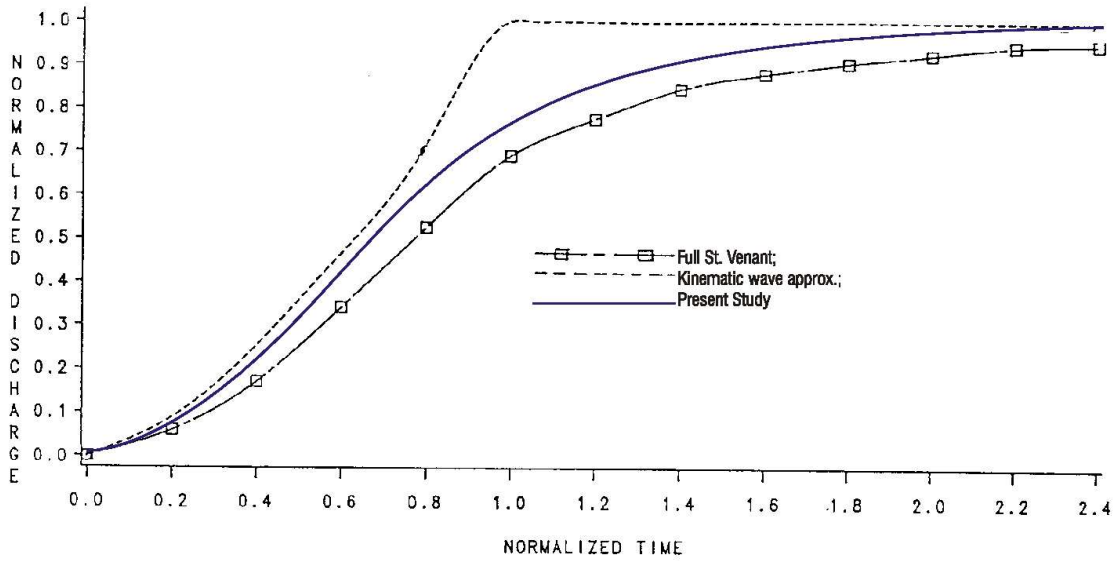


Figure 4.12: Comparison of Normalized Rising Hydrographs for Saint Venant Equations, the Kinematic Wave Approximation [*Govindaraju et al.*, 1988a], and MS-VMS for $F_o = 0.707$ and $K = 3$.

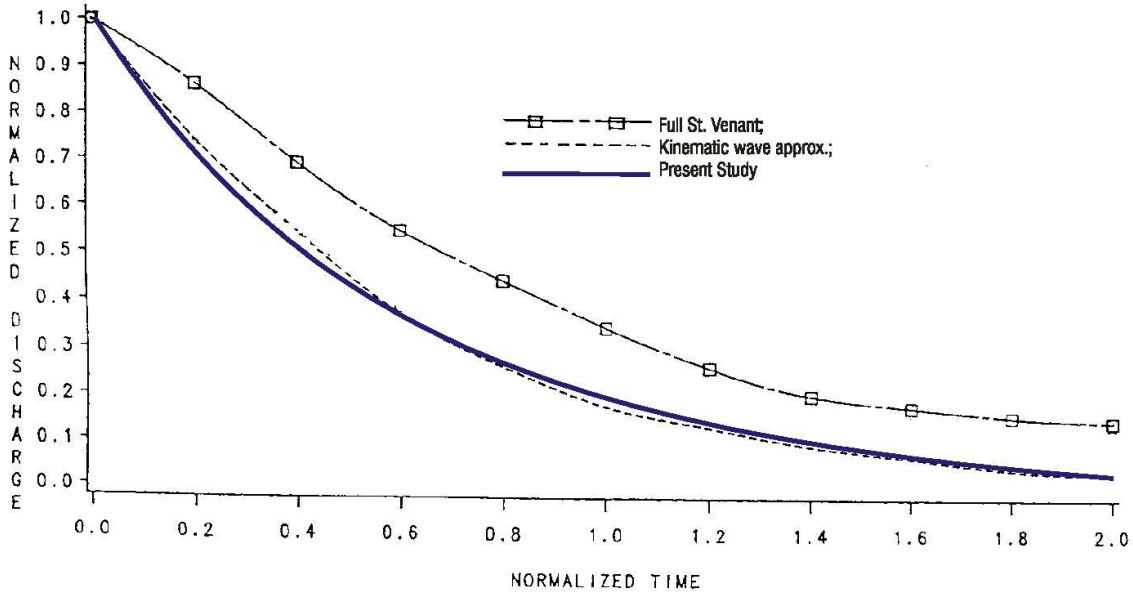


Figure 4.13: Comparison of Normalized Recession Hydrographs for Saint Venant Equations, the Kinematic Wave Approximation [*Govindaraju et al.*, 1988a], and MS-VMS for $F_o = 0.707$ and $K = 3$.

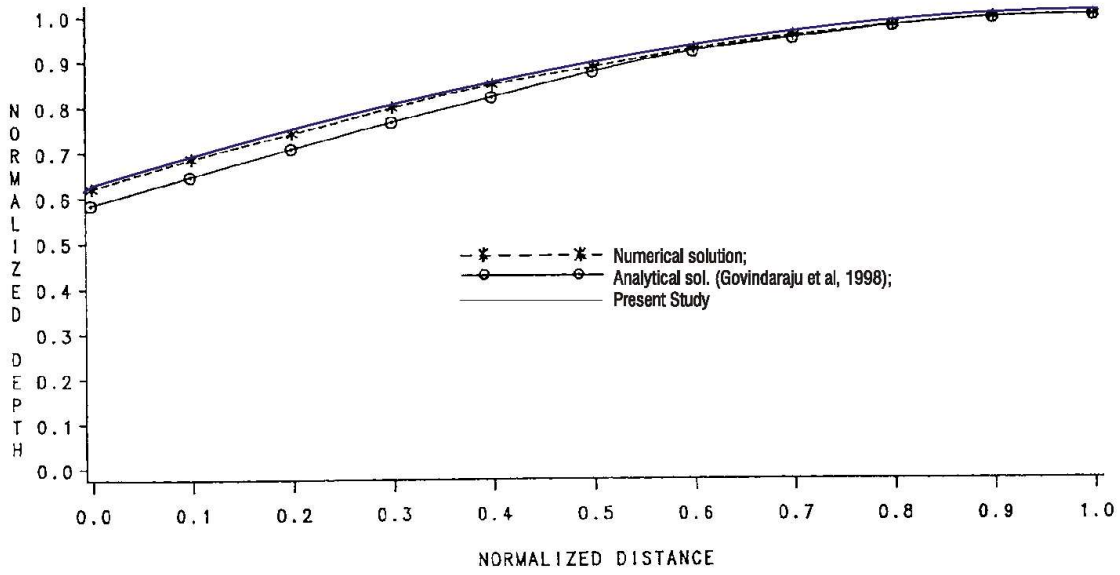


Figure 4.14: Numerical, Analytical [*Govindaraju et al.*, 1988b] and MS-VMS, Steady-state Profiles for Zero-depth Gradient Downstream Boundary Conditions for $F_o = 0.25$ and $K = 10$.

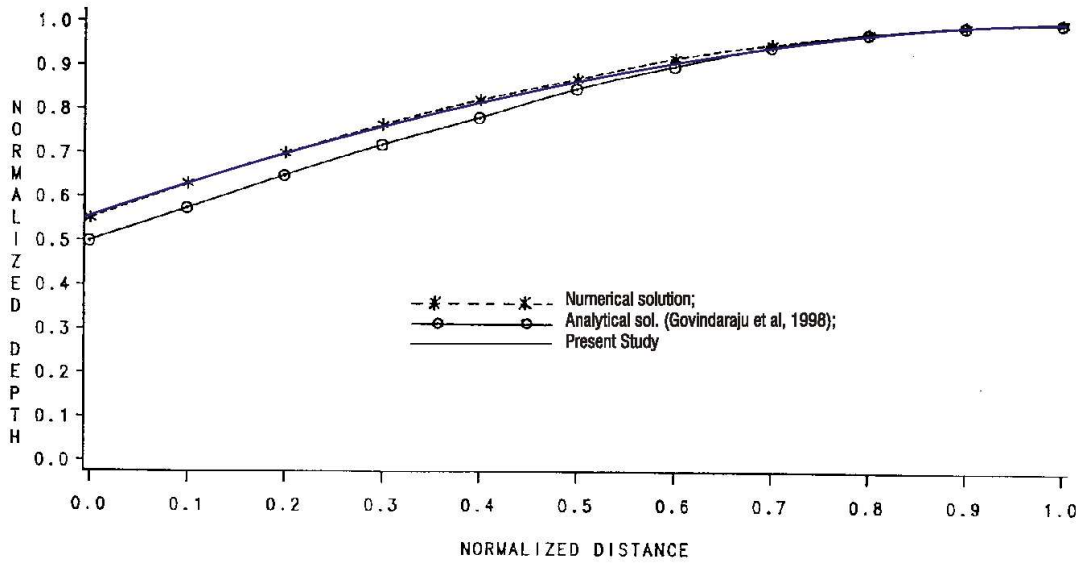


Figure 4.15: Numerical, Analytical [*Govindaraju et al.*, 1988b] and MS-VMS, Steady-state Profiles for Zero-depth Gradient Downstream Boundary Conditions for $F_o = 0.5$ and $K = 3$.

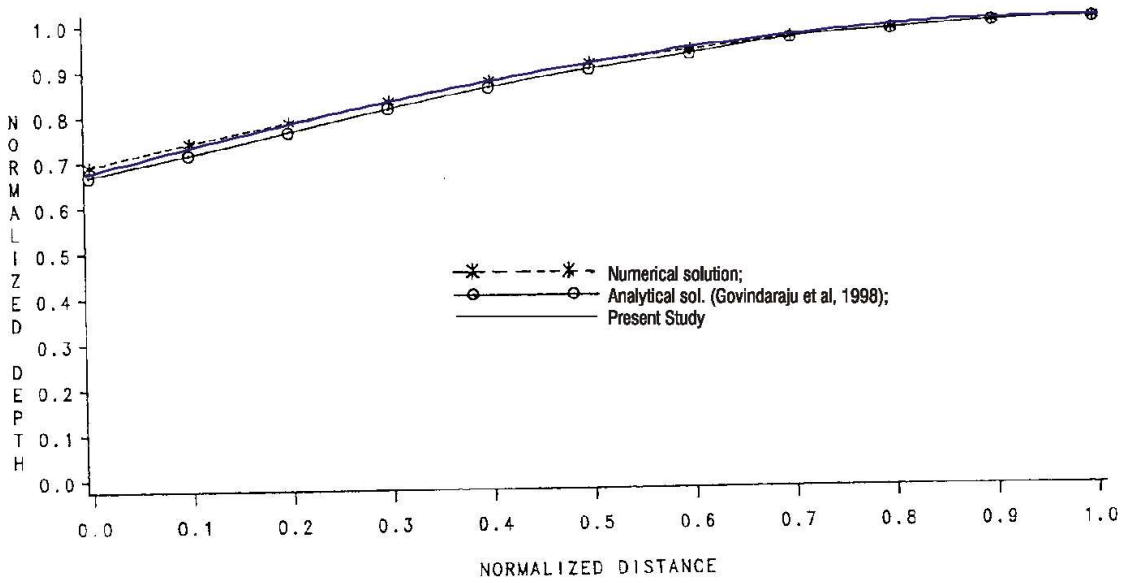


Figure 4.16: Numerical, Analytical [*Govindaraju et al.*, 1988b] and MS-VMS, Steady-state Profiles for Zero-depth Gradient Downstream Boundary Conditions for $F_o = 0.1$ and $K = 50$.

their interactions, in **HydroSphere**. The experiments consisted of providing rainfall of 25 cm/hr for 15 minutes over a soil flume 1,220 cm long, 5.1 cm wide and 122 cm deep. The flume was inclined along its length at a slope of 0.01, and the moisture movement into the soil, and downstream discharge were measured. The experimental setup is schematically depicted in Figure 4.17. Several conjunctive models of overland and subsurface flow have selected this experiment as a test case. *Smith and Woolhiser* [1971] present a model using the kinematic wave approximation for overland flow. *Akan and Yen* [1981] and *Singh and Bhallamudi* [1998] solve the full dynamic wave equation for overland flow. *Govindaraju and Kavvas* [1991] solve the diffuse wave equation for overland flow for this case. All above models solve the Richards equation for flow in the subsurface, and use flux coupling between the overland and subsurface equations.

The case considered for simulation here, involves a dry soil. The soil (a Poudre fine sand) was placed in the flume in three layers of thickness 7.65 cm, 22.95 cm, and 76.1 cm from top to bottom (denoted as layers 1, 2, and 3, respectively). Hydraulic property curves provided by *Smith and Woolhiser* [1971] for the 3 soil layers were fit to the van Genuchten functions using a regression code. Figures 4.18 through 4.23 show the moisture retention and hydraulic conductivity vs. suction curves for the three soils as provided by *Smith and Woolhiser* [1971] and the fitted van Genuchten functions. (Note that k_r = relative permeability, Ψ = suction head, Θ = moisture content, Θ_r = residual moisture content, Θ_s = saturated moisture content, α and β are Van Genuchten parameters, k_s = saturated conductivity, and S_r = residual saturation). *Singh and Bhallamudi* [1998] provide details of the Brooks-Corey relations fitted to the data of *Smith and Woolhiser* [1971]. Figures 4.18 through 4.23 show that the van Genuchten parameters used for the simulation provide a

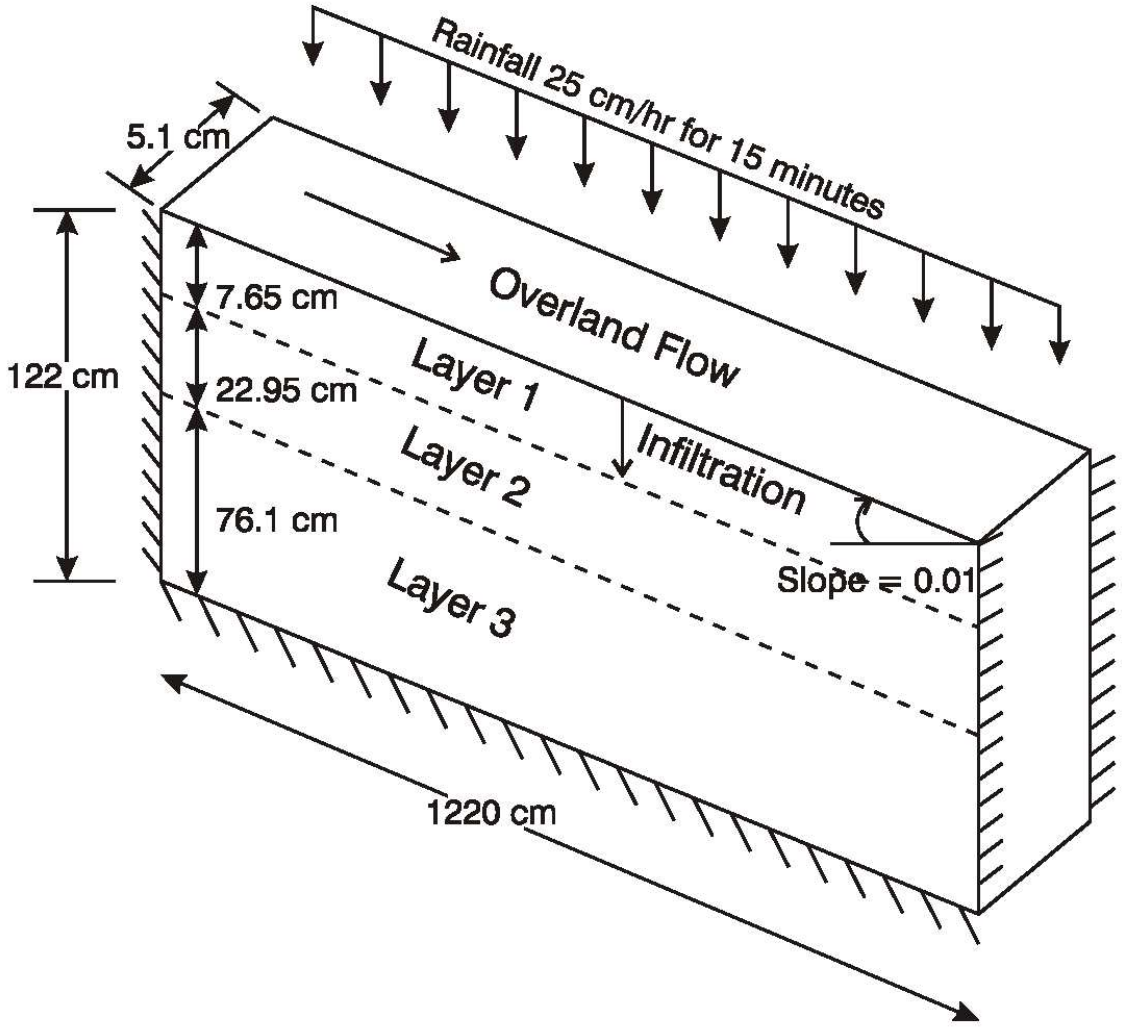


Figure 4.17: Experimental Setup of the *Smith and Woolhiser* [1971] Study.

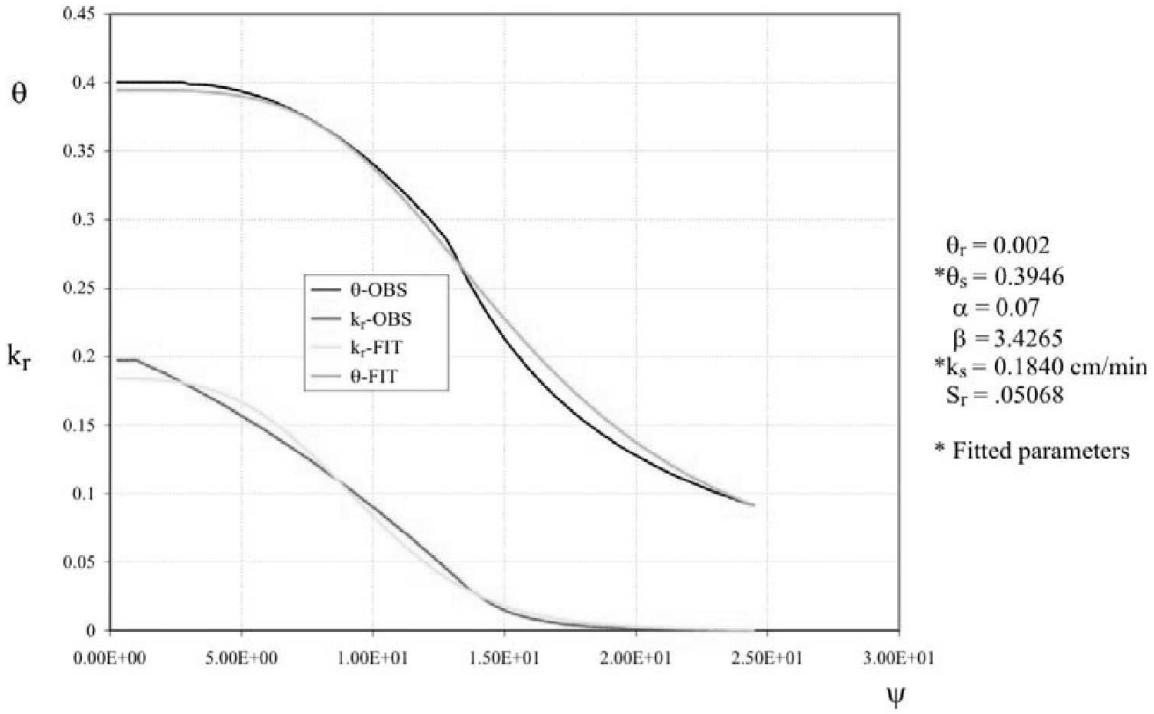


Figure 4.18: Moisture Retention Curve for Soil Layer 1.

good fit to the experimental data in the range of suction valid for this study. Table 4.7 provides the fitted van Genuchten parameter values of the simulation for the 3 soil layers. *Akan and Yen* [1981] and *Singh and Bhallamudi* [1998] provide the Darcy- Weisbach friction factor used for the simulation as $f = C_L/R_e$ where the coefficient C_L is 92 for this laminar flow problem, and R_e is the Reynolds number defined as $R_e = q/\mu$ where q is the discharge per unit width and μ is the kinematic viscosity of the liquid. The liquid used in the experiments was a light oil with a kinematic viscosity of 1.94×10^{-6} m^2/s. The average discharge for the experiments was approximately 1×10^{-5} m^3/s for a flume width of 0.051 m, giving $q = 10^{-5}/0.051 = 1.96 \times 10^{-4}$ m^2/s and thus $R_e = 101.07$, for an average number to determine the friction factor f as $92/101.07 = 0.91$. The Darcy-Weisbach equation, Chezy's equation and Manning's equation are related as $C_c = d^{1/6}/n = \sqrt{8g/f}$ where d is the depth and g is gravity, thus giving the Chezy constant for this problem as $C_c = \sqrt{8g/f} = 5569.1$ cm$^{1/2}$/min and the Manning constant as $n = d^{1/6}/C = 0.000180d^{1/6}$ min/cm$^{1/3}$. The depth of flow for this problem is less than 1 cm - assuming it to be 0.15 cm gives $d^{1/6}$=0.6873 (the one-sixth power brings it all closer to unity). Thus, Manning's n for this problem is 0.0001228 min/cm$^{1/3}$.

Modeling Approach and Results

The two-dimensional domain was discretized into 100 columns, 1 row and 40 layers. The length of each cell was 12.2 cm with a width of 5.1 cm and a thickness of 1.53 cm. The top 5 model layers represent soil layer 1, the next 15 model layers represent soil layer 2 and the remaining 20 model layers represent soil layer 3. The entire rectangular domain is

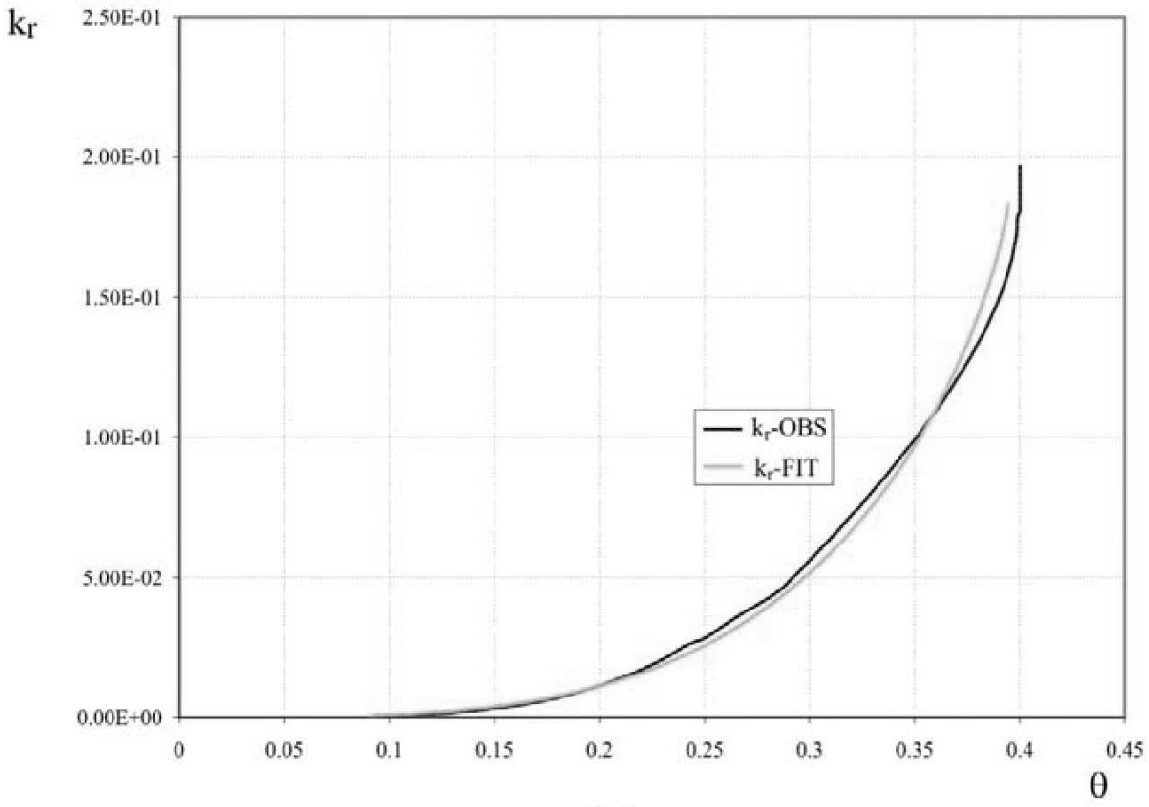


Figure 4.19: Moisture Retention Curve for Soil Layer 1.

Parameter	Value	
Soil layer 1 †		
porosity, n	0.395	
saturated conductivity, K_s	0.184	cm/min
van Genuchten parameter, α	0.07	cm^{-1}
van Genuchten parameter, β	3.9265	
residual saturation, S_r	0.0507	
Soil layer 2		
porosity, n	0.439	
saturated conductivity, K_s	0.145	cm/min
van Genuchten parameter, α	0.056	cm^{-1}
van Genuchten parameter, β	4.1371	
residual saturation, S_r	0.0570	
Soil layer 3		
porosity, n	0.476	
saturated conductivity, K_s	0.130	cm/min
van Genuchten parameter, α	0.0443	cm^{-1}
van Genuchten parameter, β	4.3565	
residual saturation, S_r	0.0525	
Darcy-Weisbach friction, f_r	0.91	
Equivalent Chezy coefficient, C_c	5569.1	$cm^{1/2}/min$
Equivalent Mannings coefficient, n	0.000122	$min/cm^{1/3}$
Rainfall intensity, i	0.417	cm/min
Channel slope	0.01	
Channel length	1220	cm
Channel width	5.1	cm
gravitaional acceleration	3.532×10^6	cm/min^2

† The soil parameters provide the fit to experimental data of *Smith and Woolhiser* [1971], as shown in Figures 4.18 through 4.23 for the three soil layers.

Table 4.7: Parameter values for simulation of the *Smith and Woolhiser* [1971] experiment

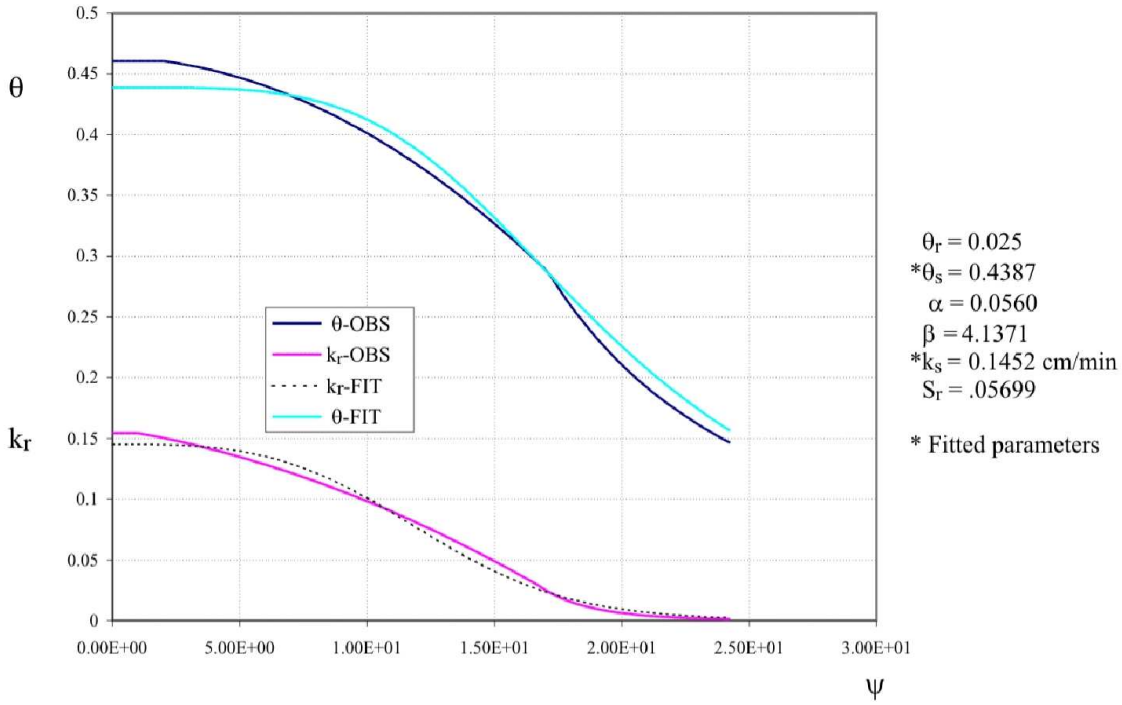


Figure 4.20: Moisture Retention Curve for Soil Layer 2.

tilted along its length to provide a slope of 0.01 to the model domain. The leakance factor between an overland flow node and a subsurface flow node for the skin leakance coupling case is determined as the saturated conductivity of the soil (K_s=0.184 cm/min) divided by the distance between the first two subsurface nodes (L=1.53/2 cm) and is calculated as K_G=0.06014 min^{-1}. Infiltration of 0.416667 cm/min is supplied for a duration of 15 minutesfollowed by 5 minutes with zero recharge. Adaptive time-stepping uses an initial time-step size of 0.1 min, a maximum time-step size of 1 min and a minimum time-step size of 10^{-3} min. Inner and outer iteration limits were set to 100 and 30 respectively, with a convergence tolerance of HCLOSE=10^{-4} cm. A zero-depth gradient boundary condition is applied at the downstream end of the overland flow nodes, while the lateral and bottom boundaries of the subsurface are provided no flow conditions.

The soil is initially dry at an approximate saturation of 0.2 as shown by *Smith and Woolhiser* [1971], which is converted to the appropriate head value using the van Genuchten relation for the appropriate soil layer. The starting heads for the overland flow nodes are set at the soil surface to represent dry starting conditions at the surface for the skin leakance case. Simulations were performed using both Picard and Newton Raphson schemes, as well as using Darcy-Weisbach (friction factor calculated by the code) and Mannings equations to represent surface flow conditions.

Figure 4.24 shows the saturation profiles within the soil at a distance of 550 cm from the upstream end at different times. Infiltration causes the saturation front to advance within the soil, with a total of 3.297×10^{-2} m^3 of water having infiltrated within the 15 minute

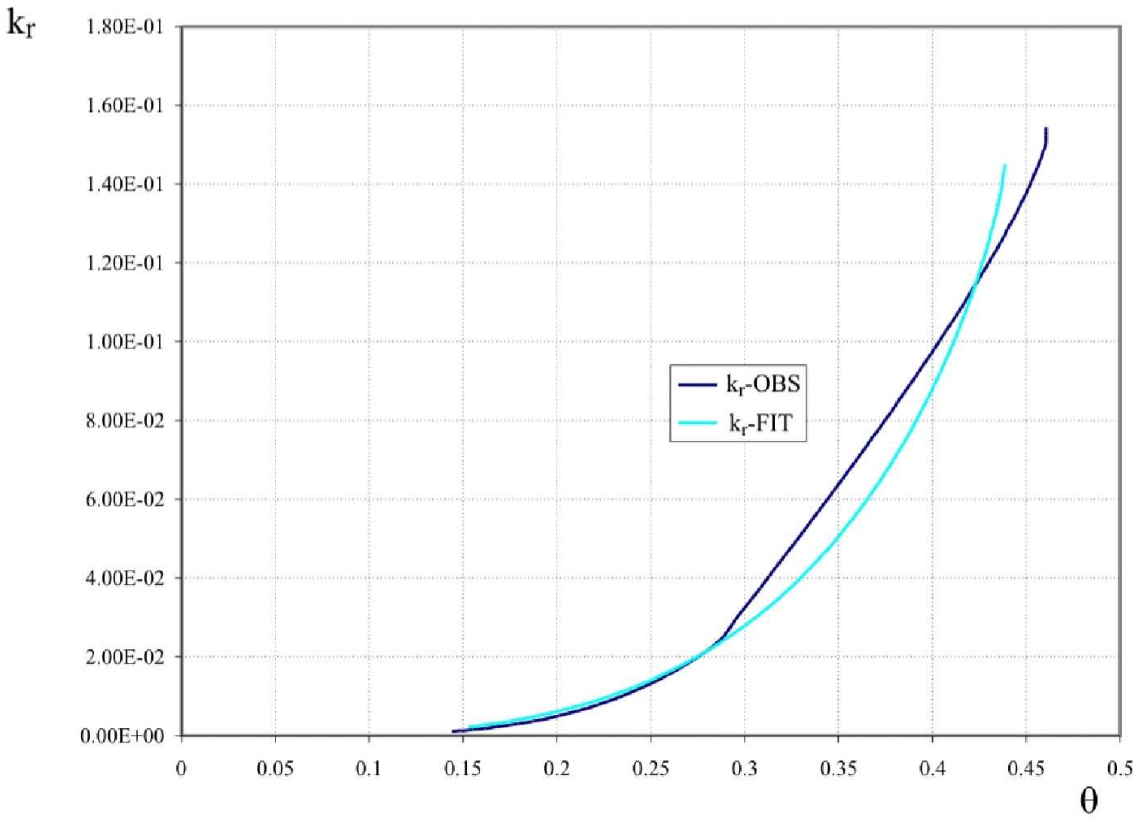


Figure 4.21: Moisture Retention Curve for Soil Layer 2.

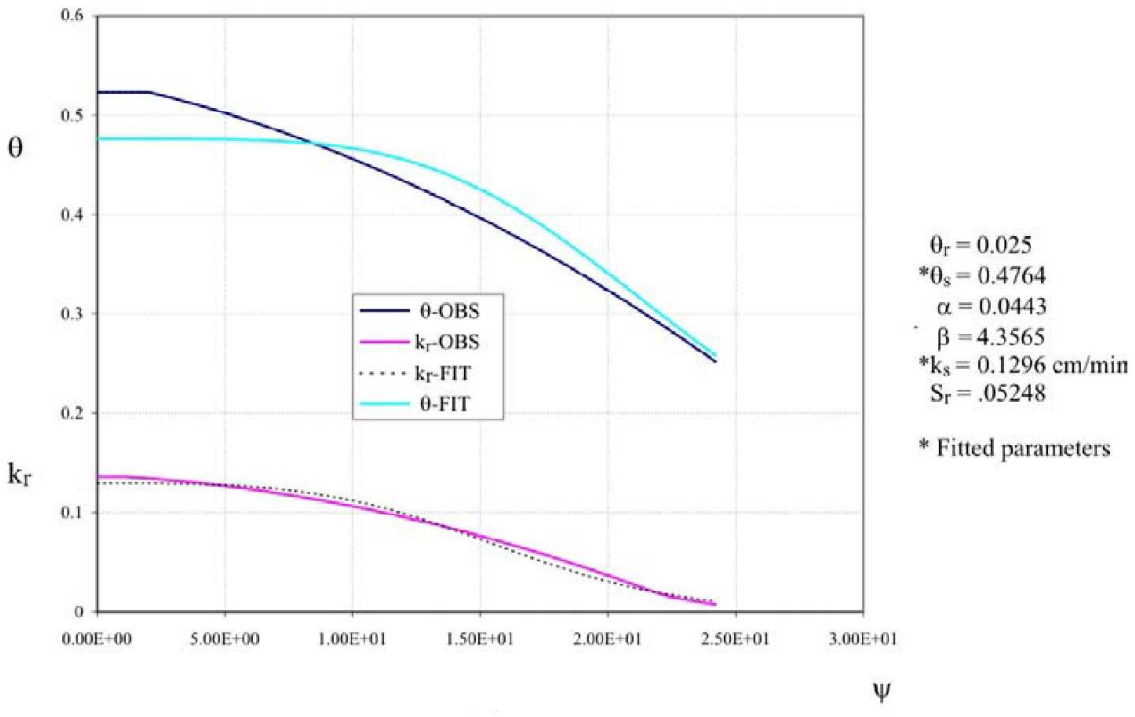


Figure 4.22: Moisture Retention Curve for Soil Layer 3.

recharge period, and 3.422×10^{-2} m^3 of water having infiltrated within 20 minutes—the period of simulation. The infiltration front predicted by **HydroSphere** compares fairly well with other simulation attempts [*Singh and Bhallamudi* 1998; *Smith and Woolhiser*, 1971] and with available experimental data. Figure 4.25 shows the outflow hydrograph at the downstream end of the overland flow nodes. This compares well with experimental data, and with other simulation attempts using the kinematic wave equation [*Smith and Woolhiser*, 1971] or the dynamic wave equation [*Singh and Bhallamudi*, 1998]. The simulation is sensitive to initial soil moisture conditions which may be calibrated for a better match. Figure 4.26 shows the surface water depth profiles at 8, 15 and 16 minutes of simulation respectively. The depth is noted to increase in time up till the end of the recharge period, and rapidly dissipates thereafter. Total outflow from the downstream overland flow node is 4.31×10^{-3} m^3 after 15 minutes of the recharge period, and 4.667×10^{-3} m^3 within the 20 minutes of simulation. Mass balance errors were less than a fraction of a percent throughout the simulation.

4.2.4 Level 2: 2-D Overland Flow Study of *diGiammarco et al.*, [1996]

Two-dimensional areal overland flow is verified for the overland flow package using the rainfall-runoff example of *diGiammarco et al.* [1996]. *VanderKwaak* [1999] presents details of the simulation, with results from various surface water flow codes benchmarked against this problem. The problem involves two-dimensional overland flow from a tilted V-

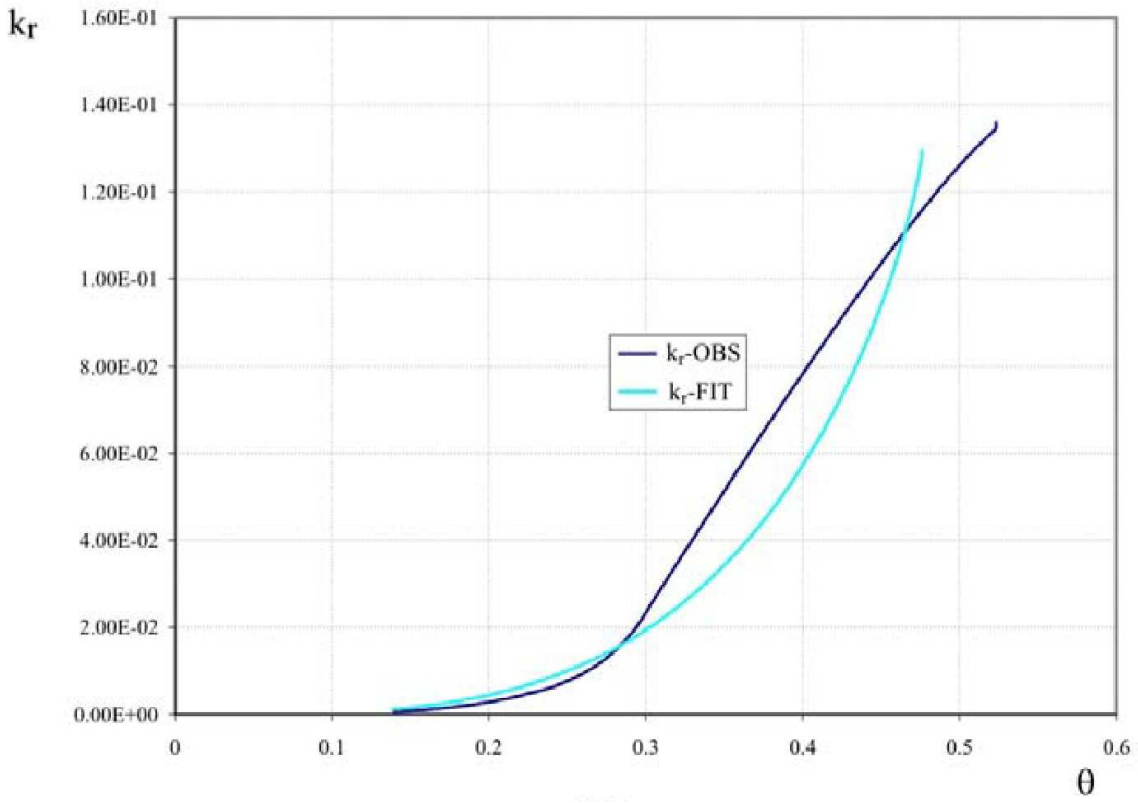


Figure 4.23: Moisture Retention Curve for Soil Layer 3.

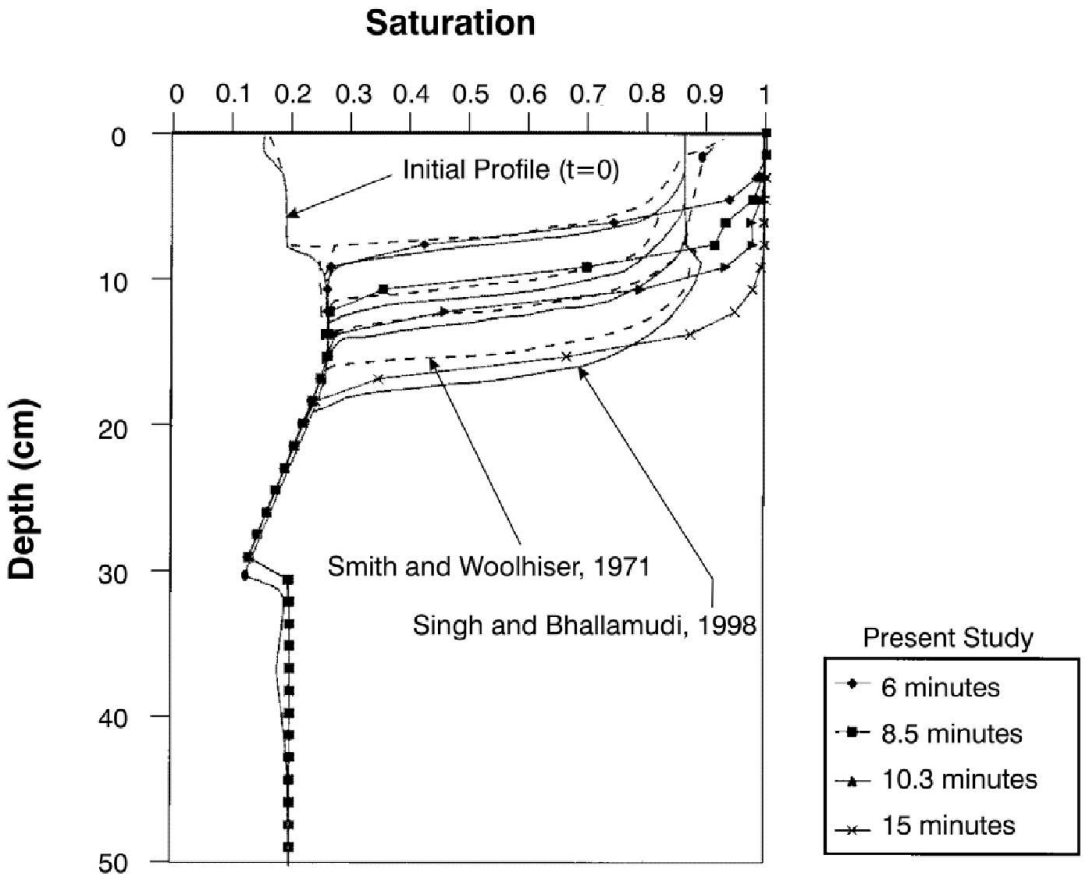


Figure 4.24: Soil Saturation Profile at 550 cm from Upstream end for Simulation of the *Smith and Woolhiser* [1971] Study.

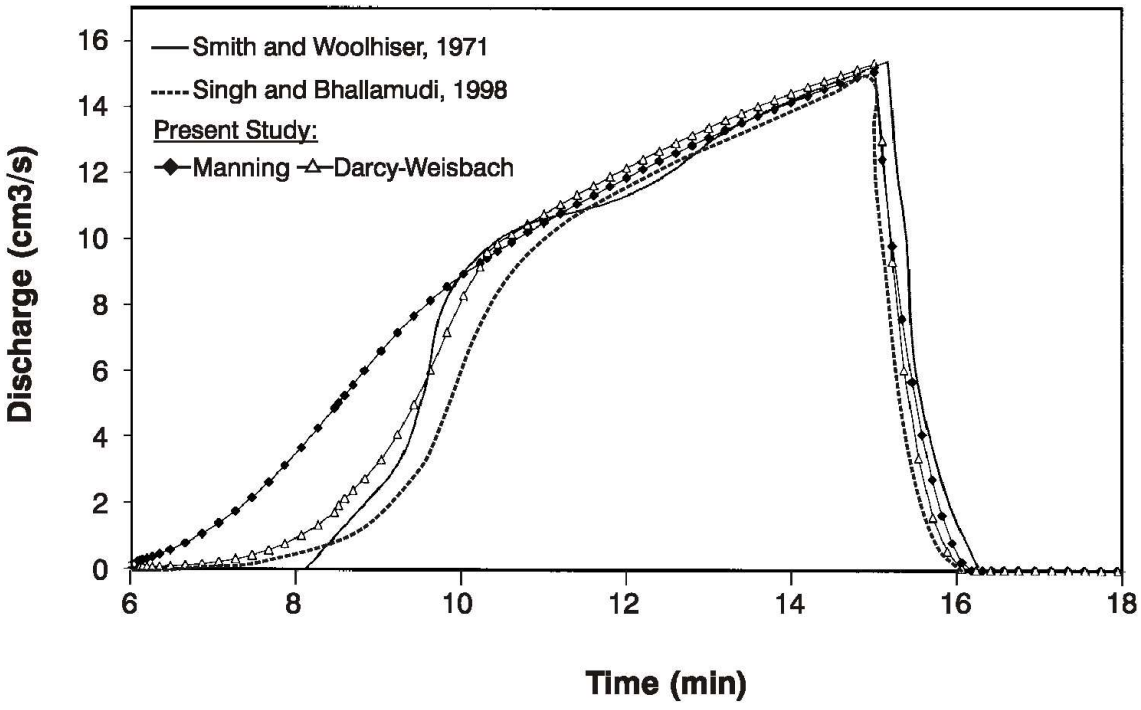


Figure 4.25: Outflow Hydrograph for Simulation of the *Smith and Woolhiser* [1971] Study.

catchment (Figure 4.27) generated by a 90 minute duration, 3×10^{-6} m/s intensity rainfall event. Only one half of the domain need be simulated due to symmetry, with simulated outflow discharge doubled, to produce equivalent results. The simulation domain therefore consists of a 1,000m by 800m slope connected to 1,000m length of channel 10m wide. Surface slopes are 0.05 and 0.02, perpendicular to, and parallel to the channel respectively. Manning's roughness coefficients of 0.015 and 0.15 are applied to the slopes and channel, respectively. A critical depth boundary condition is applied at the downstream end of the channel.

Modeling Approach and Results

The domain was discretized into 17 rows and 20 columns, with the last row representing the channel being 10m wide. The remaining grid dimensions are 50m $\times$ 50m. Only overland flow was simulated, with rainfall at the rate of 3×10^{-6} m/s for 90 minutes, followed by zero recharge for the second stress period of 90 minutes. Adaptive time stepping with an initial time-step size of 5s, a maximum time-step size of 100s and a time-step incrementing factor of 2.0 is used for the simulation. Iteration parameters include 100 and 15 maximum inner and outer iterations respectively, and a head convergence tolerance of 10^{-3}m. Figure 4.28 compares predicted discharge from **HydroSphere** with predictions from several other codes. Excellent agreement is noted between all results. Figure 4.28 shows the channel stage at the outlet. The stage at the outlet point is noted to increase and dissipate in accordance with the computed discharge fluxes. Most time-steps converged within two

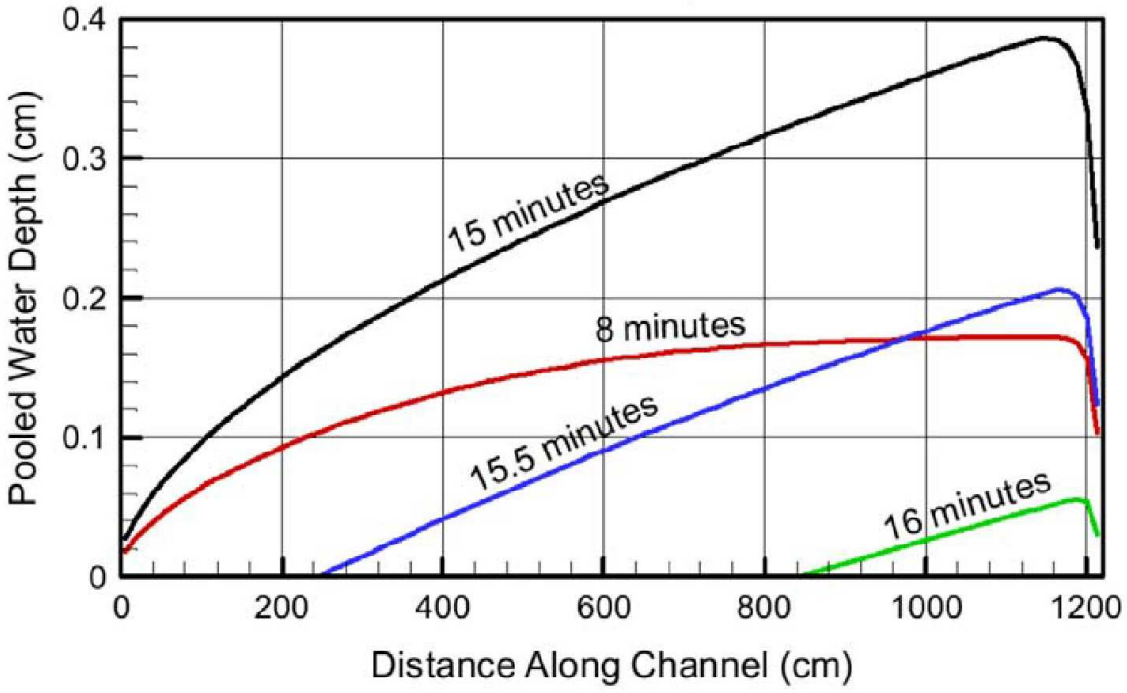


Figure 4.26: Surface Water Depth Profiles at Different Times for the Simulation of the *Smith and Woolhiser* [1971] Study.

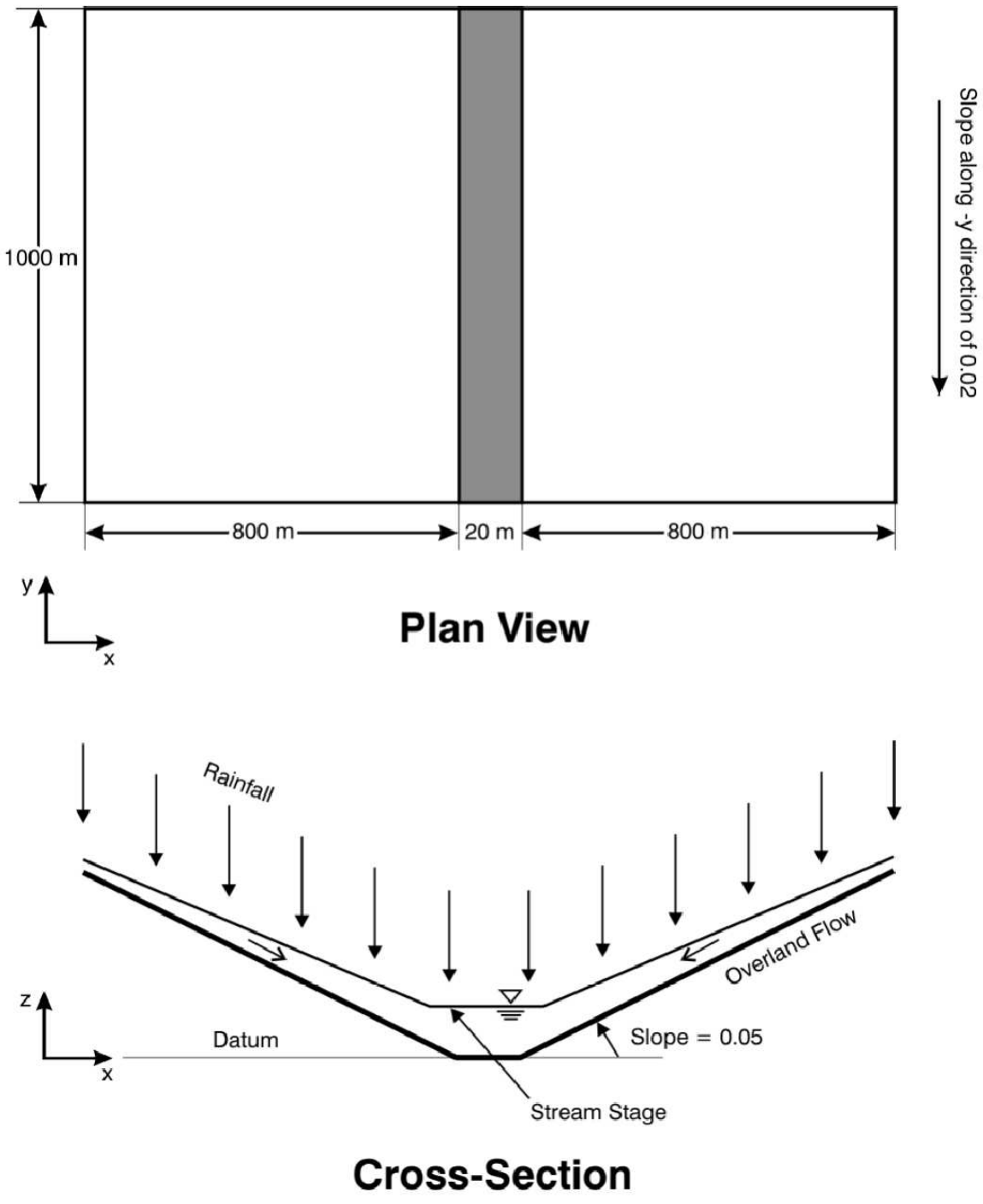


Figure 4.27: Schematic Description of 2-D Surface Water Flow Study of *diGiammarco et al.* [1996].

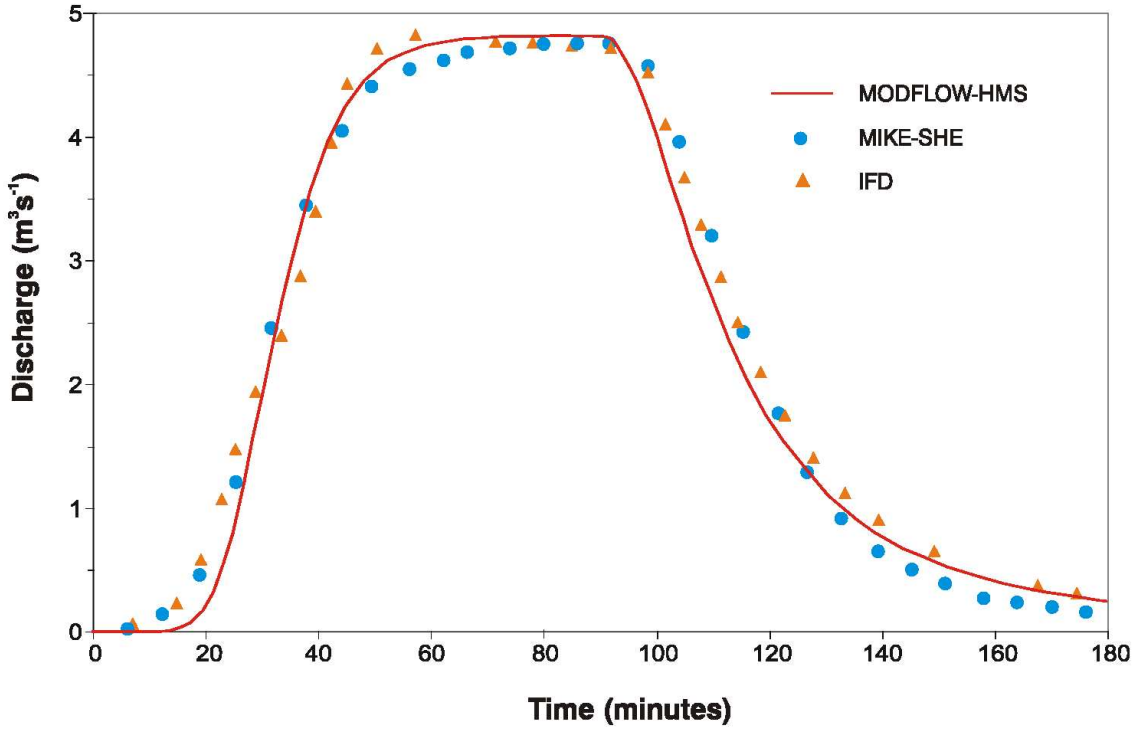


Figure 4.28: Outflow Hydrograph for Simulation of 2-D Surface Water Flow Study of *di-Giammarco et al.* [1996].

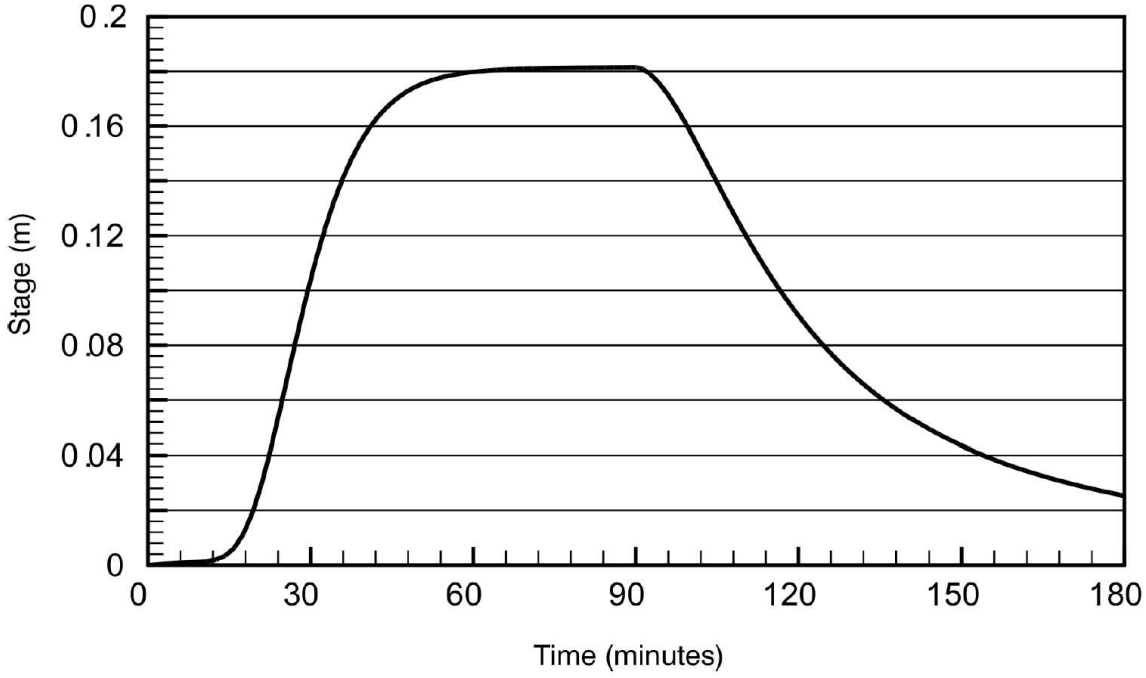


Figure 4.29: Channel Stage at Outlet for Simulation of 2-D Surface Water Flow Study of *diGiammarco et al.* [1996].

Parameter	Value	
porosity, Θ	0.37	
hydraulic conductivity, K	1×1^{-5}	m/s
storage coefficient, S_s,	1.2×1^{-7}	
van Genuchten parameter, α	1.9	m^{-1}
van Genuchten parameter, β	6	
residual saturation, S_r	0.18	
Brooks-Corey coefficient, n	3.4	
Mannings coefficient for plot	0.3	$s/m^{1/3}$
Mannings coefficient for channel	0.01	$s/m^{1/3}$
Initial water table elevation	2.78	m

Table 4.8: Parameter values for simulation of the 3-D field scale study of *Abdul* [1985]

iterations, with negligible mass balance errors throughout the simulation.

4.2.5 Level 3: 3-D Field Scale Study of *Abdul* [1985]

A field scale simulation is performed with **HydroSphere** to verify its capabilities for fully three-dimensional overland/subsurface flow modeling. Experiments conducted at Canadian Forces Base Borden, in Ontario, Canada, by *Abdul* [1985] are selected for this simulation. *VanderKwaak* [1999] presents details of the site, its characteristics, the experimental setup and the results. Figure 4.30 shows the experimental plot, approximately 80 m × 16 m areally and up to 4 m deep. A man-made stream channel lies approximately 1.2 m below the surrounding grassy land. The channel is initially dry prior to the application of the artificial rainfall via irrigation sprinklers. The initial water table lies around 22 cm below the streambed with the artificial recharge applied at a rate of 2 cm/hr for 50 minutes. Infiltration in upland regions, discharge in lower regions and runoff, all govern the behavior of the system. Soil properties and roughness coefficients are provided in Table 4.8. A critical depth boundary condition is applied at the downstream end of the channel, and the simulation is performed for 100 minutes, with no recharge occurring for the second 50 minute stress period.

Modeling Approach and Results

The domain was discretized areally into 70 rows and 23 columns of size $x = 0.7186$ m and $y = 1.143$ m. Vertically, the grid is distorted to conform with the topography of the upland regions and the channel as shown in Figure 4.31. Nine layers of nodes were used with a fine discretization of 0.02 m near the surface which was enlarged to 1 m near the bottom. Recharge is provided at the rate of 5.56×10^{-6} m/s for a first stress period of 50 minutes, with zero recharge for the second 50 minute stress period. Adaptive time-stepping is used with an initial time-step size of 5s, a minimum time-step size of 0.5s, a maximum time-step size of 100s, and time-step incrementing and decrementing factors of 1.5 and 4, respectively. Iteration parameters include 100 and 18 inner and outer iterations respectively, and a convergence tolerance of 1×10^{-4} m. Newton-Raphson linearization is

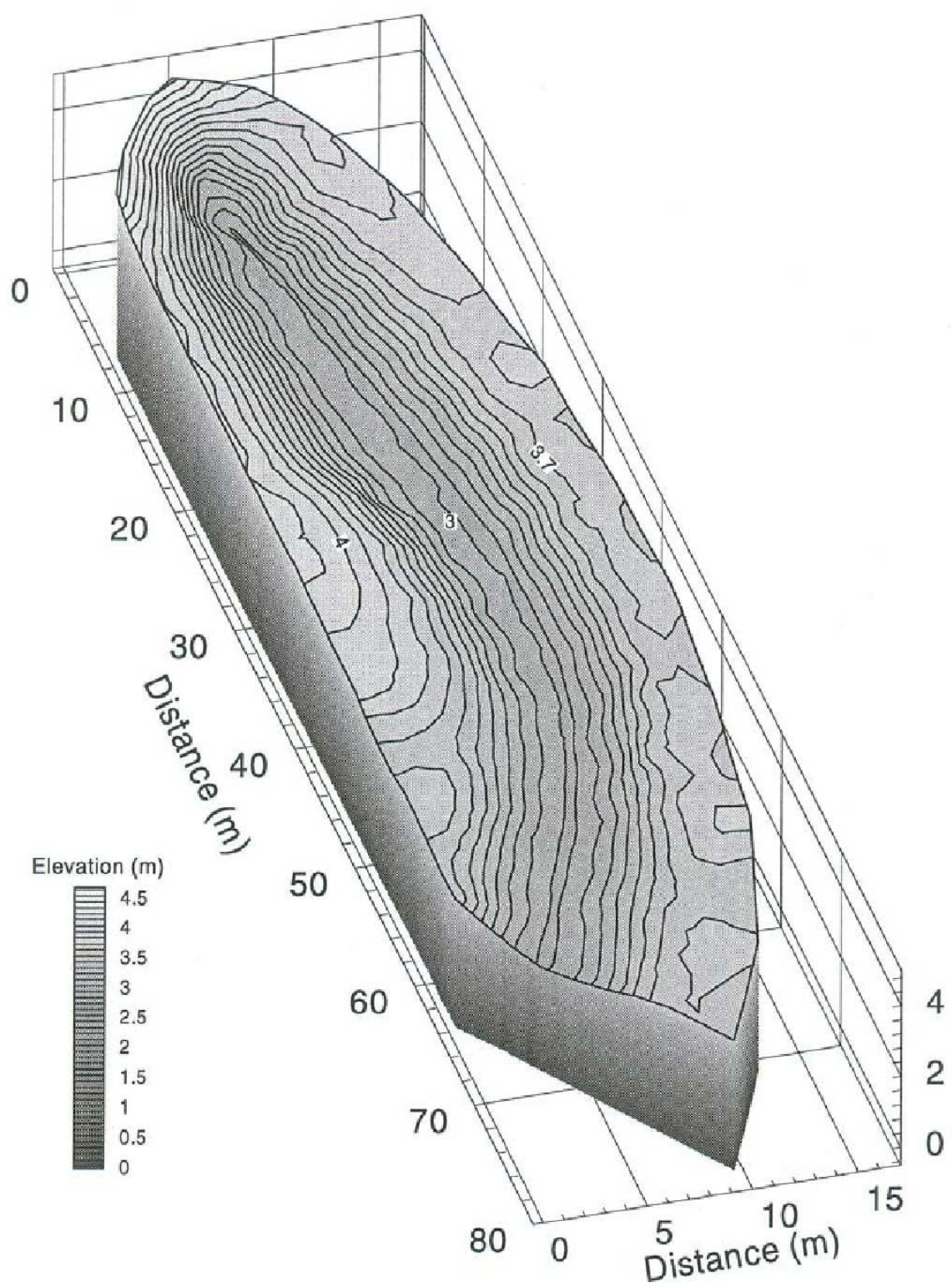


Figure 4.30: Site Description for Rainfall-Runoff Field Experiment of *Abdul* [1985] [from *VanderKwaak*, 1999].

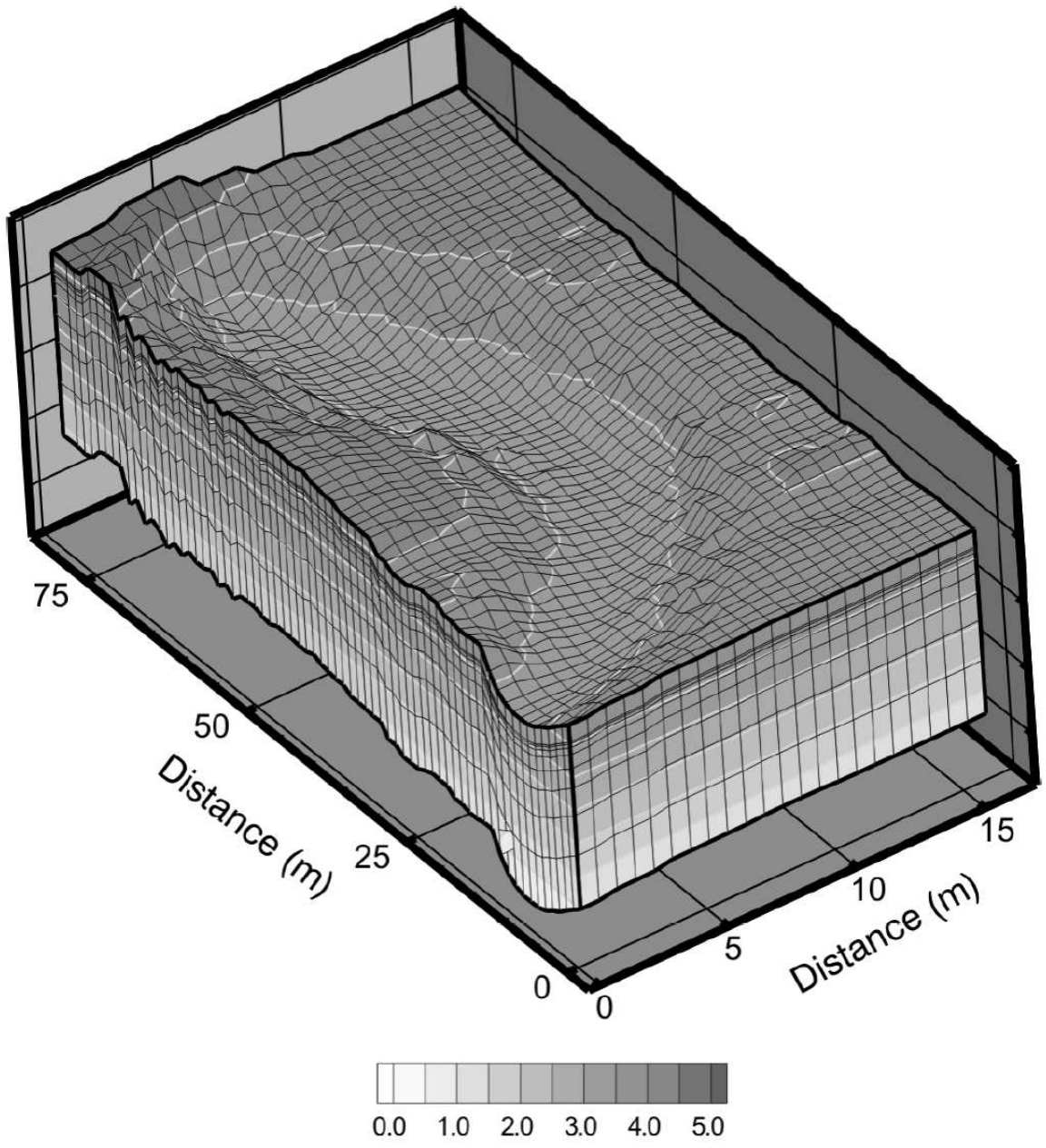


Figure 4.31: Three-dimensional View of Topography and Finite element Grid, for Simulation of the *Abdul* [1985] Rainfall-Runoff Field Experiment.

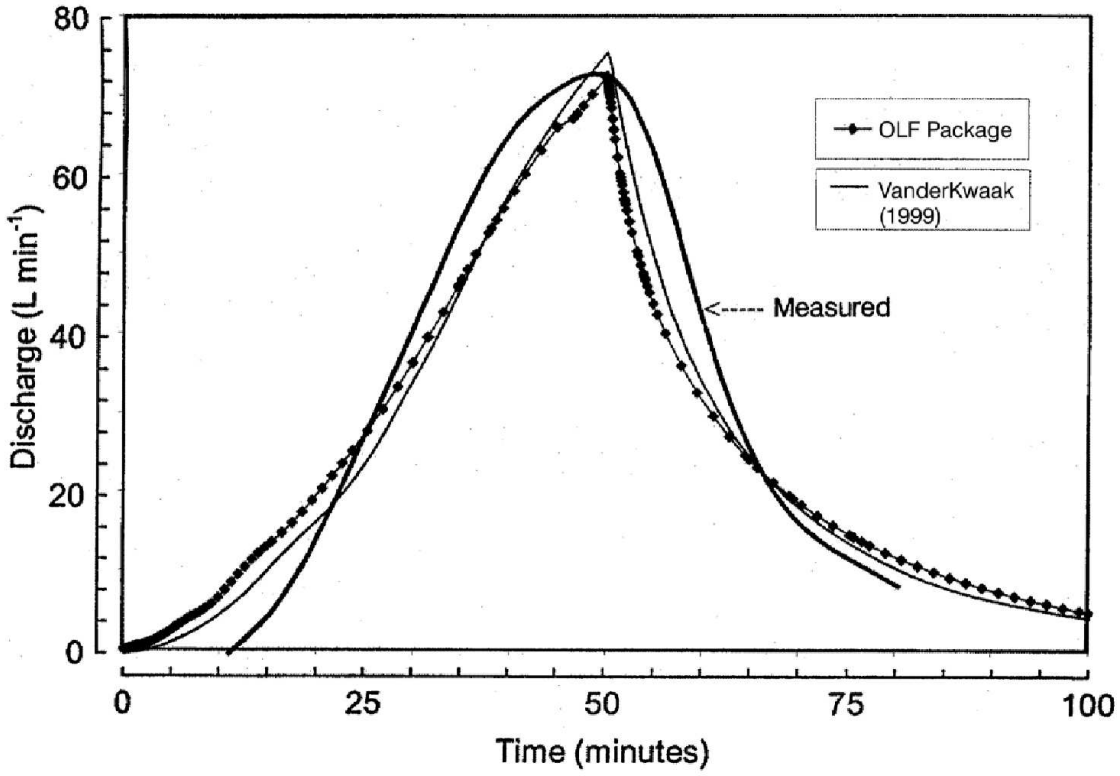


Figure 4.32: Outflow Hydrograph for Simulation of the *Abdul* [1985] study.

used to solve this system. Figure 4.32 **rgm Ed questions the results of vanderkwaak shown here** shows the outflow hydrograph to be close to the experimental values, and to the simulation of *VanderKwaak* [1999]. The spatial distribution of hydraulic heads within the domain at t=50 mins is shown in Figure 4.33. These compare well with results given by *VanderKwaak* [1999]. As noted by *VanderKwaak* [1999], this system's outflow hydrograph is extremely sensitive to channel elevations, initial water-table levels, and side-slopes of the plot. For example, inclusion of rill storage of 0.004 m decreases the peak discharge to about 63 L/min with the rest of the parameters remaining the same. If the initial water table was also raised to 2.82 m from 2.78 m, the outflow hydrograph returns to that of Figure 4.32, with a peak discharge of 72 L/min. It should be noted that the Mannings coefficient used for the channel (0.01 $s/m^{1/3}$) is different from that used by *VanderKwaak* [1999] who uses a value of 0.03 $s/m^{1/3}$. Differences between the simulations may be attributed to topographic differences between the two models, since the **HydroSphere** grid was recreated from elevations supplied by *VanderKwaak, personal communication* [1999]. The different gridding structures (tetrahedral vs. rectangular bricks) and spatial discretization approximations (finite element vs. finite difference) may also account for the differences.

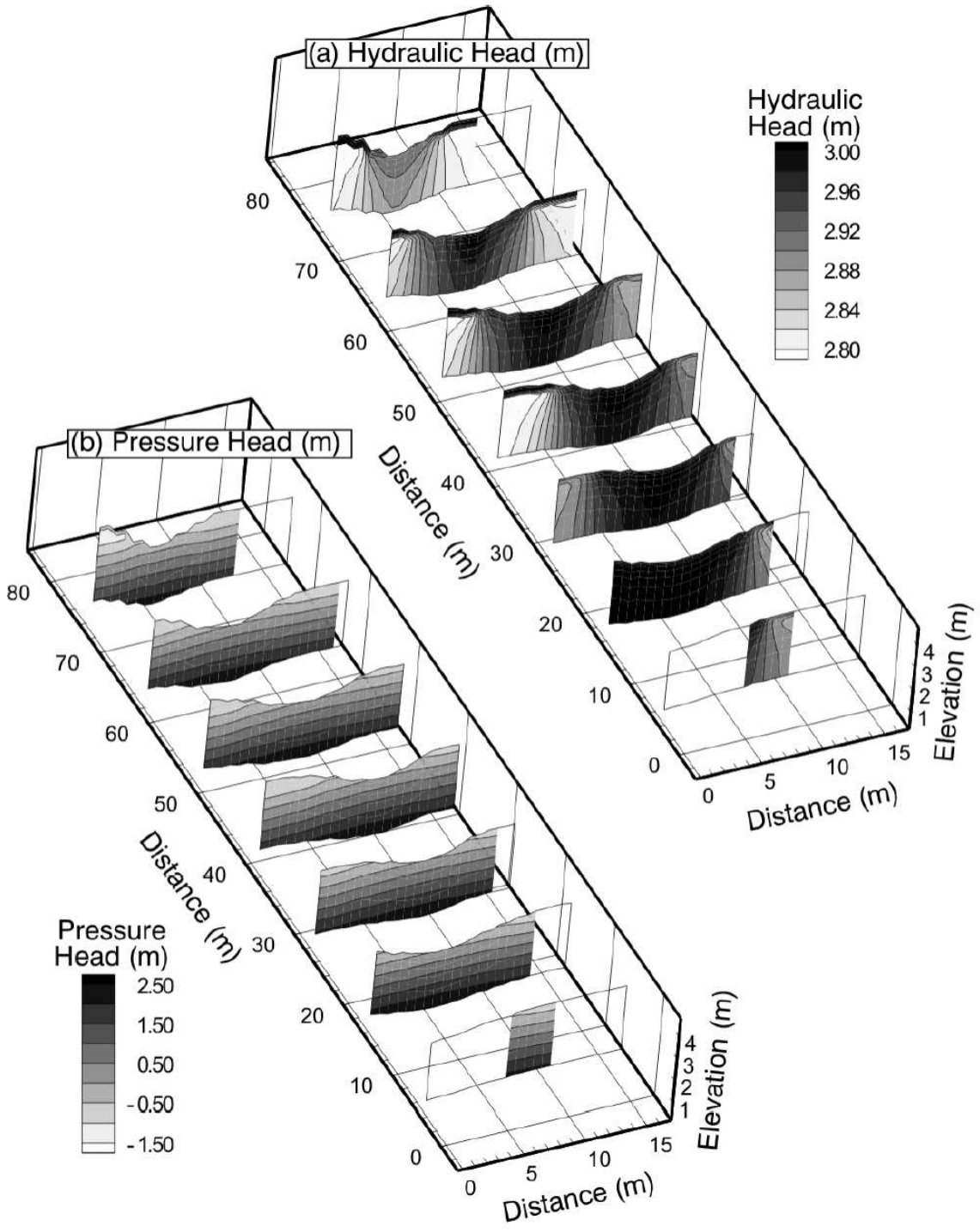


Figure 4.33: Spatial Distribution of Hydraulic Heads after 50 Minutes of Field Experiment.

Parameter	Value	
Velocity	100.0	m/yr
Dispersivity	10.0	m
Diffusion coefficient	0.0	m^2/yr
Retardation factor		
Uranium^{234}	1.43×10^4	
Thorium^{230}	5.0×10^4	
Radium^{226}	5.0×10^2	
Decay coefficient		
Uranium^{234}	2.83×10^{-6}	yr^{-1}
Thorium^{230}	9.0×10^{-6}	yr^{-1}
Radium^{226}	4.33×10^{-6}	yr^{-1}
Initial source concentration		
Uranium^{234}	1.0	
Thorium^{230}	0.0	
Radium^{226}	0.0	

Table 4.9: Parameter values for chain-decay transport in a porous medium

4.3 Transport Examples

4.3.1 Chain decay transport in a porous medium

In order to verify the accuracy of **HydroSphere** in simulating the movement of a multi-component decay chain in a porous medium it was compared with the exact analytical solution CMM [*Hydrogeologic*, 1991].

The problem was set up for the three-member decay chain:

$$\text{Uranium}^{234} \rightarrow \text{Thorium}^{230} \rightarrow \text{Radium}^{226}$$

The input parameters and values for the analytical solution are shown in table 4.9:

The analytical solution can simulate 1-D, 2-D or 3-D behavior with respect to plume development. In this case, it was set up to simulate 1-D behavior.

In the numerical model, a grid which was 500 m long in the x-direction and 1 m in the y and z directions was generated. A uniform nodal spacing of 5 m was used in the x-direction. Medium properties and flow boundary conditions were assigned such that a uniform linear flow velocity equal to 100 m/yr parallel to the x-axis was produced. A constant concentration of 1.0 was specified for Uranium^{234} at the upstream x-face.

Figure 4.34 shows concentration profiles for the three members of the decay chain at a time of 10000 years for both CMM and **HydroSphere**. It can be seen that the results are almost identical.

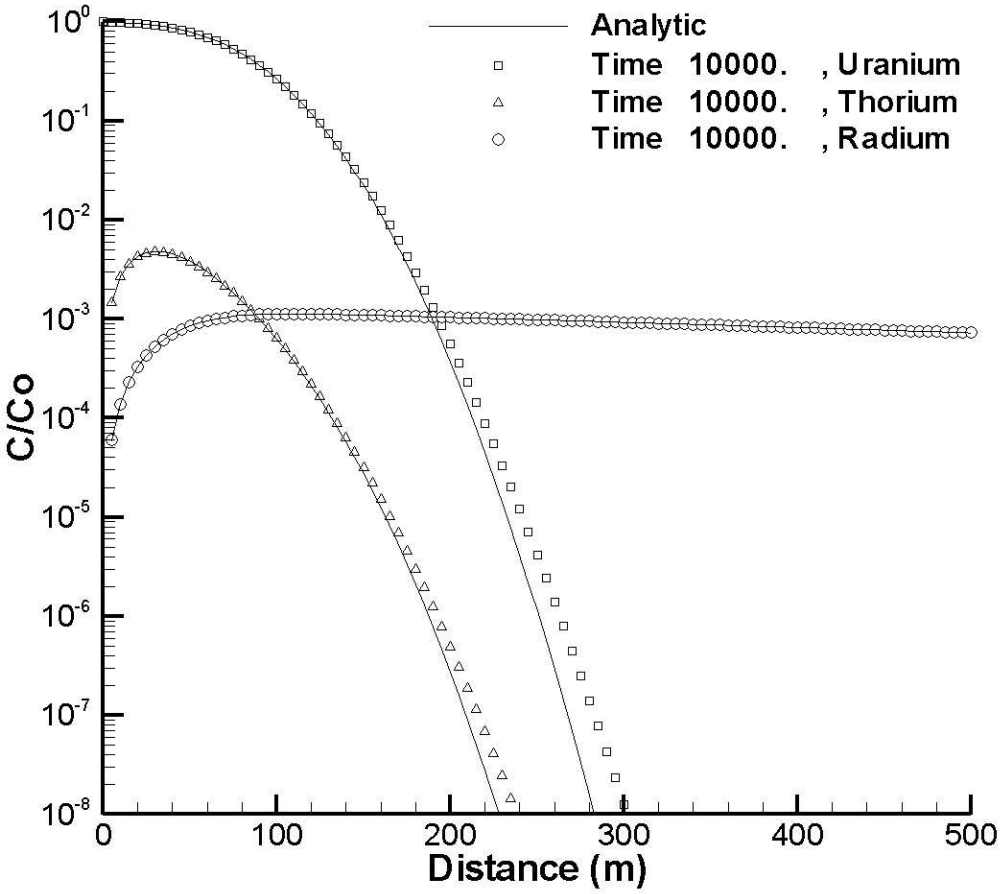


Figure 4.34: Results for a 3-member decay chain in a porous medium at 10000 years

4.3.2 Chain decay transport in a single fracture

In order to verify the accuracy of **HydroSphere** in simulating the movement of a multi-component decay chain in a fractured media it was compared with the exact analytical solution DKCRACK [*Sudicky, personal communication,* 1994].

The problem involves the same three-member decay chain used in the porous media example described above. The input parameters and values for the analytical solution are shown in table 4.10.

Note that subsurface water velocity in the matrix is 0.0, an assumption made in the analytical solution. Movement of contaminant into the matrix blocks is solely by molecular diffusion.

In the numerical model, a grid which was 200 m long in the x-direction and 1 m in the y and 0.1 m in the z directions was generated. A uniform nodal spacing of 5 m was used in the x-direction. Medium properties and flow boundary conditions were set up such that a uniform fracture flow velocity of 100 m/yr parallel to the x-axis was produced. A constant concentration of 1.0 was specified for Uranium^{234} at the upstream end of the fracture.

Figure 4.35 shows the concentration profiles of Uranium, Thorium and Radium at 10000 years for both DKCRACK and **HydroSphere** simualtions. The results are nearly identical.

4.3.3 Time-variable source condition

In order to verify the accuracy of **HydroSphere** in simulating a time-variable source condition, it was compared with an exact analytical solution SUPER1D [*Sudicky, personal communication*, 1986] for a uniform vertical, one-dimensional flow field. A time-variable Tritium source is applied with an with an input function for Tritium Units (TU) as shown in figure 4.36. Radioactive decay is neglected in the simulation. Other input parameters for the analytical solution are shown in table 4.11.

In the numerical model, a grid which was 500 m long in the x-direction and 1 m in the y and z directions was generated. A uniform nodal spacing of 5 m was used in the x-direction. Medium properties and flow boundary conditions were set up such that a uniform average linear subsurface water flow velocity of 1.0 m/yr parallel to the x-axis was produced. The time-variable source function given above was specified at the upstream x-face.

Concentration profiles for the contaminant at times equal to 7 and 24 years are shown in figure 4.37. Results from **HydroSphere** are nearly identical to the analytical solutions.

In the numerical model, the adaptive timestepping routine was used and the maximum percent concentration change allowed was 10%. The results are very close to those obtained from the analytical solution.

Parameter	Value	
Velocity in fracture	100.0	m/yr
Longitudinal dispersivity in fracture	1.0	m
Fracture aperture	1.0×10^{-4}	m
Fracture separation	0.1	m
Matrix porosity	0.01	
Matrix tortuosity	0.1	
Inlet velocity †	100.0	m/yr
Inlet dispersion	0.0	m/yr
Diffusion coefficient ‡		
$Uranium^{234}$	3.1536×10^{-2}	m^2/yr
$Thorium^{230}$	3.1536×10^{-2}	m^2/yr
$Radium^{226}$	3.1536×10^{-2}	m^2/yr
Fracture retardation factor		
$Uranium^{234}$	1.0	
$Thorium^{230}$	1.0	
$Radium^{226}$	1.0	
Matrix retardation factor		
$Uranium^{234}$	1.43×10^4	
$Thorium^{230}$	5.0×10^4	
$Radium^{226}$	5.0×10^2	
Decay coefficient		
$Uranium^{234}$	2.83×10^{-6}	yr^-1
$Thorium^{230}$	9.0×10^{-6}	yr^-1
$Radium^{226}$	4.33×10^{-6}	yr^-1
Initial source concentration		
$Uranium^{234}$	1.0	
$Thorium^{230}$	0.0	
$Radium^{226}$	0.0	

† An inlet velocity equal to the velocity in the fracture and an inlet dispersion of 0 is equivalent to a first-type source

‡ Free solution diffusion coefficient

Table 4.10: Parameter values for chain-decay transport in a fracture

Parameter	Value	
Velocity	1.0	m/yr
Dispersivity	1.0	m
Diffusion coefficient	0.0	m^2/yr
Retardation factor	1.0	

Table 4.11: Parameter values for time-varying source tranport simulation

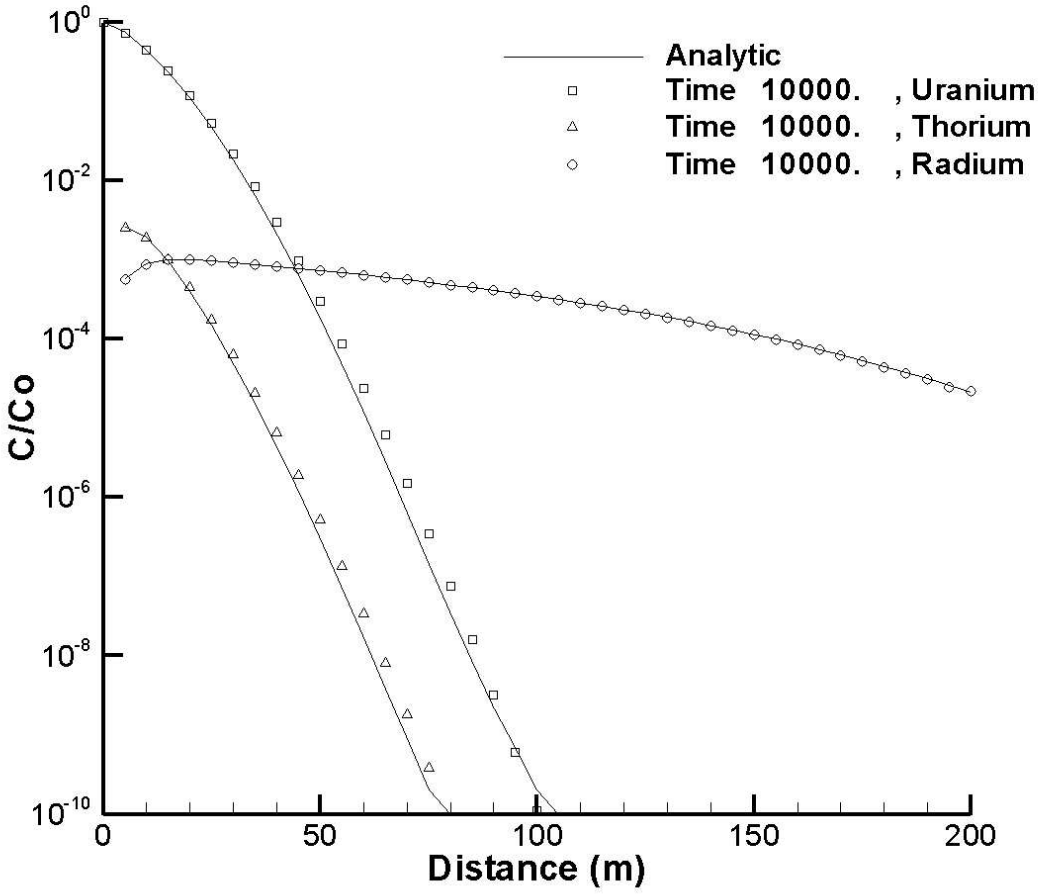


Figure 4.35: Results for a 3-member decay chain in a fractured medium at 10000 years

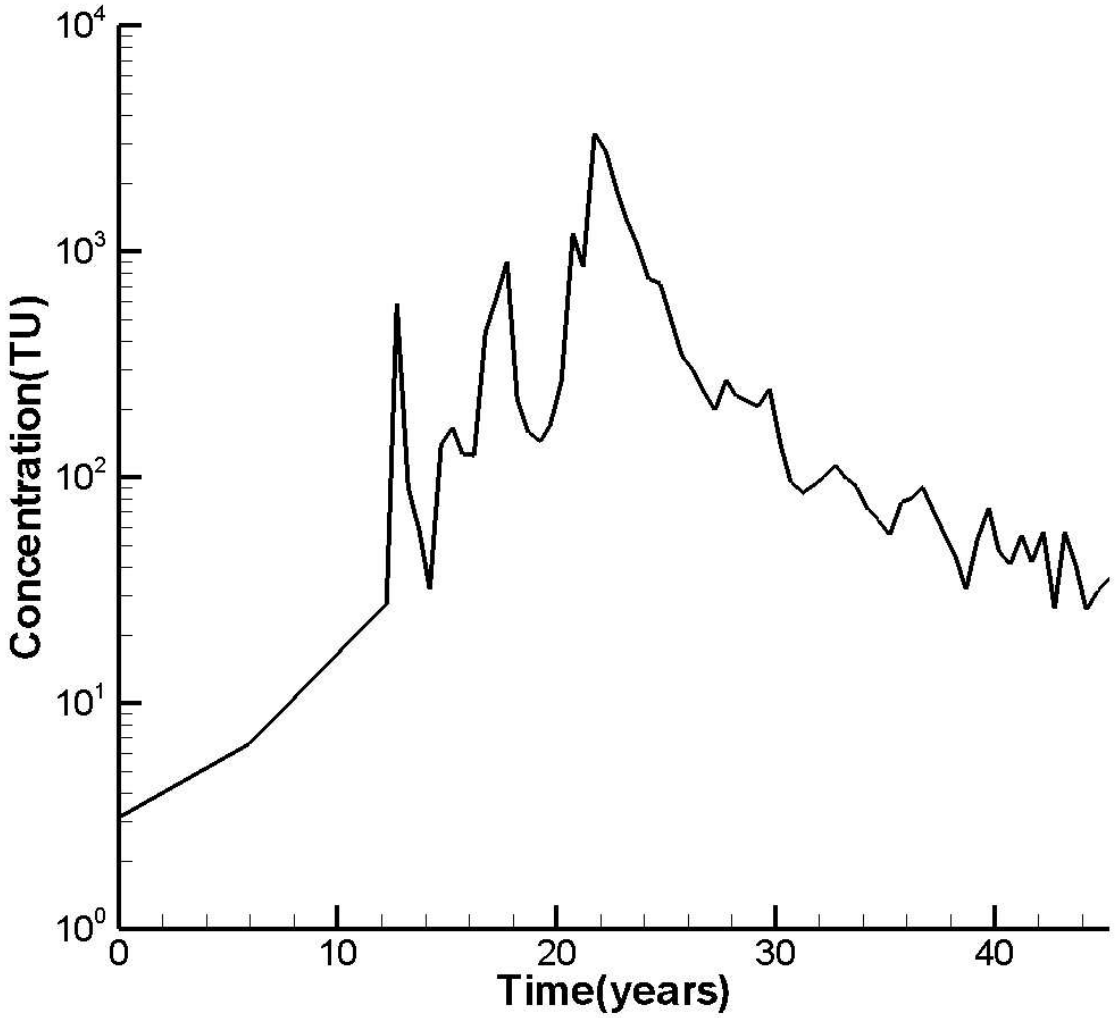


Figure 4.36: Input function for time-variable source transport

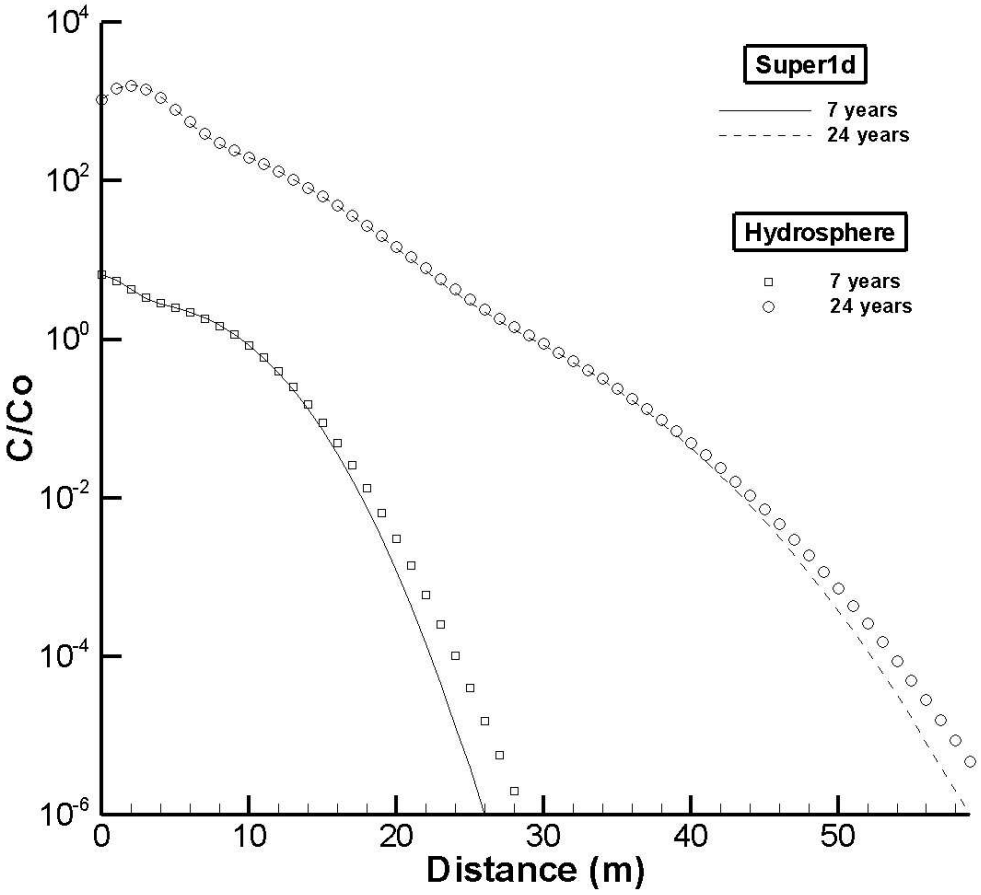


Figure 4.37: Results for a time-variable source function

Parameter	Value	
Darcy velocity	2.5	cm/d
Total porosity	0.5	
Mobile fraction	0.5	
Dispersion coefficient	10.0	cm^2/d
Retardation factor	1.0	
Mass transfer coefficient	0.1	1/d

Table 4.12: Parameter values for dual-porosity tranport simulation

4.3.4 Transport in a dual-porosity medium

In order to verify the accuracy of **HydroSphere** in simulating transport in a dual-porosity medium, it was compared with the analytical solution MPNE (*Chris Neville, personal communication*). The input parameters for the analytical solution are shown in table 4.12.

In the numerical model, a grid which was 60 cm long in the x-direction and 1 cm in the y and z directions was generated. A uniform nodal spacing of 1 cm was used in the x-direction. Medium properties and flow boundary conditions were set up such that a uniform average linear flow velocity of 10 cm/d parallel to the x-axis resulted. A constant concentration of 1.0 was specified at the upstream end of the system. The porosity of both the mobile and immobile zones was set to 0.25, which gives a total porosity of 0.5 with the mobile fraction being equal to 0.5, which is equivalent to the analytical solution input paramters. The dispersivity was set to 1.0 cm, which is equivalent to a dispersion coefficient of 10 cm^2/d when multiplied by the average linear subsurface water velocity.

Figure 4.38 shows a concentration profile at a time of 2.5 days for both MPNE and **HydroSphere**. It can be seen that the results are almost identical.

4.3.5 Transport due to an injection/withdrawal well

This verification problem considers an injection/pumping cycle for a fully penetrating well in a confined aquifer (Figure 4.39). Water with a constant concentration Co is injected into the well in the center of the domain for the first stress period. For the second stress period, the flow is reversed and the contaminated water is pumped out. The system reaches a steady state instantaneously for each stress period. An approximate analytical solution for this problem is given by *Gelhar and Collins* [1971].

The numerical model consist of 31 columns, 31 rows and one layer. All parameters are presented in Table 4.13. A constant head of 50 feet was applied on the outer boundary of the domain to produce a steady state flow-field. The simulation was done in two periods: the first stress period is an injection for 2.5 years and the second stress period is a pumping for a period of 7.5 years. The first stress period was divided into 10 time steps and the second, into 28 time steps. The simulation with **HydroSphere** is compared with the

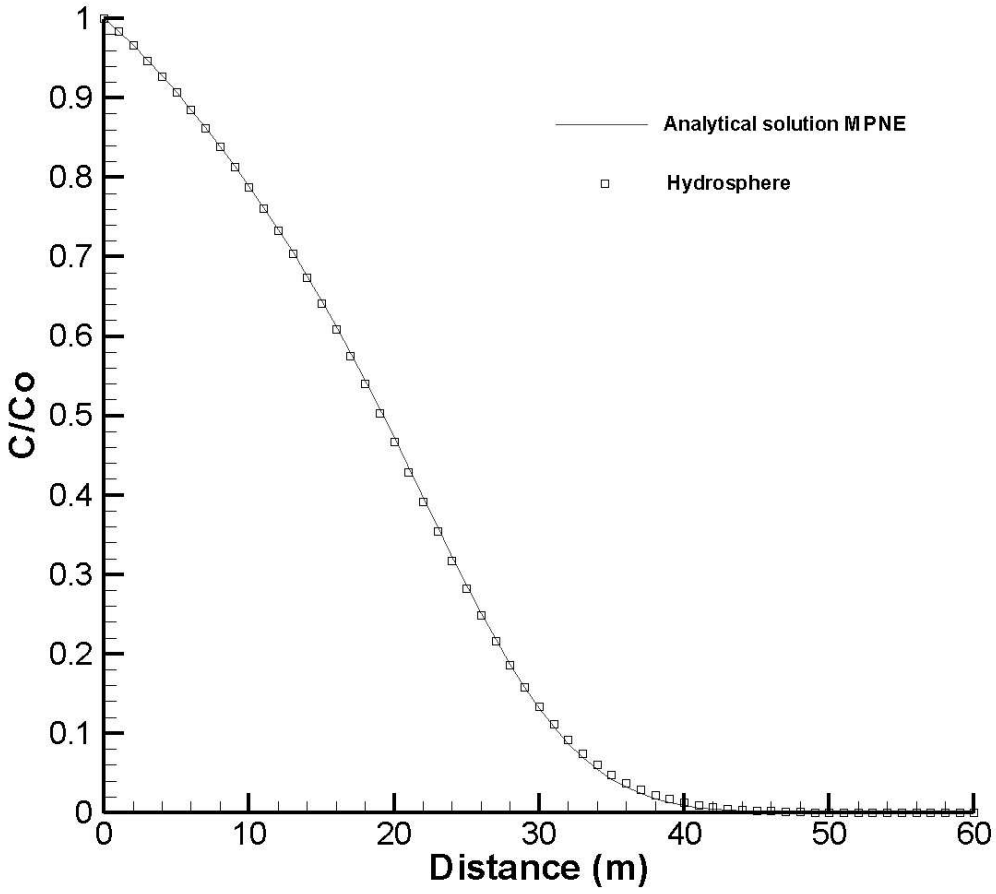


Figure 4.38: Results for transport in a dual-porosity medium

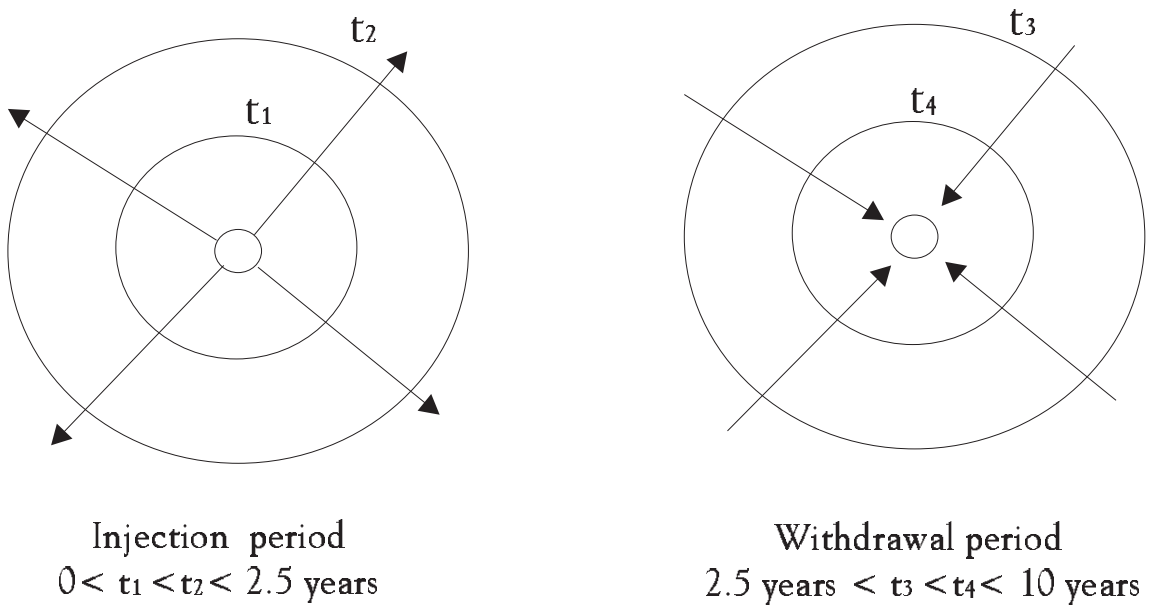


Figure 4.39: Injection/withdrawal well system

analytical solution in figure 4.40. The breakthrough curve obtained from **HydroSphere** is comparable to the analytical solution.

4.3.6 Two-Dimensional transport from a point source in a steady state uniform flow field

This problem concerns two-dimensional dispersion of solute in a uniform and steady subsurface water flow field as depicted in Figure 4.41. It assumes a small injected rate to avoid disturbance of the natural subsurface water flow field. An analytical solution for such a situation can be found in *Wilson and Miller* [1978].

Two cases involving conservative and non-conservative species were simulated using **HydroSphere**The model parameters, shown in Table 4.14, are taken from *Huyakorn et. al.* [1984b]. The selected parameter values for Case 1 were based on data from the field study of hexavalent chromium contamination reported by *Perlmutter and Lieber* [1970]. For Case 2, the values of retardation and decay constant were chosen arbitrarily to test the performance of **HydroSphere** for a non-conservative species.

Both simulations were performed using a rectangular domain containing 741 30 m $\times$ 30 m elements and 14 equal time steps of 100 days each. Concentration profiles along the x-axis at t=1400 days are plotted in Figure 4.42. Numerical results from **HydroSphere** are compared with the analytical solution, the finite-difference model MODFLOW-SURFACT (*HydroGeoLogic,* 1996), and the finite-difference solution of FTWORK(*Faust et. al.*, 1993). In each case **HydroSphere** produces solutions which are of comparable accuracy with respect to the other models.

Parameter	Value	
Cell width along rows Δx	900	ft
Cell width along columns Δy	900	ft
Thickness	20	ft
Hydraulic conductivity of the aquifer	0.005	ft/s
Porosity	0.3	
Longitudinal dispersivity α_L	100	ft
Transverse dispersivity α_T	100	ft
Volumetric injection (+)/extraction(-) rate for first/second stress periods	1	ft^3/s
Concentration during the first stress period	1	
Length of the injection period	2.5	years
Length of the extraction period	7.5	years

Table 4.13: Model parameters for simulation of transport from injection/extraction well

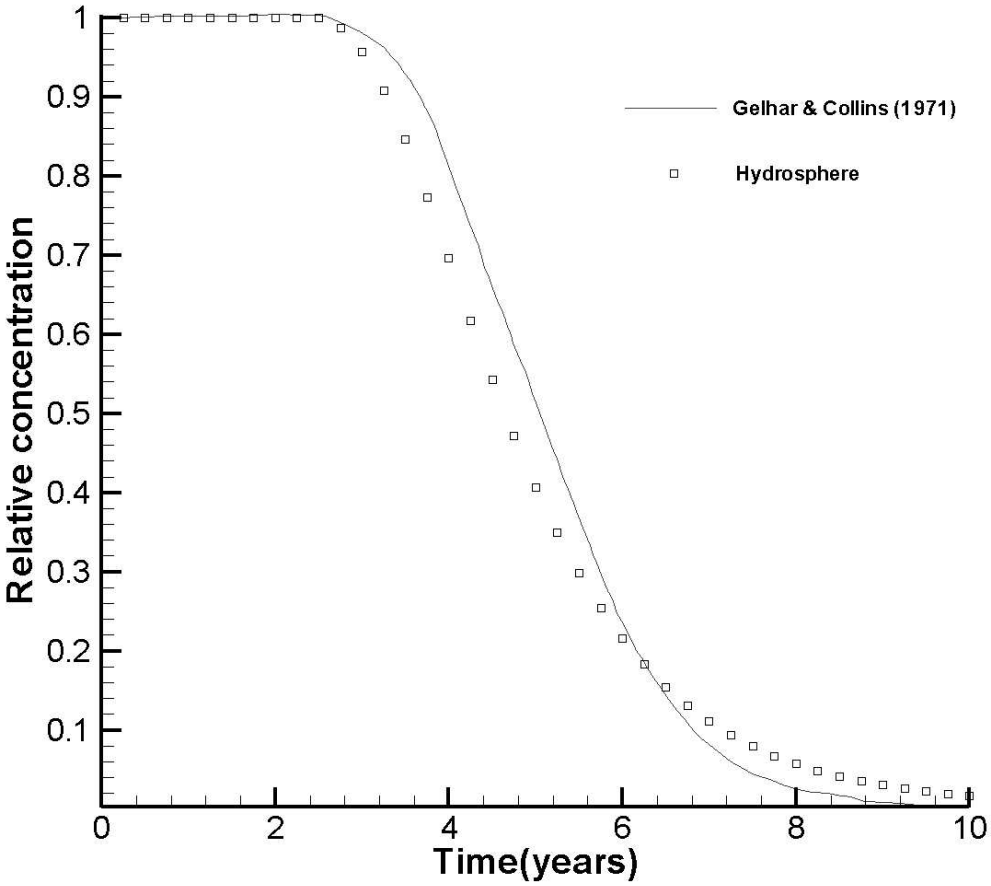


Figure 4.40: Breakthrough curve from simulation of transport due to an injection/withdrawal well

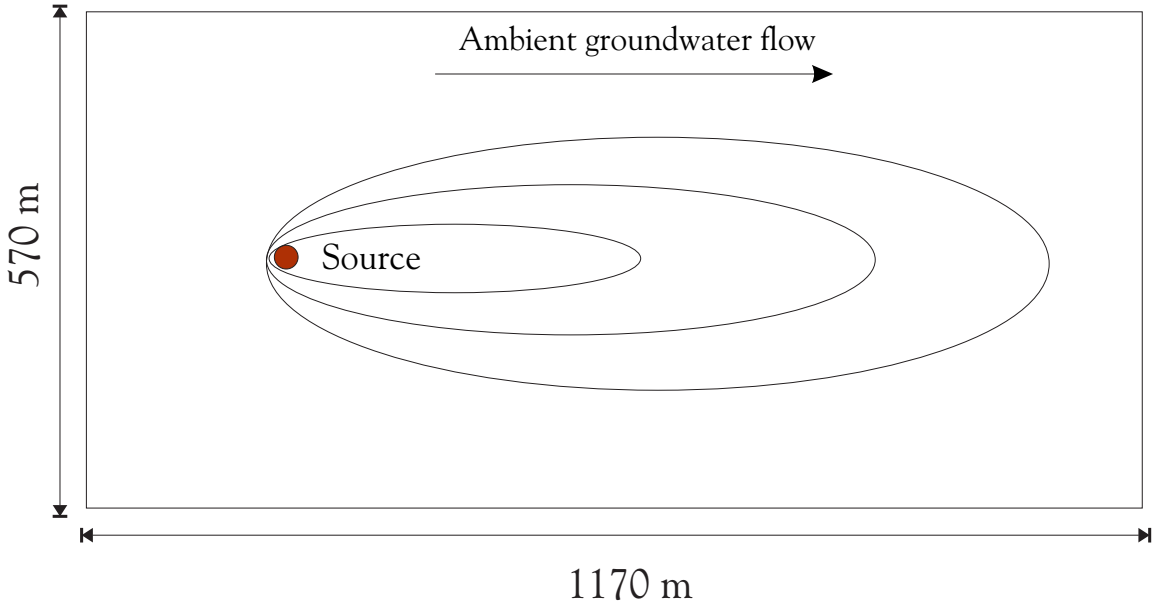


Figure 4.41: Two-dimensional transport from a point source

Parameter	Value	
Darcy velocity q	0.161	m/d
Porosity θ	0.35	
Longitudinal dispersivity α_L	21.3	m
Transverse dispersivity α_T	4.3	m
Thickness of the saturated aquifer b	33.5	m
Contaminant Mass flux per unit thickness of aquifer Qc_o	704 g/	m$\cdots$d
CASE 1:		
Linear adsorption coefficient, K_d	0	
Retardation coefficient, $R = \rho_b K_d/\varphi$	1	
Decay constant, λ	0	
CASE 2:		
Linear adsorption coefficient, K_d	0.14	m^3/Kg
Retardation coefficient $R = \rho_b K_d/\varphi$	2	
Decay constant λ	0.00019	d^{-1}
Bulk density ρ_b	2.5	Kg/m^3

Table 4.14: Parameters for simulation of 2D transport from a point source

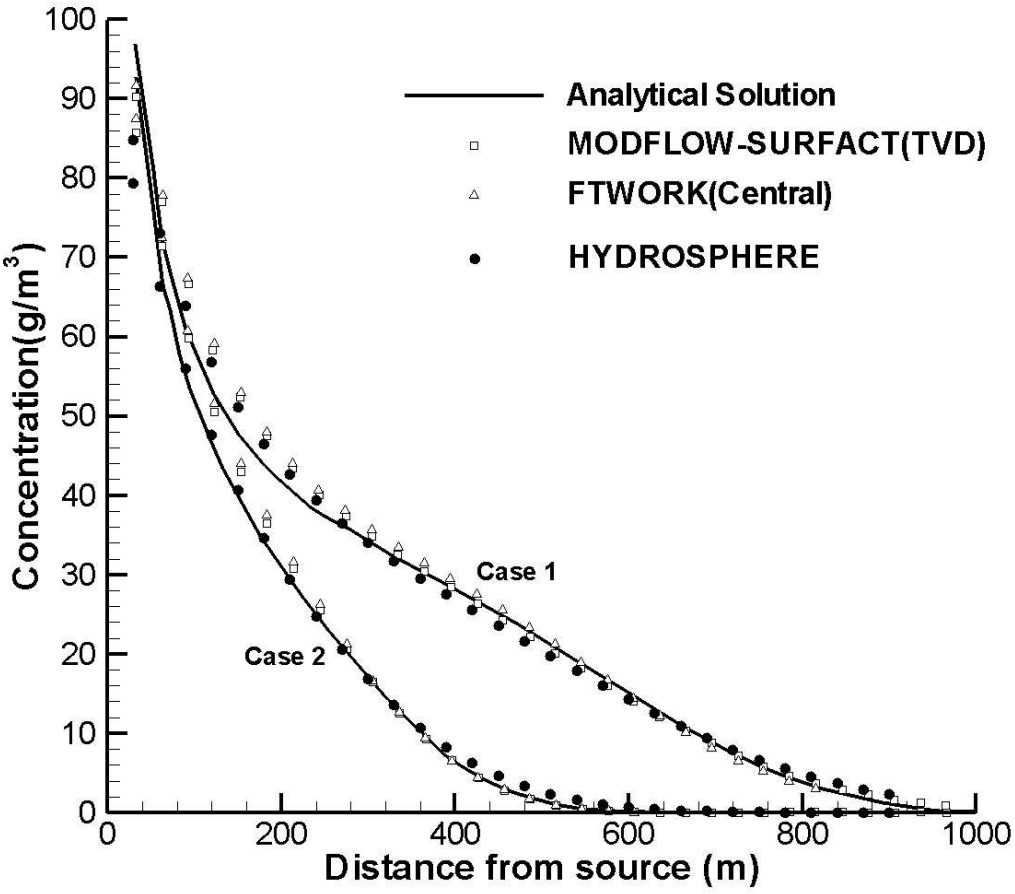


Figure 4.42: Concentration profiles along the center line for t =1400 days

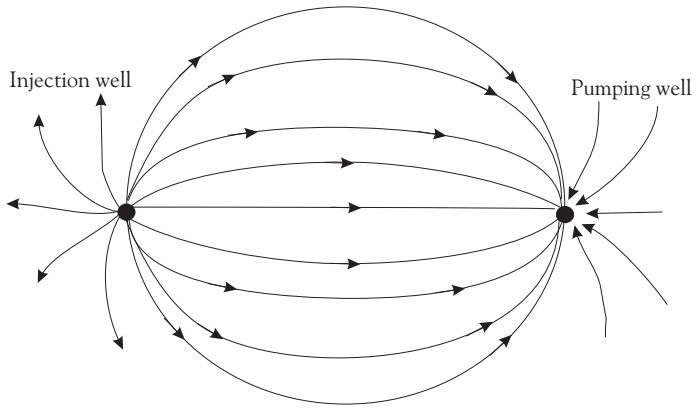


Figure 4.43: Injection-withdrawal well pair

Parameter	Value	
Well flow rate, Q_{inj}=Q_{pump}	2.339	cm^3/s
Well spacing	61.0	cm
Thickness of aquifer, b	8.9	cm
Porosity,	0.374	
Retardation coefficient, R	1	
Decay constant, λ	0	
Longitudinal dispersivity α_L	0.294	cm
Transverse dispersivity α_T	0	

Table 4.15: Parameters for simulations of transport due to an injection-withdrawal pair

4.3.7 Transport due to an injection-withdrawal well pair

This problem concerns solute transport between a pair of discharging and recharging wells. Both wells fully penetrate a constant-thickness confined aquifer that is assumed to be homogeneous, isotropic and of infinite areal extent as depicted in Figure 4.43. Wells operate at a constant flow rate and the flow field is assumed to be in a steady state.

The analytical solution which represents this case was developed by *Hoopes and Harleman* [1967]. The numerical model consist of 31 columns, 33 rows and one layer. Model parameters are presented in Table 4.15. The hydraulic conductivity used is 0.005 cm/s.

The simulation was done with fifty time steps for a total simulation of 8×10^4 s. The initial time step size is 2 s and increases by a factor of 1.2 thorough the simulation. The initial concentration in the domain is equal to zero and a unit concentration is prescribed at the injection well. Figure 4.44 presents the results obtained from **HydroSphere** which compares well with the analytical solution. Breakthrough is however diffused in the numerical solution resulting from the coarse discretization used in a complex flow field.

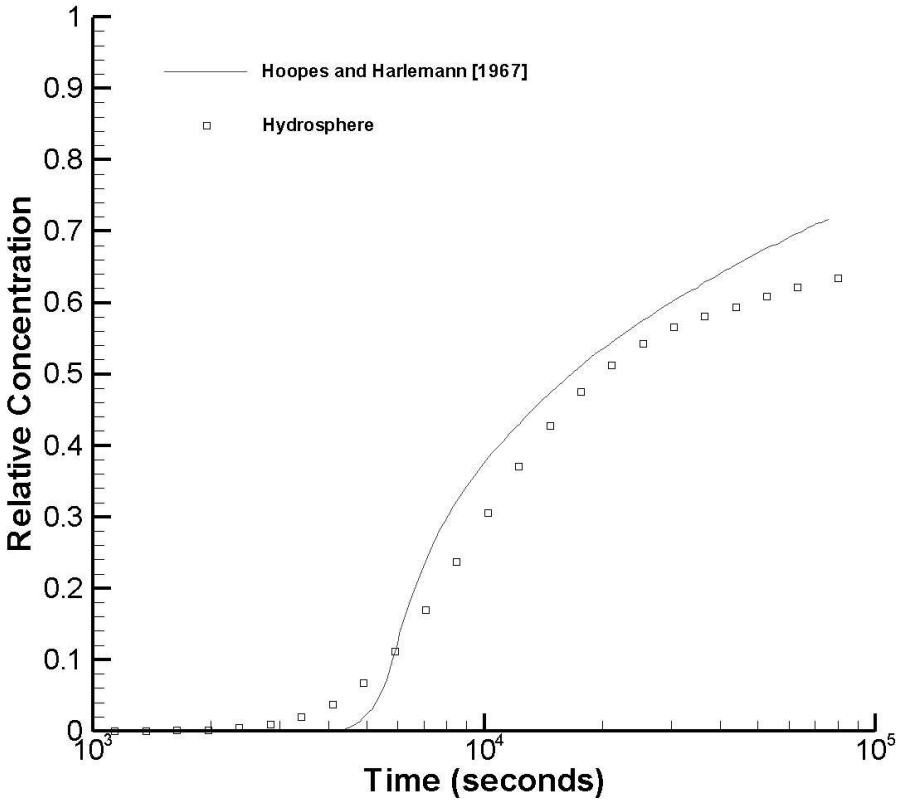


Figure 4.44: Breakthrough curve of concentration solute at the pumping well

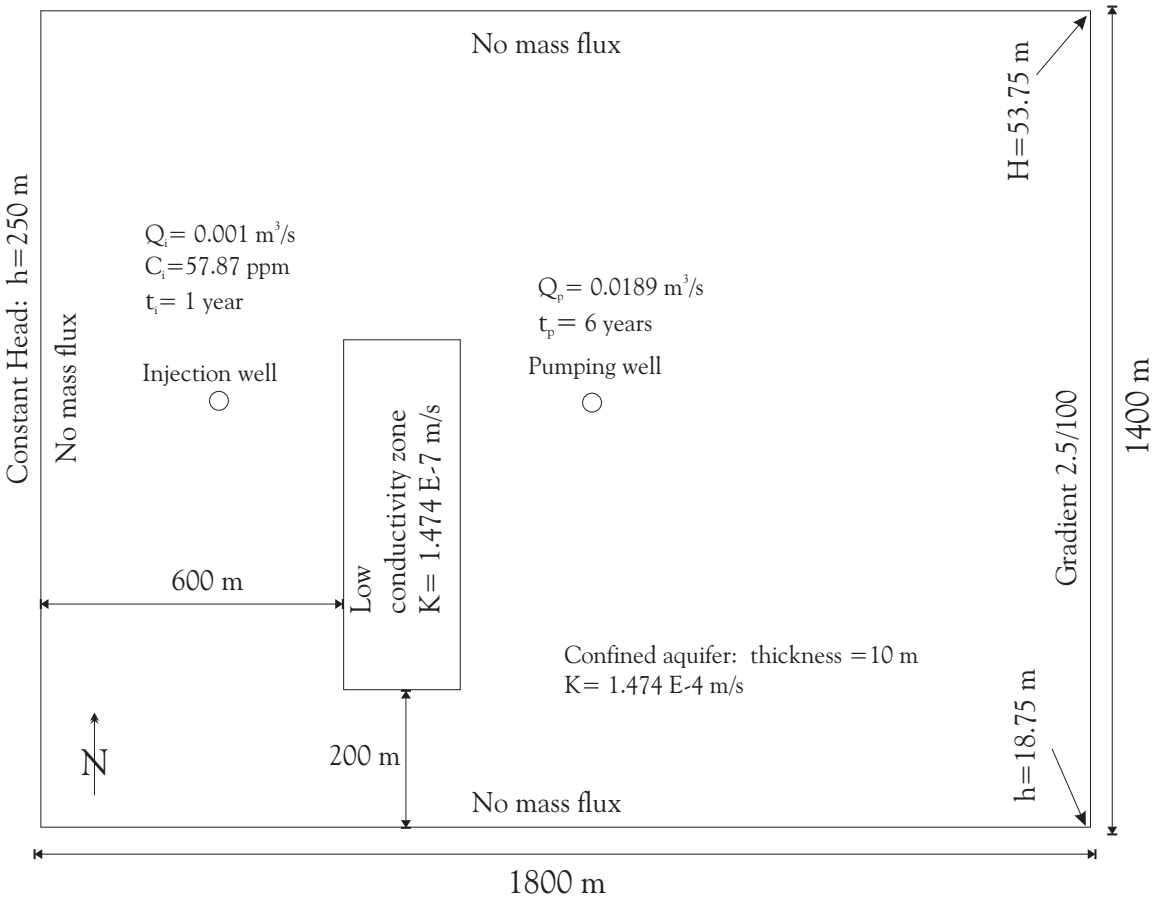


Figure 4.45: Problem description for 2D transport in a heterogeneous confined aquifer

4.3.8 Two-Dimensional (areal) transport of a contaminant plume in a heterogeneous confined aquifer with a pair of injection and withdrawal wells and strong ambient subsurface flow

This problem taken from *Zheng* [1990] is shown in Figure 4.45. It concerns contaminant transport in a heterogeneous aquifer under a strong ambient steady-state subsurface water flow field. North and South boundaries represent zero flux conditions while East and West boundaries represent constant-head conditions whereby the East boundary has also an imposed gradient. Contaminant is injected into the aquifer for a period of one year and is subsequently allowed to distribute for another five years. The total mass injected is 1825 kg (Mi=Qi Ci ti). One well located downstream pumps 0.0189 m^3/s throughout the simulation period. A low conductivity zone is located between the two wells (200 m $\times$ 600 m). For both zones, longitudinal and transverse dipersivities are respectively equal to 20 et 4 m. The effective porosity is equal to 0.3.

The grid used for **HydroSphere** simulations contains 18 columns, 14 rows and one layer of cells of uniform dimensions of 25 m $\times$ 25 m. The simulation was done with 100 equal time

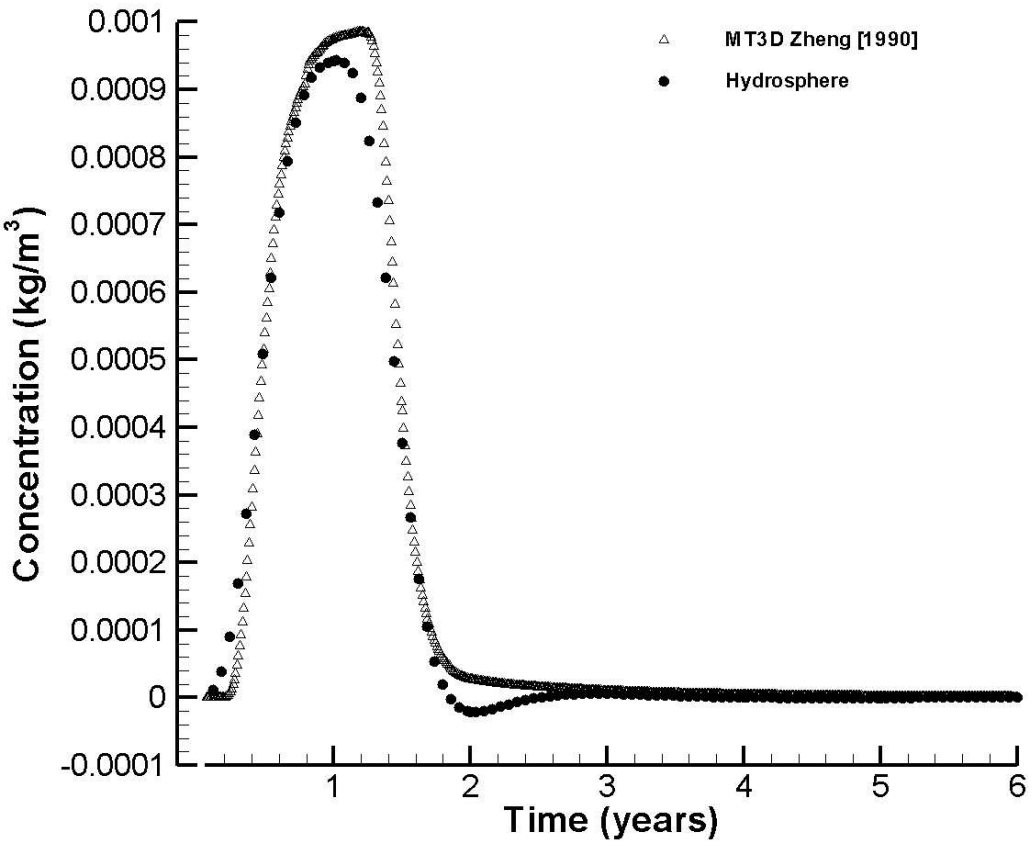


Figure 4.46: Concentrations observed at the pumping well

steps for a total time of 6 years. Results show a good correlation between the **HydroSphere** simulation and MT3D results (see Figure 4.46). **rgm Ed says fix the neg conc from hydrosphere in this fig**

The mass budget was compiled with **HydroSphere** and is presented in Figure 4.47. The mass OUT is higher than the total mass injected of 1825 kg.

4.3.9 One-dimensional transport in a contaminated aquifer remediated using a gallery and a shallow trench.

This problem concerns remediation of an aquifer contaminated with a soluble chemical from a leaky pipe. The remedial system was conceived to include a fresh-water injection gallery and a collector trench to flush the aquifer contaminant as depicted in Figure 4.48. Initially, the water table is 4 meters below the land surface and the concentration in the saturated

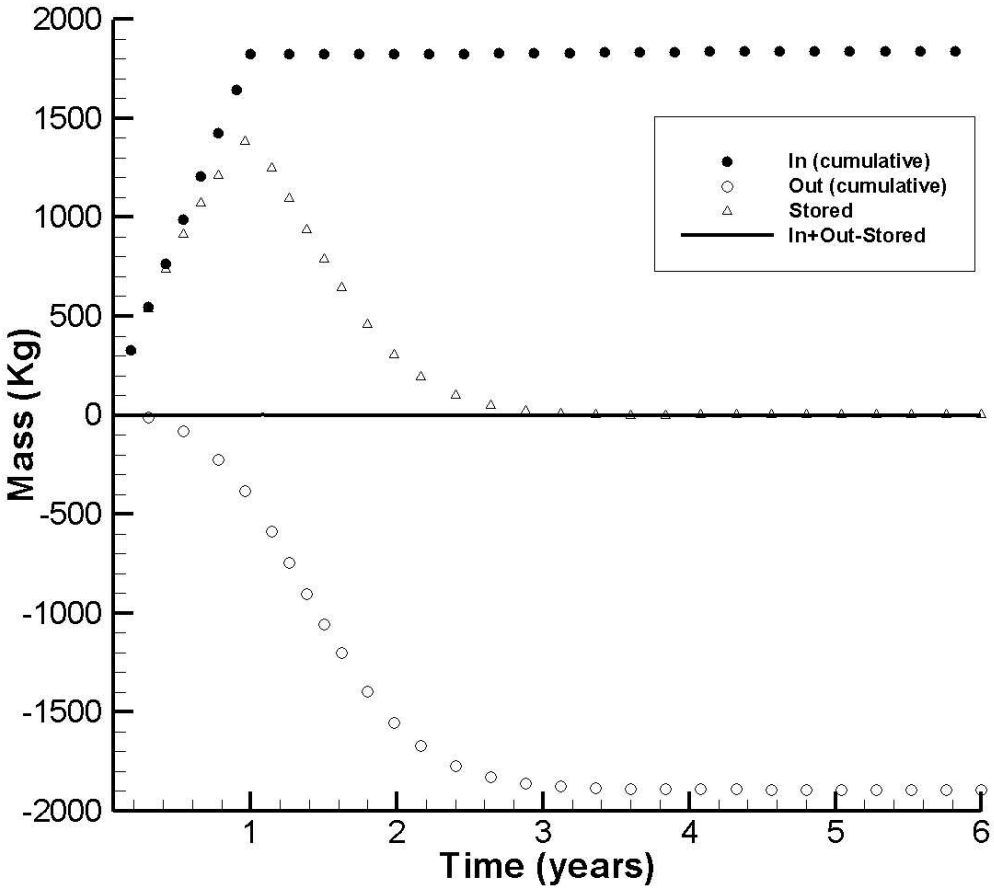


Figure 4.47: Mass budget for the 2D transport simulation

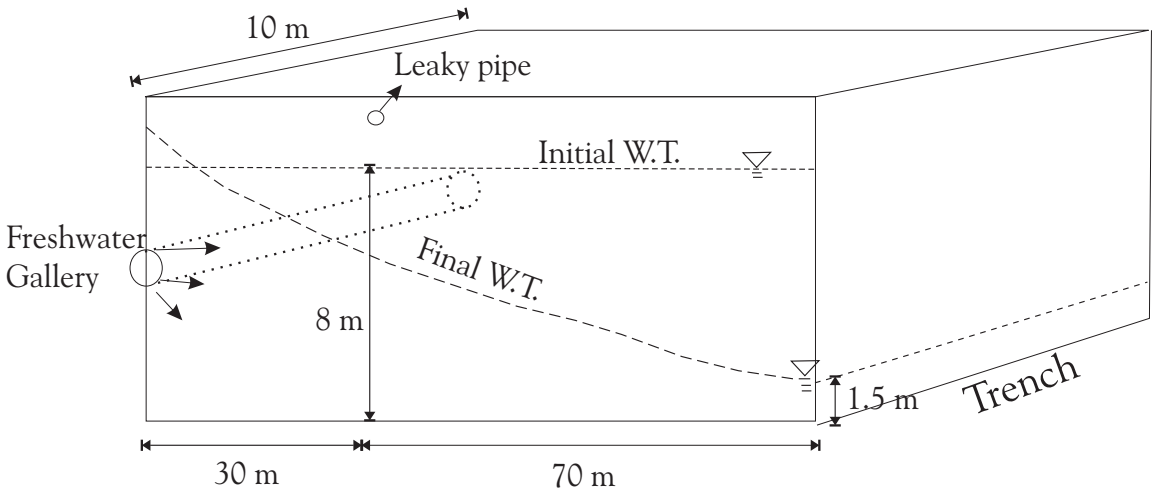


Figure 4.48: Problem description for the remediation system for an aquifer

zone is 1 kg/m^3 (length of 70 meters upstream of the trench). A constant water-level of 1.5 m is maintained in the trench. Injection of fresh water in the gallery begins after 365 days and continues for a period of 126 days with a constant flux of 2.4 m^3/d. The longitudinal and transverse dispersivities are respectively 20 and 4 meters. The hydraulic conductivity is 3 m/d, the porosity is equal to 0.3, and the specific storage is equal to 0.08.

For the purpose of the simulation, the domain between the gallery and the trench was discretized into 50 cells (50 columns, 1 row and 1 layer). The simulation was performed in a transient flow field for a total period of 850 days. Figure 4.49 presents the water-table profiles obtained at different steps of the simulation and a comparison is made with MT3D. The evolution of the contaminant cleanup is demonstrated in Figure 4.50, along with MT3D results which show a similar behaviour.

4.3.10 Two-dimensional transport of a contaminant plume in a heterogeneous confined aquifer

This problem has the same setting as the one presented in Section 4.3.8 except that the simulation is done within a transient flow field. The total thickness of the aquifer if 25 m. The set-up is shown in Figure 4.51. The total mass injected is 473.05 kg. The specific storage is estimated to be 0.01. For both zones, the porosity, the longitudinal and transverse dispersivities are equal respectively to 0.25, 20 m and 4 m.

The simulation was done in transient flow for a period of 7 years. The grid used is the same as the one in the example shown in Section 4.3.8. The pumping rate is 0.24 m^3/s for a 1 year period separated by 1 year without pumping. Pumping begins after one year and the total duration of simulation is seven years which includes 3 pumping stages. The initial time step size is 1.2 days which is incremented by a factor of 1.5 up to a maximum time step size of 12 days. Hydraulic heads observed at the pumping well are presented in

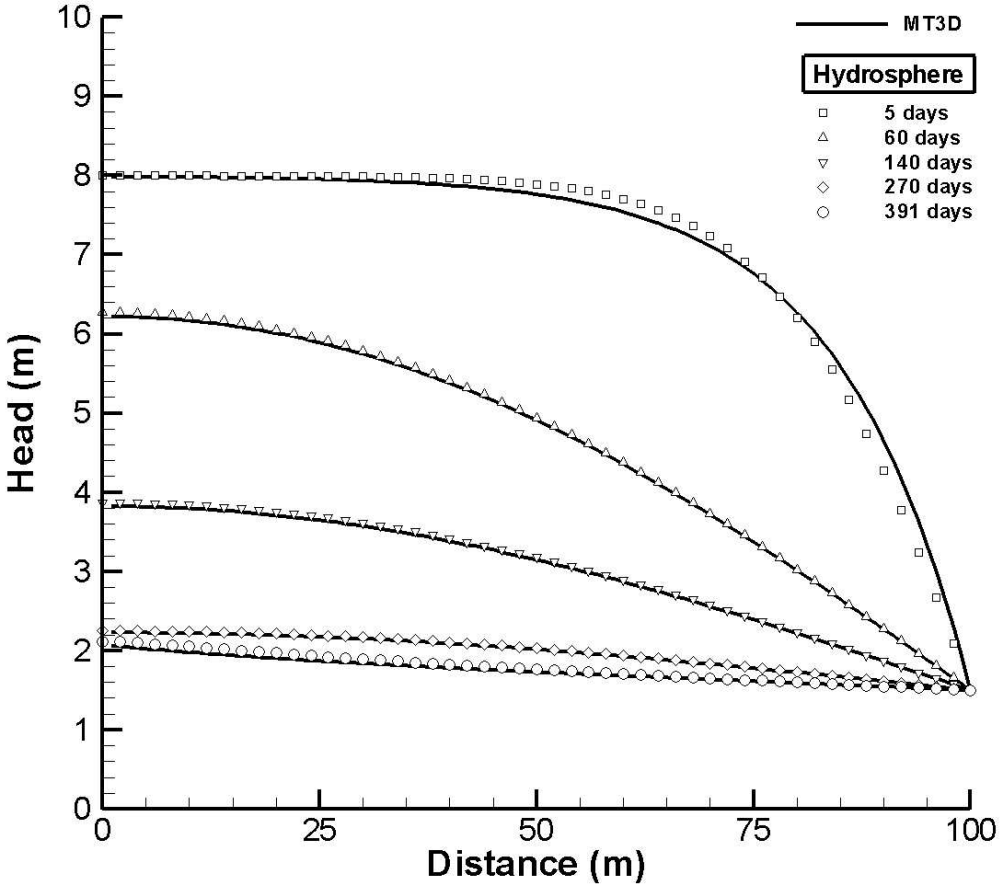


Figure 4.49: Water-table profiles from the 1D flow simulation

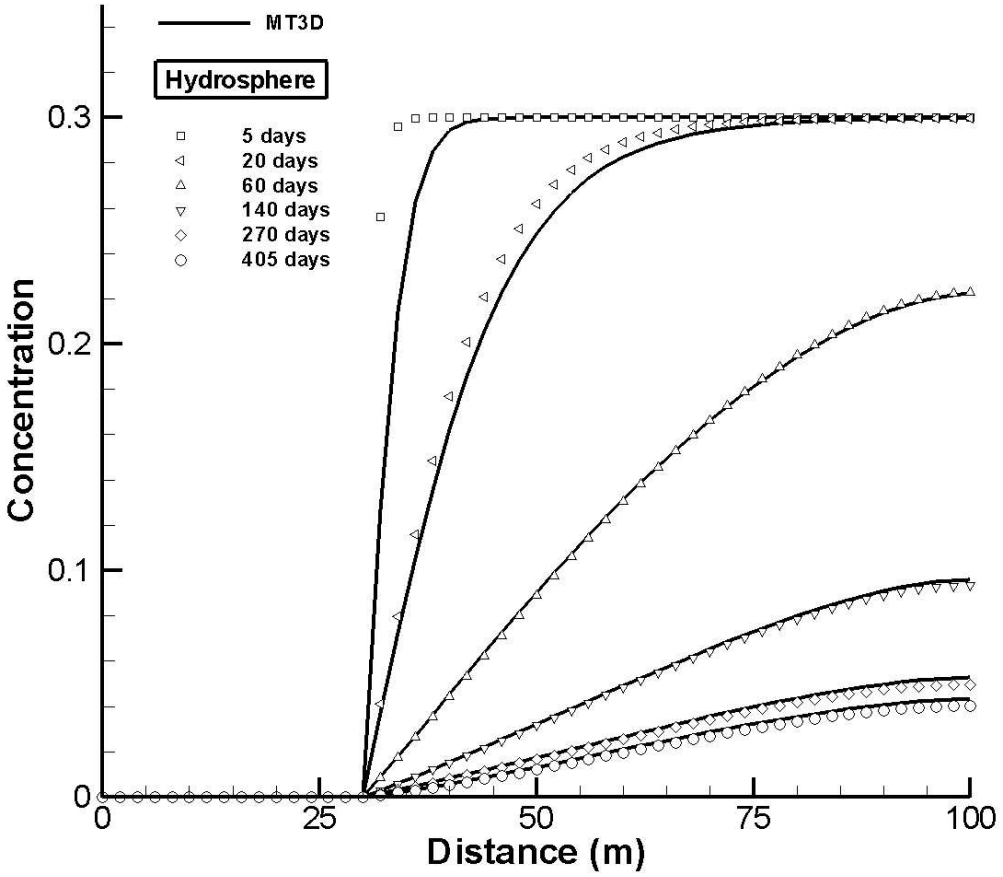


Figure 4.50: Concentration profiles from the 1D transport simulation

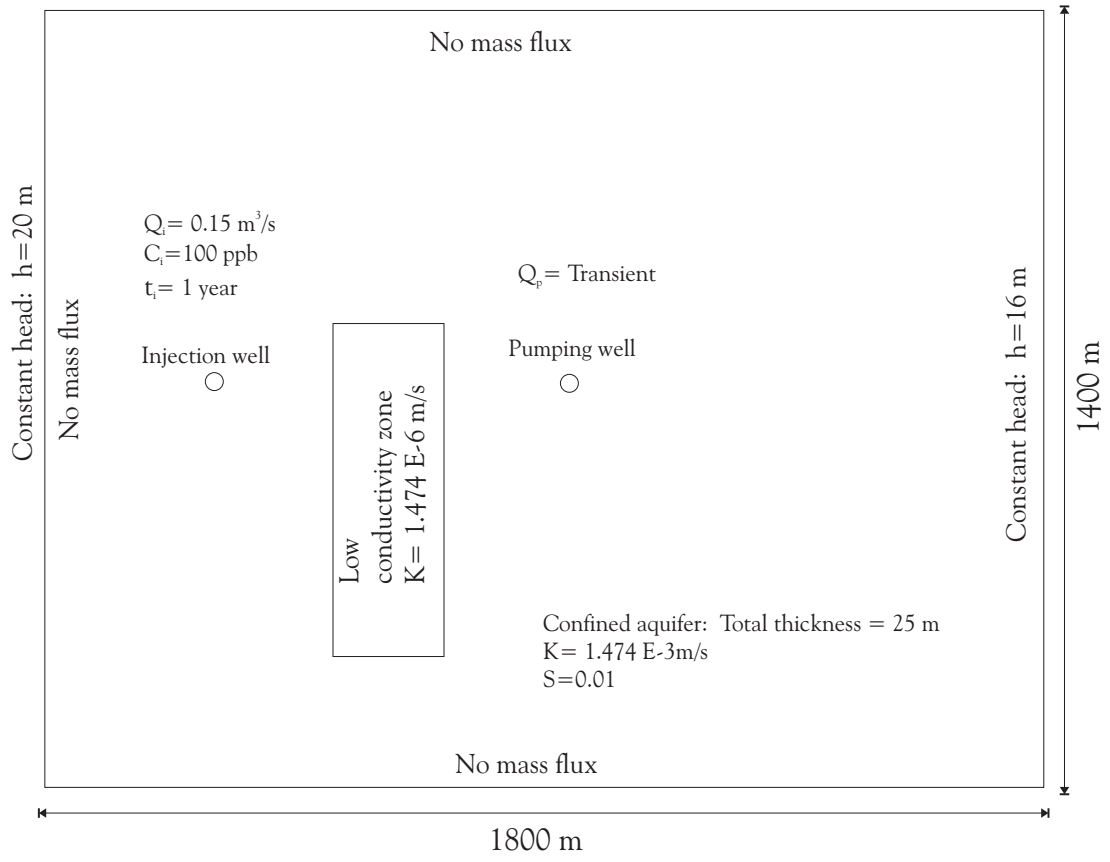


Figure 4.51: Problem description for 2D transport

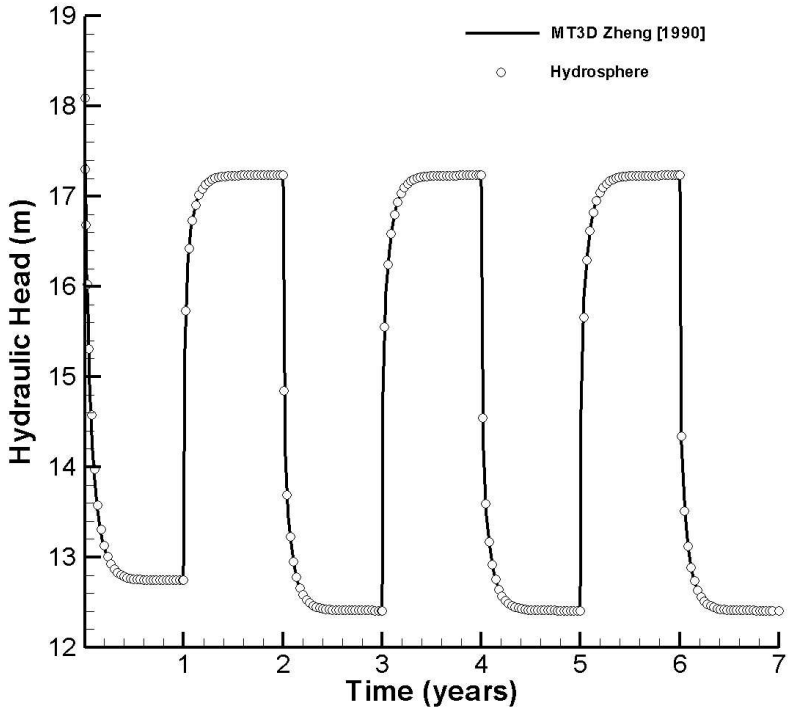


Figure 4.52: Hydraulic head distribution at the pumping well during the simulation

Figure 4.52. The comparison is made with MODFLOW.

Solute concentrations at the pumping well were evaluated during the simulation and are presented in Figure 4.53. **HydroSphere** results compare fairly well with the breakthrough obtained from MT3D. The solute mass balance evaluated in Figure 4.54 shows negligible mass balance errors throughout the simulation. The total mass OUT is about 474 g, a little more than the total mass IN which is 473.05g.

4.3.11 Two-Dimensional transport of contaminant in the water phase of an unsaturated rectangular soil slab

This problem corresponds to a 2D unsaturated flow problem discussed earlier in chapter 4. It concerns transport of a non-conservative solute in a transient unsaturated flow field. This system is initially dry and free of solutes and water with dissolved solute is allowed to enter the system at the upper portion of the left hand boundary as shown in Figure 4.55. Inflow head is 6 cm with a prescribed solute concentration of 1 ppm. Outflow occurs along the entire right hand side boundary under initial pressure conditions of -90 cm. The remaining boundaries are under no-flow, zero concentration gradient conditions.

The domain dimension is 15 cm horizontally and 10 cm vertically, discretized using a grid

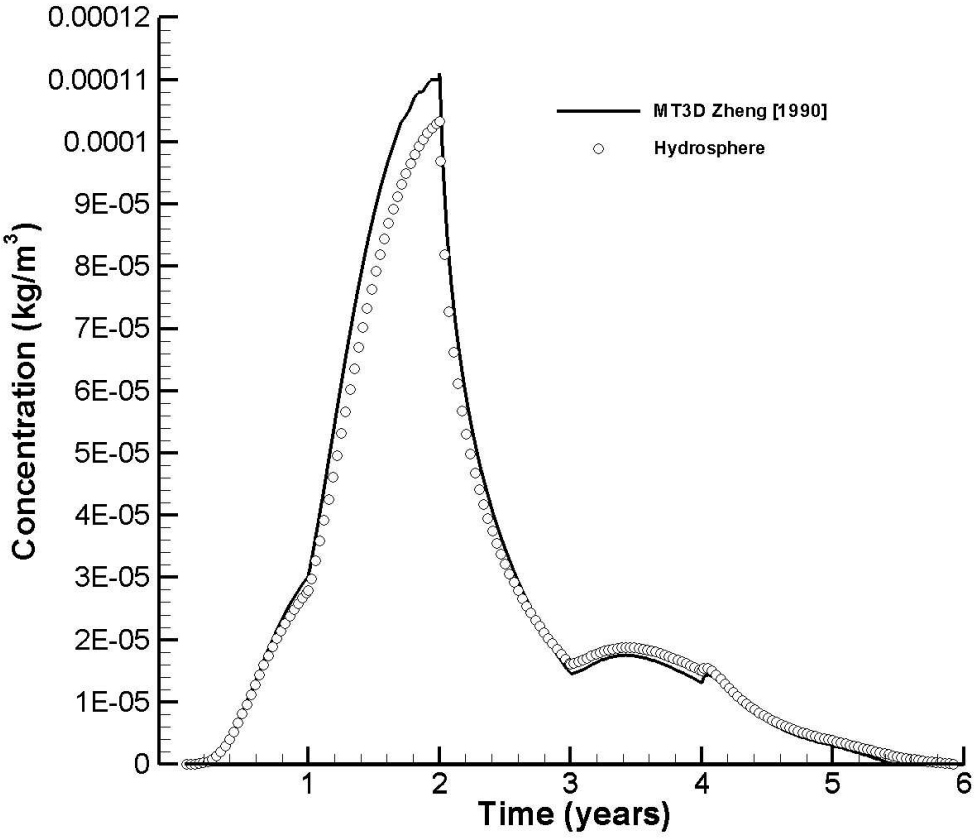


Figure 4.53: Breakthrough curve observed at the pumping well for 2D transport simulation

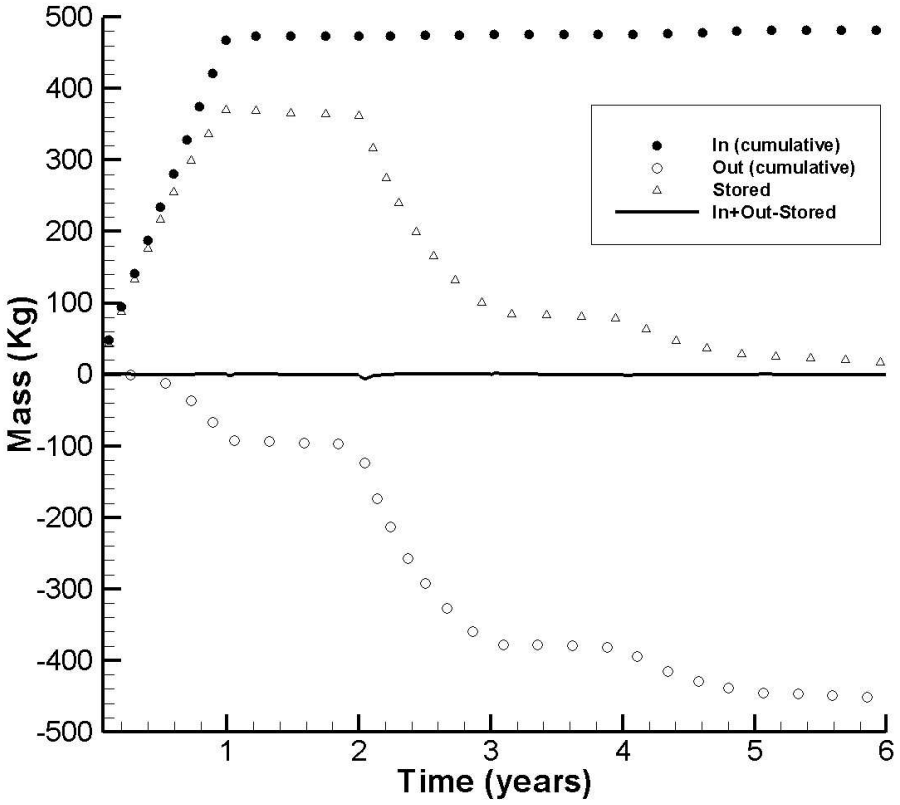


Figure 4.54: Solute mass balancet for 2D transport simulation in transient flow field

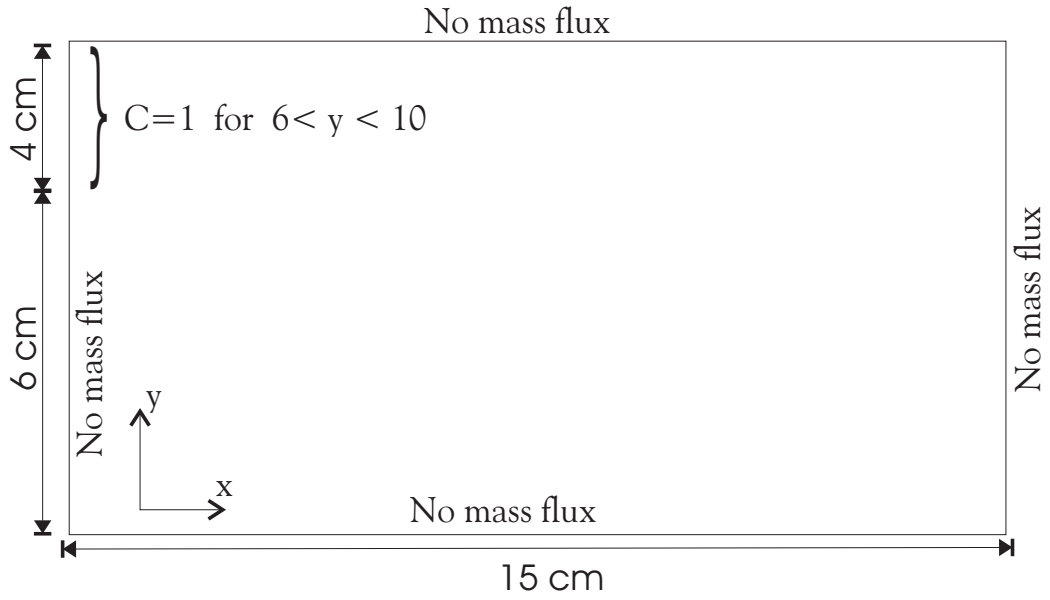


Figure 4.55: Problem description for 2D transport in an unsaturated rectangular soil slab

Parameter	Value	
Saturated hydraulic conductivity, K	1	cm/d
Specific storage, S_s	1×10^{-14}	
van Genuchten α	0.005	cm^{-1}
van Genuchten β	2	
Brooks Corey parameter, n	2	
Residual water saturation S_{wr}	0.3333	

Table 4.16: Hydraulic properties of the rectangular soil slab

with constant cell spacing of 1cm. The soil is assumed to be homogeneous and isotropic. Table 4.16 presents the hydraulic properties of the soil. The simulation was done with an initial time step size of 0.01 days which is enlarged with a factor of 1.2 up to a maximum time step size of 0.05 days. The physical transport parameters are shown in Table 4.17. The total duration of the simulation is 0.508 days. The values of concentration in the rectangular soil slab are computed for t=0.508 d and are presented in Figure 4.56.

Parameter	Value	
Porosity	0.45	
Initial concentration C_o	0	
Longitudinal dispersivity α_L	1	cm
Transverse dispersivity α_T	0	
Molecular diffusion D_o	0.01	cm^2/d
Decay constant	0.001	d^{-1}
Bulk density	1.46	g/cm^3
Distribution coefficient K_d	0.308	cm^3/g

Table 4.17: Physical parameters values for simulation of transport in an unsaturated rectangular soil slab

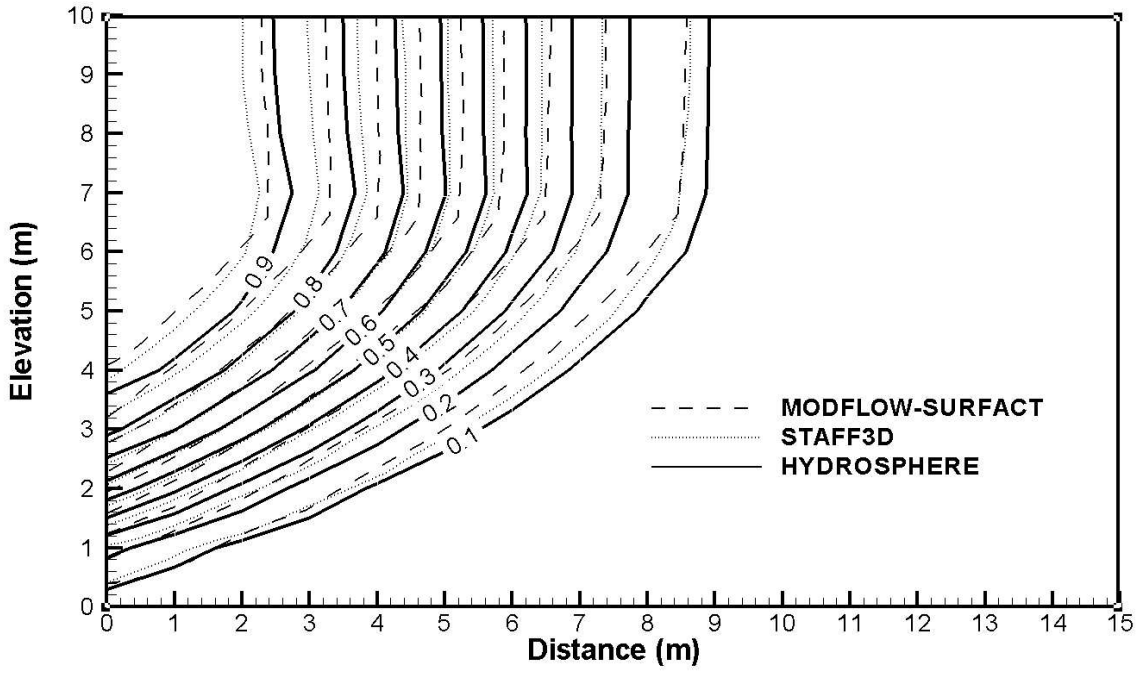


Figure 4.56: Simulated contaminant concentrations in an unsaturated rectangular soil slab at 0.508d

Chapter 5

Input/Output Instructions

5.1 General

Before presenting in detail the input data needed for the numerical simulations, some general information about the format and nature of the input data is first given.

There are two steps involved in solving a given problem. First, a data file is prepared for the pre-processor (called **grok**), which is then run to generate the input data files for **HydroSphere**. Second, **HydroSphere** is run to solve the problem and generate output data files.

The **grok** input file name consists of a meaningful prefix of up to seven characters to which the extension `.grok` is appended. This prefix will determine the input and output filenames. The **grok** listing file name will be the problem prefix to which the letter `o` and the file extension `.eco` are appended. For example, if the problem prefix specified by the user is `test`, the general input file to be created by the user will be `test.grok` and the output listing, or echo, file generated by the pre-processor will be `testo.eco`. Some simulations will require more than one input file (e.g. initial heads read from file) and will result in the generation of more than one output file. As a rule, all input files needed during a specific simulation will have the problem prefix plus a given extension as filename while all generated output files will have the problem prefix, the letter `o`, plus a given extension as filename.

After the pre-processor starts executing, it prompts the user to enter the prefix for the problem interactively from the keyboard. For cases in which the same input file is being used repeatedly, you can create a file called `batch.pfx` which consists of a single line which contains the problem prefix. If the file is present, the prefix will automatically be read from the file and you will not be prompted to enter it from the keyboard. This file should be placed in the same directory as the `.grok` file.

Briefly, the pre-processor performs its tasks in the following order:

1. Read and allocate default array sizes

2. Read problem identification information
3. Read instructions for generating grid
4. Perform grid modifications if necessary
5. Generate default properties for all parameters
6. Read optional instructions for modifying the default parameters
7. Write the **HydroSphere**-compatible data files

Tasks 3 and 6 are guided by instructions issued by the user in the `.grok` file. The generation of a complete set of default data by task 5 tends to minimize the amount of data which must be supplied by the user.

Throughout this manual, we will adopt the convention of using sans serif font for representing pre-processor instructions. In addition, when an instruction is introduced and described, it will be highlighted as shown in the following example:

Example instruction text

The instructions are enclosed in a box, and must be typed in the `.grok` file *exactly as shown*, with the exception that they are not case-sensitive, and blanks before and after the instruction are optional. Note that only one blank is allowed between any two words in an instruction. Following the instruction is a brief description of what it does and details regarding any required input. If the instruction requires input data, there will follow a series of numbered lines, each containing bolded variable names. Each numbered line corresponds to one FORTRAN read statement and the number of items required in the data file are indicated by how many bolded variable names are present on the line. The default FORTRAN variable naming conventions are in effect. This means variables starting with the letters I–N inclusive require integer values, while all the rest require real values, unless stated otherwise in the case of string or logical variables. Numerical values are read in free-format so integers and reals do not need to be lined up in columns and they can be separated by blanks or commas. A descriptive comment can be included inline after the last data value has been read from the line.

As the pre-processor reads and processes the `.grok` file it also creates the `o.eco` file. Results of the **HydroSphere** data generation procedures are written to this file so if there are any problems reported by the pre-processor you should check this file first to determine their nature and how you might fix them. The most common error is to mis-type an instruction, in which case the pre-processor writes the following to the screen:

```
unrecognized instruction
```

You should now check the last line of the `o.eco` file to see which instruction is causing the problem.

The units used in the program are not preset. The user must determine which units will be used for length (L), time (T) and mass (M) for the various input variables and consistently

use those chosen units. For example, if you want to specify the dimensions of your domain in metres and the time at which you want a solution in seconds, then all measures of time and length will have to be in seconds and metres, respectively. The hydraulic conductivity should therefore be specified in metres/second, a pumping rate in metres^3/second etc. The program does not perform any checks to ensure unit consistency.

Any line in the `.grok` file which is completely blank or which begins with an exclamation point(!) will be ignored. This allows you to include comments whenever required.

5.1.1 File process control options

The following instructions control how the pre-processor treats instructions in the `.grok` file. They can be inserted at any point in the `.grok` file, except of course when input for a specific instruction is expected, and as often as required.

Echo off

By default, as instructions are read by **grok** they are echoed to the screen. This command turns off this feature.

Echo on

This commands turns on the echoing of instructions to the screen.

Skip on

With skip mode turned on, **grok** will read but not act on any subsequent instructions.

Skip off

Turns skip mode off, so **grok** will resume acting on instructions.

Pause

This instruction causes **grok** to pause at the current location in the `.grok`file until the user presses a key.

We will now describe in detail the various actions of the pre-processor, giving instructions for setting up the `.grok` file where necessary.

5.1.2 Simulation control options

Data check only

This instruction causes **HydroSphere** to stop prior to the start of the solution procedure but after initialization of arrays, grid and checking of array sizes. This can be useful for very large problems, where it is desirable to make sure that all the input is correct before actually doing the simulation.

Finite difference mode

Changes the numerical method used to finite differences instead of the default finite elements. This mode can only be used for block element grids and if you use the instruction in any other case then the pre-processor will stop and issue a warning.

Y vertical

This instruction is intended to be used for variably-saturated flow conditions and triangular prism elements, when one wants to have the triangular mesh (which is defined in the x-y plane) to be oriented along the vertical direction.

It instructs **HydroSphere** to assume that the y-coordinate is along the vertical direction (instead of the z-coordinate). This instruction *does not switch coordinates*, it simply tells the code to use the y-coordinate of a node to calculate the total hydraulic head (pressure + elevation) for variably-saturated simulations.

5.1.3 Default Problem

Once the grid generation step is completed, the pre-processor generates a set of data for a default problem by assuming *saturated, steady-state flow in a non-fractured, homogeneous porous medium*. By default, a transport simulation is not done.

The porous medium properties for the default problem, which are hardwired in the code, are listed in Table 5.1.

If the default properties are acceptable, the only additional data required to complete the definition of the problem are flow boundary conditions which can be assigned as described in section 5.6.5.

If the default properties are not acceptable, they can be modified as desired by issuing further instructions in the `.grok` file.

5.1.4 Manipulating Zoned Properties

Before we begin discussing the concepts of zoned properties we must define what they are. There are currently four basic types of media which can be defined in **HydroSphere**:

1. porous
2. dual continuua
3. discretely-fractured
4. overland flow

Porous media and dual continuua are defined by three-dimensional 8-node (brick) or 6-node (prism) elements while discretely fracture and overland flow media are defined by two-dimensional 4-node (rectangle) or 3-node (triangle) elements. By default, every 3D

element in the problem domain is a porous media element. Elements of the other three types of media may or may not be defined for a specific problem.

Each porous media element is assigned a zone (i.e. material) number during grid generation. In simple cases, all elements will be assigned a zone number of 1, while in more complex cases, elements may have been assigned different zone numbers. For example, if a multilayered grid was generated using the instruction Generate layers from slice then the elements would have been assigned zone numbers based on the layer number (i.e. elements in the lowest layer number 1 would be assigned a material number of 1).

By default, all zones, and therefore all elements in the domain, are assigned the same default porous media properties, which are listed in Table 5.1. These values are set in software and cannot be modified by the user unless the code is changed and recompiled. However, there are other ways of changing the porous media zone properties as we will discuss below.

The first step in modifying zoned properties for a given problem is to indicate which type of medium is to be manipulated. The following instruction does this:

Use zone type

Instructs **grok** to read a string defining the type of medium to which the generic instructions that follow should be applied. The following input is required:

1. **zone_type** Can be one of the strings; porous media, dual, fracture or overland.

5.1.4.1 Defining a New Zone

In order to define a new zone, elements of the proper type must first be chosen using the instructions given in Section 5.5. For example, 3D block elements must first be selected before a new porous media or dual zone can be defined while 2D rectangular faces are selected for fracture or overland zones.

In the case of the dual, fracture or overland zone types, a new zone must be defined since the default situation after grid generation is that there are no elements of these types.This is not the case for porous media zones.

Once elements of the appropriate type have been chosen, the following command groups them into a single zone.

New zone

Generic. Assigns a zone number to the current set of chosen elements. It requires the following input:

1. **num_zone** The zone number to be assigned. If num_zone is greater than the total number of zones of the current media type, the total number of zones will be incremented and default properties for that media type will be assigned.

5.1.4.2 Modifying zone properties

There are a number of instructions which can be used to modify the property values associated with a zone or group of zones. Before these instructions are issued, it is necessary to select the appropriate type of media and then choose the zones which you want to modify.

For example, suppose that you wished to define a new porous media hydraulic conductivity to all zones, and thus all elements in the problem. The following set of instructions, inserted in the `.grok` file would accomplish this:

```
use zone type
porous media

clear chosen zones

choose zones all

k isotropic
1.e-5
```

In this case, we are applying the instruction k isotropic to zones of type porous media, although it is equally valid to use it with dual-type zones. However, if you try to use this instruction with fracture- or overland-type zones, a warning will be issued and printed to the `o.eco` file.

The instructions which are valid in specific situations are discussed in the relevant sections of the manual. For example, instructions which can be used for defining porous media properties when simulating saturated flow are described in section 5.6.3.

Another way to define zone properties is through the use of a material properties file, which should be located in the same directory as the `.grok` file. This file contains lists of media-specific instructions which can be used to define properties for one or more named materials. These material properties can then be assigned to the current set of chosen zones. For example, to assign a new hydraulic conductivity through the use of a material properties file, we would issue the following instruction. Read properties

Generic. Reads a complete set of material properties from the file and assigns them to the currently chosen set of zones. It requires the following input:

mat_name The name of the material whose properties are to be read and assigned.

The following set of instructions, inserted in the `.grok` file would accomplish this:

```
use zone type
porous media
```

```
clear chosen zones

choose zones all

read properties
sand
```

The generic instruction read Properties would, in this case, open the porous media material properties file and search for a material named sand. If found, it would read the instructions defining the material and modify the porous media properties for the current set of chosen zones.

The name of the porous media properties file will be `default.mprops`, unless you specify otherwise. To do so, use the following instruction: Properties file

1. **props_file_name Generic**. The name of the material properties file to be searched for zoned property instructions.

This instruction has two benefits. First, it allows you to create your own sets of material properties and give them meaningful file names. Second, it allows you to easily switch between material property sets merely by changing the file name given in the `.grok` file.

The default file names for the other types of media are:

dual default.dprops

fracture default.fprops

overlanddefault.oprops

Any line in the material properties file which is completely blank or which begins with an exclamation point(!) will be ignored. This allows you to include comments whenever required.

Each distinct material in the file is identified by a unique label and may contain instructions which are to be applied to the current zone type. For example, instructions which can be used for defining porous media properties when simulating saturated flow (as described in section 5.6.3) may be included in the file `default.mprops`. Here is an example from the verification problem discussed in section 4.3.1:

```
!----------------------------------------
Porous medium

k isotropic
500.0
```

```
specific storage
0.0

porosity
1.0

longitudinal dispersivity
10.0

transverse horizontal dispersivity
0.1

transverse vertical dispersivity
0.1

tortuosity
0.1

end material
```

A summary of the final data which has been defined for each zone is listed in the `.eco` file. Here is some output generated for the verification problem given above:

```
------------------------------------------------------------------------
POROUS MEDIA PROPERTIES

ZONE:  1
MATERIAL: porous medium
Consists of          100  elements out of          100
Kxx:   500.000
Kyy:   500.000
Kzz:   500.000
Specific storage:   0.00000
Porosity:   1.00000

Longitudinal dispersivity    10.0000
Transverse dispersivity   0.100000
Transverse vertical dispersivity   0.100000
Tortuosity   0.100000
Bulk density    2650.00
Immobile zone porosity    0.00000
Mass transfer coefficient    0.00000
        100  elements of          100  have been assigned properties
```

In this example, note that because flow is saturated no variably-saturated porous media flow properties need to be defined in the material properties file and they . Also, default values for properties (e.g. bulk density, immobile zone porosity etc.) which have not been modified in the `.grok` or material properties file are used.

5.1.5 Pre-processor Considerations

5.1.5.1 Array Dimensioning

When performing task 1, **grok** first checks for the existence of a file `array_sizes.default` in the directory where the `.grok` file is located. If it is not found, the file is automatically created and default array sizes are written which are then used by the preprocessor. Associated with each default are a descriptor and a default value. A portion of the file is shown here:

```
dual: material zones
        20
dual flow bc: flux nodes
     10000
dual flow bc: flux faces
     10000
dual flow bc: flux function panels
        10
dual flow bc: flux zones
        10
dual flow bc: head nodes
     10000
dual flow bc: head function panels
       100
flow: material zones
        20
flow bc: flux nodes
     10000
flow bc: flux faces
     10000
.
.
.
.
wells: 3d elements intersecting
      1000
wells: flux function panels
        10
```

```
wells: injection concentration function panels
       100
end
```

So, for example, the default maximum number of dual continuum material zones is 20. If the problem is defined such that an array exceeds the default maximum (e.g. the number of dual continuum material zones exceeds 20) then **grok** will halt execution and issue an error message of the form:

```
*** ERROR: pre-processor request exceeds default array size ***
dual: material zones
Default value: 20
Increase the default value in file ARRAY_SIZES.DEFAULT
```

Given the descriptor in the error message, you can now edit the file `array_sizes.default` and increase the appropriate value. Note that the file is sorted alphabetically by descriptor. When you run **grok** again, it will read the new default value from the file.

rgm redo this section

1. **node sheets in z for layered grids** A 3D grid is made up of 2D meshes stacked one on top of the other. This is how many 2D meshes you can stack up.
2. **x grid lines for rectangular mesh** The maximum number of grid lines perpendicular to the X-axis for a rectangular mesh.
3. **y grid lines for rectangular mesh** The maximum number of grid lines perpendicular to the Y-axis for a rectangular mesh.
4. **z grid lines for rectangular mesh** The maximum number of grid lines perpendicular to the Z-axis for a rectangular mesh.
5. **material zones** Each element has it's own material number. This is how many different types of porous media (e.g. sand, clay etc) you can have in a given problem.
6. **target times HydroSphere** generates a list of target times based on the boundary conditions and the user defined output times. The user can also define targets using the 'generate target times' instructions. This is how many the model can accomodate.
7. **output times** The number of output times the user specifies. Usually no more than 5 or so due to disk space limitations etc.
8. **specified head nodes** The total number of specified head (first-type) flow nodes allowed in the model.
9. **second-type flow nodes** As in 8 but for specified flux (second-type) flow nodes.

10. **specified head function panels** The specified head boundary condition can vary with time. This is defined by a function with a certain number of panels (e.g. coordinate pairs). The more panels you have the more detailed the definition of the function can be. This is the limit on the number of panels for the current problem.

11. **total line elements for wells** If you define wells they are made up of line elements, each having 2 nodes. This is the total number of line elements for the current problem, and should be greater than or equal to the sum of all line element in all wells.

12. **injection well concentration function panels** As in 10 but the concentration history of an injection well for the transport solution.

13. **nodes in a well** This number should be greater than or equal to the number of nodes in the largest well for the current problem. For example, a well made up of 10 line elements would have 11 nodes.

14. **elements intersecting wells** Well line elements are coincident with 3D block or prism element edges. This number should be greater than or equal to the number of elements in the mesh which have one edge that is also defined as a well line element .

15. **fracture elements intersecting wells** As in 14 but for fracture elements (2d planes) which share a well line element node.

16. **nodes in observation well** As in 13 but for observation wells.

17. **nodes in a seepage face** As in 13 but for a seepage face.

18. **elements connected to a seepage node** As in 14 but for a seepage face.

19. **first-type concentration nodes** As in d:shn but for specified (first-type) concentration nodes.

20. **first-type concentration function panels** As in 10 but for the concentration history at a specified concentration node.

21. **second-type solute flux nodes** As in d:shn but for specified fluid flux nodes.

22. **second-type solute flux function panels** As in 10 but for the concentration history at a specified fluid flux node.

23. **third-type concentration nodes** As in d:shn but for third-type concentration nodes.

24. **third-type concentration function panels** As in 10 but for the concentration history at a third-type concentration node.

25. **output nodes** The user can choose a set of nodes for detailed head, flux and concentration output to be written. This defines how many.

26. **fracture elements** This is how many fracture elements (2D rectangles or triangles) you can have in the current problem.

27. **fracture zones** As in 5 but for fracture materials.
28. **nodes in a tile drain** As in 13 but for tile drains. Tile drains are similar to wells in that they are made up of line elements.
29. **tile elements** As in 14 but for tile drains.
30. **tile drain concentration function panels** As in 10 but for the concentration history at a tile drain.
31. **elements intersecting tiles**As in 14 but for elements intersecting a tile drain.
32. **zero-order source function panels** As in 10 but for the concentration history at a zero-order source node.

grok uses these default values to allocate arrays for the storage of all information which will be generated by instruction found in the `.grok` file. In some cases, these values will not be sufficiently large, and the program may crash when an array bound is exceeded. In such cases, the user should increase the appropriate value in the file `np_array_sizes.default` and re-run **grok**. Note that re-compilation of the code is not necessary, since it uses Fortran 95 ALLOCATE statements to define array sizes at run-time.

In some cases, you may have to decrease a value in this file. For example, if a very large 2D mesh is being used to generate a 3D mesh, the value of 80 for **node sheets in z for layered grids** may cause an array allocation error at run-time, since this value coupled with the size of the 2D mesh results in very large array requirements for storage of the 3D mesh.

Note that **HydroSphere** does not utilize the file `np_array_sizes.default` but instead uses exact array sizes determined and passed by **grok**.

Remember, this process is problem dependent, and each time you run **grok** in a different directory, a fresh file `np_array_sizes.default` will be generated with default values.

5.2 Problem Identification

The first section of the `.grok` file should consist of a description of the problem being defined. As for the rest of the file, blank lines and lines beginning with an exclamation point (!) are ignored.

The description can contain from zero up to as many lines as the user requires to describe the problem. Each line can contain up to 60 characters. The description is printed at the beginning of the listing files for **grok** (`.eco`) and **HydroSphere** (`.lst`).

The user must signal the end of the description using the End title instruction.

End title

This instruction signals the end of the description, and control is then passed back to the

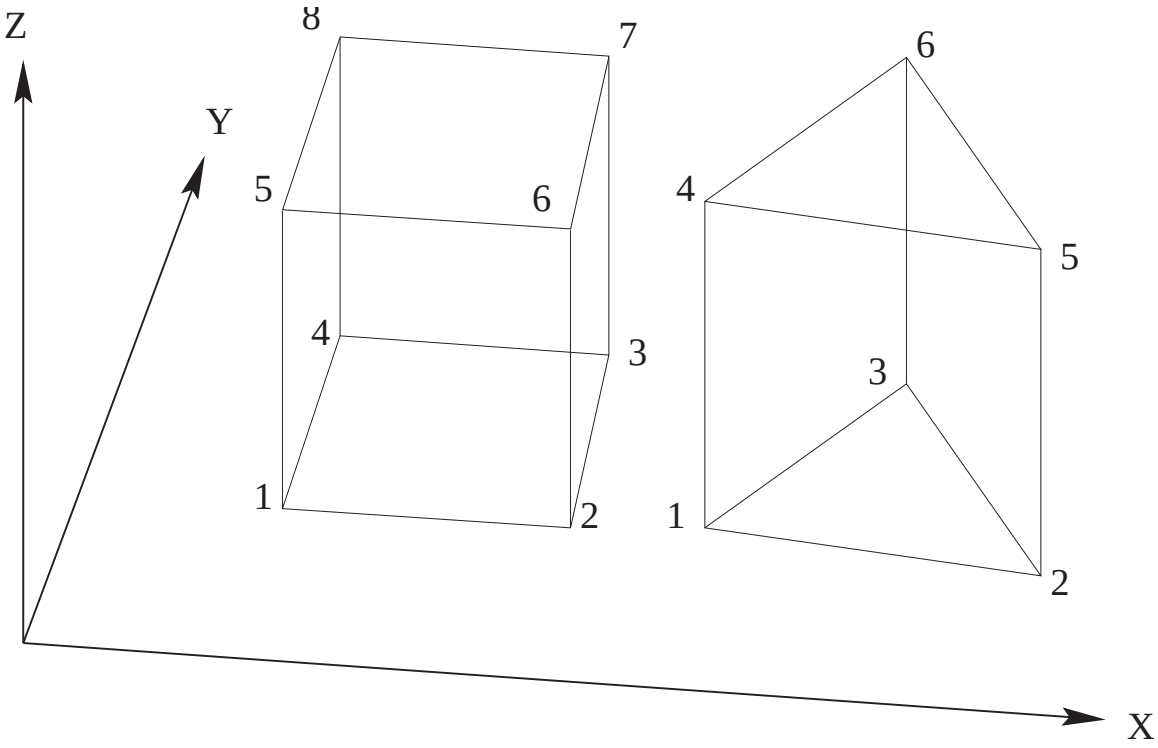


Figure 5.1: Element types and local node numbering conventions.

pre-processor.

5.3 Grid Generation

The next section of the `.grok` file should consist of instruction for grid generation.

Currently, **grok** is capable of generating grids which are composed of either hexahedral blocks or triangular prisms. Figure 5.1 shows the local node numbering conventions for each of these elements and also the positive directions of the X, Y, and Z axes.

There is also an option for subdividing hexahedral block elements into 4-node tetrahedral elements (see section 5.3.5). We will first discuss options for generating simple grids, followed by irregular grids.

5.3.1 Simple Grids

Simple grids can be generated for rectangular domains which are adequate for many problems. They can have uniform or variable element sizes and can be made of hexahedral block or triangular prismatic elements. Each element in the grid is given a default zone number of 1.

Generate uniform blocks

Generates a grid for a rectangular domain made up of uniform blocks. It requires the following input:

1. **xl, nbx** Domain length and number of blocks in the x-direction
2. **yl, nby** Domain length and number of blocks in the y-direction
3. **zl, nbz** Domain length and number of blocks in the z-direction

In this case, the grid is formed by subdividing the domain in the x-direction into nbx blocks, each of length xl/nbx. The domain is subdivided in a similar fashion in the y- and z-directions, using the other input parameters.

Generate uniform prisms

Generates a grid for a rectangular domain made up of uniform prisms. Requires identical input to the routine Generate uniform blocks described above. In this case though, instead of generating block elements, this instruction generates prism elements by subdividing each block into two prism elements.

Generate variable blocks

Generates a grid for a rectangular domain made up of variably-sized blocks. It requires the following input:

1. **nx** Number of nodes in the x-direction
2. **ny** Number of nodes in the y-direction
3. **nz** Number of nodes in the z-direction
4. **xi(i),i=1,nx** X-coordinates of the nx nodes. You can place several values per line but they must be separated by a blank or a comma.
5. **yi(i),i=1,ny** Y-coordinates of the ny nodes.
6. **zi(i),i=1,nz** Z-coordinates of the nz nodes.

It is almost identical to the generate uniform blocks instruction except that instead of entering a domain length in each direction we enter a list of coordinates, which are each used to define the position of a plane of nodes along that axis. The structure **xi(i),i=1,nx** is called an implied do and means that you must supply **nx** values for the array **xi()**. One or more values can be entered per line until the read statement is satisfied, then a new line should be started for the next read statement.

Generate variable prisms

Generates a grid for a rectangular domain made up of variably-sized prisms. Requires

identical input to the routine Generate variable blocks described above. In this case though, instead of generating block elements, this instruction generates prism elements by subdividing each block into two prism elements.

5.3.2 Interactive Block Grids

These instructions can be used to generate a grid made up of variably-sized blocks. The user can grade the mesh as desired in each of the 3 principal directions by repeating the grade x/grade y/grade z instructions. This is particularly useful for cases in which fine meshes are required, for example, near a discrete fracture.

Generate blocks interactive

This instruction signals the start of the mesh grading information to follow. This information should consist of at least one instruction for each of the principal directions.

Grade x

This instruction allows you to generate grid lines (ie. elements) along the x-axis which grade up in size from the start point to the end point. The following input is required.

1. **x1, x2, dxstart, xfac, dxmax** Where x1 is the starting x-coordinate, x2 is the ending x-coordinate, dxstart is the starting element size, xfac is a multiplication factor which determines the rate at which element size increases as we approach X2 and dxmax is the maximum element size allowed.

Grade y

As above but for the y-axis.

Grade z

As above but for the z-axis.

End

This instruction signals the end of the mesh grading information, and control is then passed back to the pre-processor.

The following instructions were used to generate the mesh shown in Figure 5.2.

```
!-------------------
Generate blocks interactive

! refine x near well at 25 and well at 75
grade x
75.0     0.0    .01    1.5    5.
grade x
75.0   100.0    .01    1.5    5.
grade x
```

```
125.0  100.0  .01   1.5    5.
grade x
125.0  200.0   .01   1.5   5.

! refine y near wells at 50
grade y
100.0    0.0   .01   1.5   5.
grade y
100.0  200.0   .01   1.5   5.

! refine z near fracture at 3
grade z
1.0   0.0   .25   1.   0.25
grade z
3.0   1.0   .01   1.3    0.25
grade z
3.0   11.0   .01   1.3   0.25
grade z
11.0   12.0   .25   1.   0.25

end
```

5.3.3 Block Grids with Random Fracture Generation

Rfgen driver

Calls the rfgen subroutine which reads a file with the extension `.rfg` which contains the information for generating the grid and fractures. There is currently no documentation describing the structure of the `.rfg` file so you are referred instead to an example which can be found on the distribution disk in the file `frac3d\rf\flow.grok`, which reads the file `frac3d\rf\flow.rfg`.

5.3.4 Irregular grids

Irregular grids can be generated by supplying node coordinates, element incidences and element zones for a 2D slice which is composed of triangular or quadrilateral elements. Currently, triangles and quadrilaterals can not be mixed in the same slice. These slices can then be replicated to form a 3D mesh composed of 6-node prisms (from triangles) or 8-node hexahedra (from quadrilaterals).

5.3.4.1 Obtaining the 2D slice

The following instructions can be used to obtain 2D slice data.

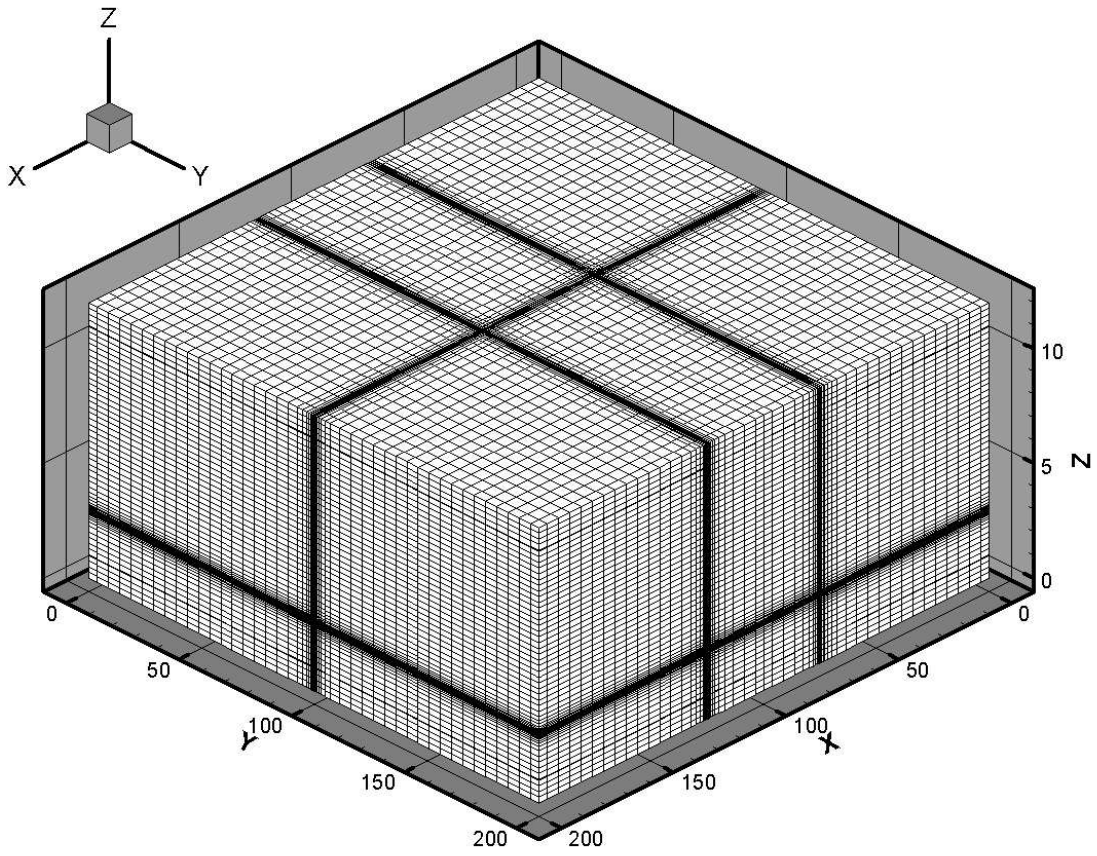


Figure 5.2: Example grid which was created using Generate blocks interactive instructions.

Generate uniform rectangles

Generates a 2D grid for a rectangular domain made up of uniform rectangles. Each rectangular element will be assigned a default zone number of 1. It requires the following input:

1. **xl, nbx** Length and number of rectangles in the x-direction
2. **yl, nby** Length and number of rectangles in the y-direction

It is identical to the generate uniform blocks instruction except that we drop the z-axis parameters.

Generate variable rectangles

Generates a 2D grid for a rectangular domain made up of variably-sized rectangles. Each rectangular element will be assigned a zone number of 1. It requires the following input:

1. **nx** Number of nodes in the x-direction
2. **ny** Number of nodes in the y-direction
3. **xi(i),i=1,nx** X-coordinates of the nx nodes. You can place several values per line but they must be separated by a blank or a comma.
4. **yi(i),i=1,ny** Y-coordinates of the ny nodes.

It is almost identical to the generate variable blocks instruction except that we drop the z-axis parameters.

Read slice

Reads a file which contains data defining a 2D slice. The format of this file is compatible with that produced by the Groundwater Modeling System (GMS) software which was developed at Brigham Young University for the US Department of Defense and is described in detail in Section A.1. It requires the following input:

1. **gmsfile** The name of the file which contains the 2D slice data. This is a string variable.

Read grid builder slice

Reads the files which contain data defining a 2D slice composed of 3-node triangular elements. These files are compatible with output which was generated by the GRID BUILDER program and which is described in detail in Section B.1. It requires the following input:

1. **prefix** The prefix of the GRID BUILDER files which contain the node coordinates, element incidences and element zone numbers for the 2D triangular mesh. This is a string variable.

5.3.4.2 Generating the 3D mesh

Once you have read a 2D slice, if you do not instruct the pre-processor to generate a layered system with one of the instructions outlined below, it will automatically default to a unit thickness 3D grid. It does this by duplicating the 2D slice obtained above and constructing the appropriate 6-node prism or 8-node hexahedral element incidences and assigning a unit element length. The element zone numbers for the slice will be used to assign default zones for each element. Such a grid can be useful, for example, when simulating simple 2D cross-sectional problems.

Generate layers from slice

Uses the currently defined 2D slice to generate a 3D grid with characteristics other than a single layer of unit thickness. It requires the following input:

1. **zone_by_layer** This is a logical variable which controls how zone numbers are assigned to each element. If true, the layer numbers (as defined below) will override the zone numbers defined for the slice. If false, they are assigned based on the zone numbers given for the slice.
2. **constant_base** This is a logical variable which controls whether the base elevation is constant (ie. flat) or read from a file (ie. sloping, undulating etc.). In this case the file is assumed to have been created by GMS. If true read the following:
   (a) **base_elev** The elevation of the base of the grid.

   If false read the following:
   (a) **basefile** The name of the data file containing the base elevation values for each node in the 2D grid. This is a string variable. The file should be formatted as outlined in Section A.2.
3. **nlayers** This value corresponds to the number of layers you want in the system. It is also used to assign a default zone to each element in the layer if the variable zone_by_layer (see above) is true. For each of the layers read the following:
   (a) **layer_name** The name of the current layer. This is a string variable. This name is only used as a lable in subsequent output and is not used to assign material properties.
   (b) **nsublayers** The number of sublayers which make up this layer.
   (c) **constant_top** This is a logical variable which controls whether the top elevation of the current layer is constant (ie. flat) or read from a file (ie. sloping, undulating etc.). If true, read the following:
      i. **top_elev** The elevation of the top of the layer.

   If false read the following:
      i. **topfile** The name of the ascii scalar data file containing the top elevation values for this layer for each node in the 2D grid. This is a string variable. The file should be formatted as outlined in Section A.2.

Generate layers from grid builder slice

Generates a 3D prism grid with characteristics other than a single layer of unit thickness. This instruction requires the same input as Generate layers from slice. The only difference is that GRID BUILDER format is expected when reading scalar data files containing the base elevation for the grid or the top elevation for a layer. The scalar data files should be formatted as outlined in Section B.2.

Generate layers from slice with functions

5.3.5 Tetrahedral Element Grids

Tetrahedra

Subdivides block elements into tetrahedra (4-node) elements. If the grid does not contain block elements the pre-processor will stop and you will be issued a warning.

Block elements must be orthogonal or nearly so in order to avoid introducing numerical error, while tetrahedral elements can handle irregular geometries. The subdivision into tetrahedra is carried out by **HydroSphere** during problem solution and is transparent to the user. Faces which can be designated as fracture elements are restricted to the original block elements. This is merely a matter of convenience and future versions may allow tetrahedral element faces to be designated as fractures.

5.3.6 Axisymmetric flow

Axisymmetric flow

This instruction is used for simulating radial flow to a well. *It should only be applied to a a vertical cross-section, of unit thickness in the y-direction.* The x coordinate is taken as the radial distance.

One should define a vertical cross-section of unit thickness in the y-direction (with 2 nodes in that direction), and locate a pumping/injection well at the origin (x=0.0).

5.3.7 Reading an existing 3D grid

In some cases, the grid generation step can be very time consuming. If this is so, then the following instruction can be used to read a grid which was generated in a previous run:

Read 3D grid

The grid whose prefix matches that of the `.grok` file will be read in.

The following example shows how to set up the grid generation section of the `.grok` file:

```
!-----------------------------------------------------
```

```
!Generate the grid for first run
!skip on

read grid builder slice
g4

generate layers from grid builder slice
.true.
.true.
0.0
1
Layer 1
1
.true.
1.0

write faces and segments
!skip off
!-------------------------------------------------------
! Read previously defined grid for subsequent runs
skip on

read 3D grid

skip off
!-------------------------------------------------------
done grid definition
```

In the first run, one can read the slice and generate the layered grid. It is important that the instruction write faces and segments be included if you are using any of the 'choose face ...' or 'choose segment ...' options later in the `.grok` file.

On subsequent runs, one can skip over the grid generation commands and use the read 3D grid instruction instead.

5.3.8 Ending grid Generation

Done grid definition

Signals the end of the user-controlled portion of the grid definition section of the input data file. At this stage, the pre-processor will automatically perform grid modifications if appropriate. For example, if you read in 2D slice data but did not specify layer information using the generate layers from slice instruction, the pre-processor would generate a default 3D system by duplicating the 2D slice to form a single layer of unit-thickness elements.

5.4 Grid output instructions

The following instructions can be useful in checking the results of the grid generation section.

Echo coordinates

Causes node coordinates to be written to the `o.eco` file.

Echo incidences

Causes element incidences to be written to the `o.eco` file.

Echo element area numbers

Causes element area numbers (as read in section 5.3.4 for a 2D slice generated by GRID BUILDER) to be written to the `o.eco` file.

Set kpmsh

Assigns a value to the variable KPMSH, which controls the output of mesh information to the `o.lst` file. It requires the following input:

1. **kpmsh** For KPMSH equal to 2, the nodal coordinates and elements incidences, for the 3D elements and for the 2D elements if there are fractures, are printed. For KPMSH equal to 1, only the nodal coordinates are printed. For KPMSH equal to 0, no grid information is printed. The default value is 0.

These instructions can be used to export 2D and 3D meshes and other data generated by **grok** to GMS.

Mesh to gms

Writes the 3D mesh (blocks or prisms) information to a file in GMS readable format. It requires the following input:

1. **gmsfile** The name of the file which will contain the 3D mesh information to be read by GMS.

2D mesh to gms

Writes the 2D mesh (quadrilaterals or triangles) information to a file in GMS readable format. It requires the following input:

1. **gmsfile** The name of the file which will contain the 2D mesh information to be read by GMS.

Mesh to tecplot

Writes the 3D mesh (blocks or prisms) information to a file in TECPLOT readable format. It requires the following input:

1. **tecfile** The name of the file which will contain the 3D mesh information to be read by TECPLOT.

Write faces and segments

Whenever **HydroSphere** generates a 3D mesh, it makes lists of the nodes which comprise each unique face and line segment. This information is used by certain instructions which choose faces or segments. You can use this instrucion to write the information to a file which will receive the name `prefixo.fac`. These files can become quite large, so the default is not to save them. See also Section 5.3.7.

The following instructions should be placed near the end of the `.grok` file, after any instructions which are used to generate wells or fractures.

Wells to gms

Writes the 1D line element information which has been generated to define wells and/or tile drains in GMS readable borehole file format. It requires the following input:

1. **gmsfile** The name of the file which contains the 1D line information which will be read by GMS.

Fractures to gms

Writes the 2D element (quadrilaterals or triangles) information which has been used to define fractures in GMS readable 2D mesh file format. It requires the following input:

1. **gmsfile** The name of the file which contains the 2D fracture element information which will be read by GMS.

5.5 Selecting mesh components

In order to assign boundary conditions, material properties etc. we need to be able to choose subsets of the grid. The method of choice must be flexible and easy to use as well as being able to handle complex input requirements.

The following is a list of grid components, ranked in order of increasing complexity:

1. nodes – used to assign initial heads and first-type boundary conditions
2. segments – used to represent wells, tile drains or obsevation wells
3. faces (triangles or rectangles) – used to represent fractures or high-conductivity planes (as 2-D triangular or rectangular elements) and to assign second and third-type boundary conditions to these as well as 3D prism or block elements.

4. elements(blocks, prisms or tetrahedra) – sometimes used to assign hydraulic conductivities or distribution coefficients

5. zones – generally used to assign material properties such as hydraulic conductivity. Elements are grouped into zones by assigning them the same ID number.

We will assign to all members of a grid component an attribute called chosen, which can be toggled on or off by the user. If the attribute is chosen for certain members of a component, then subsequent instructions issued by the user will affect those members only. For example, the following section of a hypothetical `.grok` file would initially turn off all chosen nodes (ie. instruction clear chosen nodes which requires no further input), then turn on only those nodes satisfying the requirement that they are within 1.e-5 distance units of the plane defined by the equation $x = 0.0$ (ie. instruction choose nodes x plane followed by two lines of input).

```
clear chosen nodes
choose nodes x plane
0.0                  X coordinate of plane
1.e-5                distance criteria
```

Once these nodes were chosen, we could set the property of interest by issuing another instruction like:

```
specified head
1
0.0 10.0
```

In this case we are assigning a constant head of 10.0 to all nodes on the plane at time 0.0, which will apply for the duration of the simulation. Note that the instruction specified head acts on nodes by definition. It is up to the user to be aware of which components each group of instructions acts on.

The effect of issuing two such instructions in succession is cumulative. For example, the following input would choose nodes which were within 1.e-5 distance units of the planes at $x = 0.0$ and $x = 10.0$.

```
clear chosen nodes
choose nodes x plane
0.0                  X coordinate of plane
1.e-5                distance criteria
choose nodes x plane
10.0                 X coordinate of plane
1.e-5                distance criteria
```

The following sections introduce all the instructions which are available for choosing subsets of the various grid components.

5.5.1 Selecting nodes

We can use the following instructions to alter the set of chosen nodes.

Clear chosen nodes

Returns the set to the default state, in which no nodes are chosen. This is recommended if you are unsure of which nodes are chosen due to previously issued instructions.

Choose nodes all

Selects all nodes. This is useful if you wish to assign a property to all nodes in the grid. For example, you could issue this instruction and then assign a uniform initial head for the problem.

Choose node

Select the node closest to the given coordinate. It requires the following input:

1. **x1, y1, z1** The coordinates of the node to be chosen.

Choose node number

Select the specified node. You should use this instruction with caution since node numbering will change if the grid structure changes. It requires the following input:

1. **i** The number of the node to be chosen.

Choose nodes x plane

Select nodes which are within the distance PTOL of the plane which is defined by the equation $X = X1$. This command is particularly useful when assigning boundary conditions to a specific face of a rectangular domain. It requires the following input:

1. **x1** The X-coordinate of the plane.
2. **ptol** Distance from the plane. If a node is within this distance of the plane, it will be chosen.

Choose nodes y plane

As above but for the y-plane.

Choose nodes z plane

As above but for the z-plane.

Choose nodes 3pt plane

Select nodes which are within the distance PTOL of the plane which is defined by 3 points. This allows you to choose planes of nodes with an arbitrary orientation. It requires the following input:

1. **x1, y1, z1** The coordinates of the first point.

2. **x2, y2, z2** The coordinates of the second point.
3. **x3, y3, z3** The coordinates of the third point.
4. **ptol** Distance from the plane. If a node is within this distance of the plane, it will be chosen.

Choose nodes block

Select nodes which are within the rectangular block which is defined by 3 ranges. It requires the following input:

1. **x1, x2** The X-range of the block.
2. **y1, y2** The Y-range of the block.
3. **z1, z2** The Z-range of the block.

Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line or point respectively.

Choose nodes top

Selects all nodes in the top slice of the domain.

Choose nodes bottom

Selects all nodes in the bottom slice of the domain.

Choose nodes top gb

Reads a GRID BUILDER chosen nodes file and selects nodes which are in the top slice and have been chosen by GRID BUILDER. It requires the following input:

1. **fname** The name of the GRID BUILDER file which contains the chosen node information. This is a string variable.

Choose nodes gb

Reads a GRID BUILDER chosen nodes file and selects nodes which are between two specified sheets (inclusive) and have been chosen by GRID BUILDER. It requires the following input:

1. **fname** The name of the GRID BUILDER file which contains the chosen node information. This is a string variable.
2. **nsheet_bot,nsheet_top** The bottom and top sheet numbers which are used to limit the selection process. Nodes which are between the bottom and top sheet (inclusive) and have been chosen by GRID BUILDER are selected.

5.5.2 Selecting segments

We can use the following instructions to alter the set of chosen segments.

Clear chosen segments

Returns the set to the default state, in which no segments are chosen. This is recommended if you are unsure of which segments are chosen due to previously issued instructions.

Choose segments all

Selects all segments. This is useful if you wish to assign a property to all segments in the grid.

Choose segments polyline

Select segments which fall on or close to a polyline.

1. **npts** The number of points defining the polyline. Read the following npts times:
   (a) **x1, y1, z1** The coordinates of a point on the polyline.

The points should be given in order from one end of the polyline to the other. The routine finds the nodes closest to the coordinates of the points given and then finds the segments forming the shortest path between the nodes.

Choose segments line

As above but for a single line with 2 endpoints.

1. **x1, y1, z1** The coordinates of the first endpoint.
2. **x2, y2, z2** The coordinates of the second endpoint.

5.5.3 Selecting faces

We can use the following instructions to alter the set of chosen faces.

Clear chosen faces

Returns the set to the default state, in which no faces are chosen. This is recommended if you are unsure of which faces are chosen due to previously issued instructions.

Choose faces all

Selects all faces.

Choose faces x plane

Selects faces which are within the distance PTOL of the plane which is defined by the equation $X = X1$. This command is particularly useful when assigning boundary conditions to a specific face of a rectangular domain. It requires the following input:

1. **x1** The X-coordinate of the plane.
2. **ptol** Distance from the plane. If a face is within this distance of the plane, it will be chosen.

Choose faces y plane

As above but for the y-plane.

Choose faces z plane

As above but for the z-plane.

Choose faces 3pt plane

Selects faces which are within the distance PTOL of the plane which is defined by 3 points. This allows you to choose planes of faces with an arbitrary orientation. This is particularly useful for setting up a set of sloping fractures. It requires the following input:

1. **x1, y1, z1** The coordinates of the first point.
2. **x2, y2, z2** The coordinates of the second point.
3. **x3, y3, z3** The coordinates of the third point.
4. **ptol** Distance from the plane. If a face is within this distance of the plane, it will be chosen.

Choose faces block

Selects faces which are within the rectangular block which is defined by 3 ranges. It requires the following input:

1. **x1, x2** The X-range of the block.
2. **y1, y2** The Y-range of the block.
3. **z1, z2** The Z-range of the block.

Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line or point respectively.

Choose faces block by layer

Selects faces whose centroids are within the rectangular block which is defined by 3 coordinate ranges, and which lie within the element layers defined by nlaybot and nlaytop. These layer numbers do not correspond to those given during grid generation but are simply defined by numbering each sheet of elements from 1 (bottom) to nz-1 (top) where nz is the number of sheets of nodes (2d meshes) making up the grid.

This instruction is designed for grids that are regular in x and y, but which have a variable z for a given element layer, and can be used if the top and bottom elevations of a 3D element layer vary spatially.

It requires the following input:

1. **x1, x2** The X-range of the block.
2. **y1, y2** The Y-range of the block.
3. **z1, z2** The Z-range of the block.
4. **nlaybot, nlaytop** The bottom and top layer numbers. Only faces lying within this range (inclusive) may be chosen. The element layer number increases in the z direction.

Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line or point respectively.

Choose faces sheet

Selects faces which are between two specified sheets (inclusive). It requires the following input:

1. **nsheet_bot,nsheet_top** The bottom and top sheet numbers which are used to limit the selection process. Faces which are between the bottom and top sheet (inclusive) and have been chosen by GRID BUILDER are selected.

Note that only faces lying in the sheet are chosen. Faces which are oriented perpendicular to the sheet will not be chosen.

Choose faces top

Selects all faces in the top slice of the domain.

Choose faces bottom

Selects all faces in the bottom slice of the domain.

Choose faces top gb

Reads a GRID BUILDER chosen elements file and selects faces which are in the top sheet and have been chosen by GRID BUILDER. It requires the following input:

1. **fname** The name of the GRID BUILDER file which contains the chosen element information. This is a string variable.

Note that only faces lying in the sheet are chosen. Faces which are oriented perpendicular to the sheet will not be chosen.

Choose faces gb

Reads a GRID BUILDER chosen elements file and selects faces which are between two

specified sheets (inclusive) and have been chosen by GRID BUILDER. It requires the following input:

1. **fname** The name of the GRID BUILDER file which contains the chosen element information. This is a string variable.
2. **nsheet_bot,nsheet_top** The bottom and top sheet numbers which are used to limit the selection process. Faces which are between the bottom and top sheet (inclusive) and have been chosen by GRID BUILDER are selected.

Note that only faces lying in the sheet are chosen. Faces which are oriented perpendicular to the sheet will not be chosen.

Choose horizontal faces on layer

Selects horizontal faces which are which are on the layer of nodes numbered nlayer and within the rectangular block which is defined by xy ranges. It requires the following input:

1. **nlayer**
2. **x1,x2** The X-range of the block.
3. **y1,y2** The Y-range of the block.

This instruction can be used to select horizontal faces (to make fractures) when the elevation of a given layer of nodes is irregular.

write chosen faces

This instruction writes the set of currently chosen face numbers to a file. Setting up complex fracture networks with combinations of 'choose face...' instructions can be very time consuming in **grok** and this step does not need to be repeated as long as the grid structure remains the same.

1. **fname** The name of the file to which the chosen face information will be written. This is a string variable.

read chosen faces

This instruction reads a set of face numbers from the file and sets those faces as chosen.

If you want only those faces read from the file to be chosen then make sure to issue the instruction 'clear chosen faces' before you use 'read chosen faces'. If not, the results will be merged with the currently chosen set of faces. This could be useful if you want to apply a certain set of fracture material properties to more than one group of faces at a time.

1. **fname** The name of the file from which the chosen face information will be read. This is a string variable.

PLANE ID	LOCAL NODES
1	1-2-7-8
2	4-3-6-5
3	2-3-8-5
4	1-4-7-6
5	1-3-7-5
6	2-4-8-6

5.5.4 Selecting inclined faces

These instructions only work for rectangular meshes with the standard element numbering scheme. They are intended to be used in conjunction with the Make inclined fractures instruction.

For each block element, there are 6 potential inclined faces which may be selected. These are given ID numbers according to the following convention:

clear chosen inclined faces

Returns the set to the default state, in which no inclined faces are chosen. This is recommended if you are unsure of which inclined faces are chosen due to previously issued instructions.

choose faces 3pt inclined plane

Selects inclined faces which are within the distance PTOL of the plane which is defined by 3 points, have the appropriate plane ID and are within a block defined by xyz coordinate ranges. It requires the following input:

1. **nplane** The plane ID number, as defined above.
2. **x1, y1, z1** The coordinates of the first point on the plane.
3. **x2, y2, z2** The coordinates of the second point on the plane.
4. **x3, y3, z3** The coordinates of the third point on the plane.
5. **ptol** Distance from the plane. If the centroid of the face is within this distance of the plane, it will be chosen.
6. **xmin, xmax** Only faces whose centroid x-coordinate falls within this range may be chosen.
7. **ymin, ymax** Only faces whose centroid y-coordinate falls within this range may be chosen.
8. **zmin, zmax** Only faces whose centroid z-coordinate falls within this range may be chosen.

5.5.5 Selecting elements

We can use the following instructions to alter the set of chosen elements.

Clear chosen elements

Returns the set to the default state, in which no elements are chosen. This is recommended if you are unsure of which elements are chosen due to previously issued instructions.

Choose elements all

Selects all elements.

Choose elements x plane

Selects elements which are within the distance PTOL of the plane which is defined by the equation $X = X1$. This command is particularly useful when assigning boundary conditions to a specific element of a rectangular domain. It requires the following input:

1. **x1** The X-coordinate of the plane.
2. **ptol** Distance from the plane. If a element is within this distance of the plane, it will be chosen.

Choose elements y plane

As above but for the y-plane.

Choose elements z plane

As above but for the z-plane.

Choose elements 3pt plane

Selects elements which are within the distance PTOL of the plane which is defined by 3 points. This allows you to choose planes of elements with an arbitrary orientation. This is particularly useful for setting up a set of sloping fractures. It requires the following input:

1. **x1, y1, z1** The coordinates of the first point.
2. **x2, y2, z2** The coordinates of the second point.
3. **x3, y3, z3** The coordinates of the third point.
4. **ptol** Distance from the plane. If a element is within this distance of the plane, it will be chosen.

Choose elements block

Selects elements which are within the rectangular block which is defined by 3 coordinate ranges. It requires the following input:

1. **x1, x2** The X-range of the block.
2. **y1, y2** The Y-range of the block.
3. **z1, z2** The Z-range of the block.

Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line or point respectively.

Choose elements block by layer

Selects elements which are within the rectangular block which is defined by 3 coordinate ranges, and which lie within the layers defined by nlaybot and nlaytop. These layer numbers do not correspond to those given during grid generation but are simply defined by numbering each sheet of elements from 1 (bottom) to nz-1 (top) where nz is the number of sheets of nodes (2d meshes) making up the grid.

This instruction is designed for grids that are regular in x and y, but which have a variable z for a given element layer, and can be used if the top and bottom elevations of a 3D element layer vary spatially.

It requires the following input:

1. **x1, x2** The X-range of the block.
2. **y1, y2** The Y-range of the block.
3. **z1, z2** The Z-range of the block.
4. **nlaybot, nlaytop** The bottom and top layer numbers. Only elements lying within this range (inclusive) may be chosen. The element layer number increases in the z direction.

Note that the values given for one, two or all of the ranges can be identical and in that case, the block will collapse to a plane, line or point respectively.

Choose elements list

Selects elements which are listed in a user defined file. It requires the following input:

1. **fname** The name of the file which contains the list of element numbers. This is a string variable. The first line of the file is the number of elements in the list, followed by the list of element numbers, one entry per line.

Choose elements xyz list

This instruction only works for rectangular meshes with the standard element numbering scheme. Selects elements using coordinates which are listed in a user defined file. It requires the following input:

1. **fname** The name of the file which contains the list of xyz coordinate triplets, one entry per line. For each coordinate triple, determines which element it falls in and then selects that element.

Choose elements gb

Reads a GRID BUILDER chosen elements file and selects elements which are between two specified sheets and have been chosen by GRID BUILDER. It requires the following input:

1. **fname** The name of the GRID BUILDER file which contains the chosen element information. This is a string variable.
2. **nsheet_bot,nsheet_top** The bottom and top sheet numbers which are used to limit the selection process. Elements which are between the bottom and top sheet and have been chosen by GRID BUILDER are selected.

5.5.6 Selecting zones

We can use the following instructions to alter the set of chosen zones.

Clear chosen zones

Returns the set to the default state, in which no zones are chosen. This is recommended if you are unsure of which zones are chosen due to previously issued instructions.

Choose zones all

Selects all zones. This is useful if you wish to assign a property to all zones in the grid.

Choose zone number

Selects a specific zone. It requires the following input:

1. **num_zone** The number of the zone to be chosen.

5.6 Subsurface Flow

5.6.1 Flow Input/Output Considerations

The following instructions affect the I/O for the flow solution.

Pressure head input

Causes all heads which are input to be treated as pressure heads instead of hydraulic heads.

Pressure head output

Causes all heads which are output to the ASCII file to be treated as pressure heads instead of hydraulic heads.

Echo to output

Causes heads, saturations,concentrations and velocities to be written to the `.lst` file as well as to the binary output files.

No fluid mass balance

This instruction suppresses the calculation of fluid mass balance information which is, by default, computed at each time step.

Set kphead

Assigns a value to the variable KPHEAD, which controls the output of nodal heads in binary format to a file which will automatically be assigned the suffix `o.h01`. It requires the following input:

1. **kphead** Set KPHEAD equal to 0 if no output of heads is desired. If steady-state flow is simulated then set KPHEAD greater than 0 for output of heads. If transient flow is simulated then set KPHEAD equal to 1 to output heads at the output times specified in instruction output times. The default value is 1.

Set kpsat

Assigns a value to the variable KPSAT, which controls the output of elemental saturations in binary format to a file which will automatically be assigned the suffix `o.sat`. A similar file containing the nodal saturations is also written, and assigned the suffix `o.sw`. It requires the following input:

1. **kpsat** Set KPSAT equal to 0 if no output of saturations is desired. If steady-state flow is simulated then set KPSAT greater than 0 for output of saturations. If transient flow is simulated then set KPSAT equal to 1 to output saturations at the output times specified in instruction output times. The default value is 1.

Set kpvel

Assigns a value to the variable KPVEL, which controls the output of elemental velocities in binary format to a file which will automatically be assigned the suffix `o.vel`. It requires the following input:

1. **kpvel** Set KPVEL equal to 0 if no output of velocities is desired. If steady-state flow is simulated then set KPVEL greater than 0 for output of velocities. If transient flow is simulated then set KPVEL equal to 1 to output velocities at the output times specified in instruction output times. The default value is 1.

Set kwrith

Assigns a value to the variable KWRITH, which controls whether head values for the last time step are output in BINARY form. This enables the use of the last time step heads as initial conditions for a subsequent simulation, for example, if one wants to carry on the simulation further in time without restarting from time zero. The file will automatically be assigned the suffix `o.hen`.

It requires the following input:

1. **kwrith** Set KWRITH equal to 1 to write the head values for the last time step to a file. Set KWRITH equal to 0 if no output of final heads is desired. The default value is 0.

Set kpmasb

Assigns a value to the variable KPMASB, which controls the output of fluid mass balance information in binary format to a file which will automatically be assigned the suffix `o.bal`. It requires the following input:

1. **kpmasb** Set KPMASB equal to 1 to write the fluid mass balance information to a file. Set KPMASB equal to 0 if no output of fluid mass balance information is desired. The default value is 1.

Flux output nodes

This instruction allows you to generate detailed mass flux information at specific nodes for each timestep. It requires the following input:

1. **new_noutfc** The number of new output nodes desired. Read the following new_noutfc times:
   (a) **ioutfc(i)** Flux output node number. These values should be entered one per line.

Specifying flux output nodes causes **HydroSphere** to create a file called `prefixo.flu`. For each timestep in the flow solution, one line per flux output node will be written to the file. Each line contains the node number, time, fluid flux and nodal coordinates. Such output can be imported into an editor (e.g. Microsoft Excel) and sorted by column to facilitate, for example, the creation of a plot of fluid flux versus time at a node.

For an example which uses flux output nodes, see verification problem in Section 4.3.1.

5.6.2 Physical constants

Default values are assigned for the gravitational acceleration and fluid properties which correspond to standard values in the kilogram-metre-second system. These parameters are used when defining the properties of fractures, open wells and tile drains.

The following default values will be used for the physical constants and correspond to typical values in the kilogram-metre-second system:

- Gravitational acceleration $g = 9.80665$ m/s^2
- Fluid density $\rho = 1000.0$ Kg/m^3
- Fluid viscosity $\mu = 1.124 \times 10^{-3}$ Kg/(m·s)
- Fluid compressibility $\beta = 4.4 \times 10^{-10}$ (m·s^2)/Kg
- Fluid surface tension $\gamma = 0.07183$ Kg/s^2

If you are using different units or you want to change the default values you can do so using the following instructions.

units: kilogram-metre-hour

Converts the default values given above into the kilogram-metre-hour system. This instruction also converts the porous media, dual continuum, fractured media and overland flow default properties which are defined in the code. NOTE: It does not convert properties specified in any `.grok.`,`.mprops`, etc. file. Similar instructions exist for converting to the following systems:

- kilogram-metre-day
- kilogram-metre-year
- kilogram-centimetre-second
- kilogram-centimetre-hour
- kilogram-centimetre-day
- kilogram-centimetre-year

You can change the default values of the physical constants using the following instructions. If you change the default units from the kilogram-metre-second system make sure the values given here are in the new system.

Gravitational acceleration

1. **grav** Gravitational acceleration constant [L/T^2]

Fluid density

1. **rho** Fluid density [M/L^3]

Parameter	Value	
Name	Sand	
Hydraulic conductivity terms:		
K_{xx}	7.438d-5	m/s
K_{yy}	7.438d-5	m/s
K_{zz}	7.438d-5	m/s
K_{xy}	0.0	m/s
K_{xz}	0.0	m/s
K_{yz}	0.0	m/s
Specific storage S_s	1.0d-4	1/m
Porosity	0.375	
Unsaturated flow relation type	Pseudo-soil	

Table 5.1: Default values for porous media saturated flow properties

Fluid viscosity

1. **visc** Fluid viscosity [M/(LT)]

Fluid compressibility

1. **wcomp** Fluid compressibility [$(LT^2)/M$]

Zero fluid compressibility

Assigns a value of zero for fluid compressibility (i.e. incompressible)

Fluid surface tension

1. **tensn** Fluid surface tension [M/T^2]

5.6.3 Saturated Porous Media Properties

HydroSphere is designed to perform the flow simulation in saturated mode unless instructed otherwise, and unless you modify the default values, all zones (and elements) in the domain will be assigned the default porous media properties which are listed in Table 5.1. These values are representative of a sand.

5.6.3.1 Modifying the default material property distribution

You can use the methods and instructions outlined in section 5.1.4 to modify the default distribution of saturated porous media properties.

The following instructions can be used in the `.grok` file or a porous media material properties file following instructions to define material parameter values for the current set of chosen zones. Setting the chosen zones is described in section 5.5.6.

k isotropic

Assigns an isotropic hydraulic conductivity (ie. $K_{xx} = K_{yy} = K_{zz}$) to the currently chosen set of zones. It requires the following input:

1. **kval** Hydraulic conductivity [L/T].

k anisotropic

Assign an anisotropic hydraulic conductivities to the currently chosen set of zones. It requires the following input:

1. **kvalx, kvaly, kvalz** Hydraulic conductivities [L/T] in the x-, y- and z-directions respectively.

k tensor

This option currently only works for saturated flow conditions. Assign hydraulic conductivities which include the off-diagonal terms to the currently chosen set of zones. It requires the following input:

1. **valx, valy, valz** Main-diagonal terms of the hydraulic conductivity tensor K_{xx}, K_{yy} and K_{zz} [L/T].
2. **valxy, valxz, valyz** Off-diagonal terms of the hydraulic conductivity tensor K_{xy}, K_{xz} and K_{yz} [L/T].

The currently chosen zone(s) will be assigned the given values for the K tensor. Note that a finite element scheme is automatically assumed to consider the off-diagonal terms of the tensor.

specific storage

Assigns a specific storage to the currently chosen set of zones. It requires the following input:

1. **val** Specific storage [1/L]

Porosity

Assigns a porosity to the currently chosen set of zones. It requires the following input:

1. **val** Porosity [L^3/L^3]

The following instructions are included for various purposes and may only be used in the `.grok` file.

Element K isotropic

Assigns isotropic hydraulic conductivities (ie. $K_{xx} = K_{yy} = K_{zz}$) to the currently chosen set of elements. It requires the following input:

1. **kval** Hydraulic conductivity [L/T].

Element K anisotropic

Assigns anisotropic hydraulic conductivities to the currently chosen set of elements. It requires the following input:

1. **kvalx, kvaly, kvalz** Hydraulic conductivities [L/T] in the x-, y- and z-directions respectively.

Read elemental k from file

Reads a variable K field from a user supplied file. It requires the following input:

1. **variable_k_file_name** The name of file which contains the variable K data.

In the file specified, the K values for each element comprising the grid is given, line by line, with the following format:

element_number, kxx, kyy, kzz.

For example, if there a 4 elements, with $K_{xx} = K_{yy} = 5$ m/day and $K_{zz} = 2$ m/day, the file would contain:

```
1   5.0   5.0   2.0
2   5.0   5.0   2.0
3   5.0   5.0   2.0
4   5.0   5.0   2.0
```

Write element k

Writes a file of element hydraulic conductivity values. This was included for use in Advanced Visual Systems (AVS) visualization software, to show regions of different hydraulic conductivity, but could otherwise be generally useful. It requires the following input:

1. **filenm** The name of file to which the hydraulic conductivity information will be written. The following FORTRAN code segment shows how the file is opened and the data written:

```
open(8,file=filenm,status='unknown',form='unformatted')
write(8) (kxx(iprop(i)),i=1,ne)
```

where kxx is the hydraulic conductivity (real*8), iprop(i) is the element zone id number and ne is the number of elements in the mesh.

Get element k

For the group of currently chosen elements, this instruction computes the average hydraulic conductivity and writes it to the `.lst` file. This is useful for example, when a random conductivity field has been generated and the user would like to know the average hydraulic conductivity of a region of the domain.

AECL properties

This instruction can be used to read an AECL Motif grid and map element material numbers onto the existing **HydroSphere** mesh. The mapping is performed based on the proximity of **HydroSphere** and AECL Motif element centroids.

1. **aecl_nd_file** The name of the file which contains the nodal coordinates for the AECL Motif mesh.
2. **aecl_ne_file** The name of the file which contains the element incidences and material property numbers for the AECL Motif mesh.

The file `default.mprops` can be set up so that the material properties are in an order which corresponds to the AECL material numbers.

5.6.3.2 Random Hydraulic Conductivity Fields

You can treat hydraulic conductivity and distribution coefficient as an elemental property by setting `MAXKZN` and `MAXKDZN` to be equal to `MAXNE`, the maximum number of elements. You can then use the following instructions to define element hydraulic conductivities for the current set of chosen elements. If these array dimensions are not set properly, you will be warned and **grok** will stop. Setting the chosen elements is described in section 5.5.5.

Random K field

Reads a random K field which was generated by the program FGEN [*Robin et al.*, 1993]. It requires the following input:

1. **fgenfile** The name of file which contains the random hydraulic conductivity information. The FGEN code generates two cross-correlated 3D random fields having user-specified geostatistical properties. The user can also control the type and degree of cross-correlation. The user should contact the authors regarding the availability of FGEN.

5.6.3.3 Inactive Elements

These instruction can be used to discretize irregular boundaries with block elements by deactivating portions of the grid. In deactivated portions of the grid, all elements become inactive for both the flow and transport simulation. Elemental assembly is skipped and all nodes that only belong to the inactive element, and are not at all connected to active elements, are assigned default values of head and concentration equal to -9999.0. This option is similar to what is done in MODFLOW to specify inactive cells.

Make element inactive

All chosen elements will become inactive.

Make zones inactive

All elements in the current set of chosen zones will become inactive.

5.6.4 Initial conditions

Initial heads should be given for both steady-state and transient problems since the iterative solver uses them as a starting point in achieving a solution. These heads can be assigned or read from a file.

Initial head

Assigns an initial head value to the current set of chosen nodes. It requires the following input:

1. **hval** Initial head to be assigned.

Restart file for heads

Restarts the simulation with initial heads being read from a file. It requires the following input:

1. **flow_restart_file_name** The name of the file which contains the results of the previous flow solution.

This has the same effect as setting the variable KRESTAR to true. This switch is used in conjunction with the KWRITH flag (see above). If KWRITH was set to 1 during a previous simulation, the results from this previous simulation are saved in a file called, for example, `previous_run.hen`. This file can be used as initial condition when KRESTAR is true. It is recommended that the file be renamed in order to avoid overwriting it and changing the restart conditions.

5.6.5 Boundary conditions

There are two basic options available for assigning boundary conditions to the flow solution. These are to specify either the hydraulic head or the fluid flux at a node. Although these are typically applied to nodes located on the surface of the domain, they can also be applied to internal nodes.

Bear in mind that the definition of wells or tile drains (section 5.6.11) in the problem may include a non-zero specified flux boundary condition and also that the definition of a seepage face (section 5.6.5.2) may lead to the formation of a specified head boundary condition at the seepage nodes.

Once they are defined, you can check the flow boundary conditions using the following instruction:

Echo flow boundary conditions

Cause the currently defined flow boundary conditions to be written to the `.eco` file.

The following instructions can be used to specify flow boundary conditions.

5.6.5.1 Specified Head

This is also known as a first-type, Dirichlet, or constant head boundary condition. It is a nodal property so you should first choose the subset of nodes for which you want to apply the condition and then issue one of the following instructions.

If the node was assigned a specified head or fluid flux value by a previous instruction then it will not be modified by subsequent specified head instructions.

Specified head

Assigns a time-variable head value to all currently chosen nodes. It requires the following input:

1. **npanel** Number of panels in the time-variable head function. A panel is a point in time at which the specified head is set to a new value. The first panel would normally start at time zero. The head given for the last panel will be maintained until the end of the simulation. For each panel, enter the following:
2. **ton_val(), bc_val()** Ton_val() is the time at which bc_val, the head, takes effect.

You can assign a static head for the duration of the simulation by setting npanel to 1 and ton_val() to 0.0.

Specified head from list

Assigns a unique head value to all nodes listed in a separate file. Note that complex time-varying head functions are not supported by this option, just a simple time on/time off scenario. It requires the following input:

1. **fname** Name of the file which contains the number of nodes for which heads will be assigned and the list of node number, time on, time off and head to be assigned. It is formatted as follows:
   (a) **nnde** Number of nodes in list. For each node in the list:
   (b) **nde, ton_val, toff_val, bc_val** Nde is the number of the node to which the head bc_val is to be assigned, ton_val and toff_val are the times at which the head is turned on and off respectively.

For an example, see verification problem THEIS **rgm cross-references**.

Function x head

Assign head values as a linear function of x given heads at two points. The heads specified using this instructions are currently assumed to apply for the duration of the simulation. It requires the following input:

1. **x1,h1** The x-coordinate and head value for the first point.
2. **x2,h2** The x-coordinate and head value for the second point.

Function y head

As above but for the y direction.

Function z head

As above but for the z direction.

5.6.5.2 Seepage Faces

Any surface node can be flagged as a seepage face node. If the hydraulic head at a seepage face node rises above it's elevation, it is flagged as a first-type node, the pressure head is set to zero and water is allowed to flow out. The flow rates for individual nodes are currently reported in the `prefixo.lst` file and overall rates for all seepage face nodes in the mass balance output sections of the `prefixo.lst` file. If the hydraulic head drops below it's elevation, it reverts to it's initial state, which would normally be as a zero or non-zero second-type boundary condition node.

The following two instructions can be used to set up a seepage face. If the node was assigned a specified head or fluid flux value by a previous instruction then it will not be set as a seepage node. *However, unlike the other flow boundary condition instructions, the seepage face condition will override a previous specified fluid flux condition.*

Make seepage face

Makes nodes on all chosen faces seepage nodes unless they were previously flagged as such. These faces should be on the surface of the domain.

Make seepage nodes

Makes all currently chosen nodes seepage nodes. These nodes should be on the surface of the domain.

5.6.5.3 Specified flux

This is also known as a second-type, Neumann, specified or constant flux boundary condition. It is an areal property and so you should first choose the subset of faces for which you want to apply the condition. These faces should be part of the outer boundary of the grid.

If the node was assigned a specified head by a previous instruction then it will not be modified by specified flux instructions.

If the node was assigned a specified fluid flux value by a previous instruction then fluid fluxes assigned in subsequent instructions will be cumulative. this is because fluid fluxes are applied to faces, and any node common to two such faces requires a contribution from each face.

Specified flux

Assigns a given uniform flux value normal to the currently chosen faces. It requires the following input:

1. **npanel** Number of panels in the time-variable flux function. A panel is a point in time at which the specified flux is set to a new value. The first panel would normally start at time zero. The flux given for the last panel will be maintained until the end of the simulation. For each panel, enter the following:
2. **ton_val(), bc_val()** Ton_val() is the time at which bc_val, the normal flux, takes effect.

Specified rainfall

Assigns a given uniform flux value normal to the x-, y-plane to the currently chosen faces. If you want to assign a given quantity of rainfall to an uneven surface you should use this instruction. It requires the following input:

1. **npanel** Number of panels in the time-variable rainfall function.
2. **ton_val(), bc_val()** Ton_val() is the time at which bc_val, the rainfall, takes effect.

Nonuniform flux

Reads a unique flux value for each currently chosen face from a file. It requires the following input:

1. **fname** The name of the file which contains the list of flux values to be assigned.

Note that it is the responsibility of the user to insure that there are enough flux values supplied in the file to satisfy the current number of chosen faces, and that the order of fluxes given matches the order of numbering of the faces in the 3D mesh. This instruction would normally be used to assign spatially variable fluxes to the top face, and in this case, the number of faces chosen would be equal to the number of elements in a 2D slice, and the file of flux values could be generated by a program such as GRID BUILDER.

5.6.5.4 Imported from GMS

Dirichlet (first-type) boundary conditions can be assigned in GMS using the *Select Boundary Nodes* tool in the *3D mesh* module. Once the approprate nodes are chosen, you can assign the boundary condition using the *Assign Node/Face BC...* tool under the *FEMWATER* menu option. In the GMS Node BC dialog, you can select either Constant or Variable head to produce a static or time-variable head function respectively.

Neumann (second-type) boundary conditions can be assigned in GMS using the *Select Boundary Faces* tool in the *3D mesh* module. Once the approprate faces are chosen, you can assign the boundary condition using the *Assign Node/Face BC...* tool under the *FEMWATER* menu option. In the GMS Node BC dialog, you should select Gradient Flux then Constant produce a static fluid flux function. Time-variable fluid fluxes are not currently supported by **HydroSphere**.

Read gms flow boundary conditions

Reads a GMS boundary condition file which has been produced by the FEMWATER module and extracts pertinent flow boundary condition information. It requires the following input:

1. **gmsfile** The name of the file which contains the GMS boundary condition data. Currently, only Dirichlet and Neumann flow boundary conditions are recognized by **HydroSphere**.

5.6.5.5 Imported from GRID BUILDER

Dirichlet (first-type) and Neumann (second-type) boundary conditions can be assigned in GRID BUILDER and exported to a file using the menu option Edit/Boundary/Export_.BND.

Read gb flow boundary conditions

Reads a GRID BUILDER boundary condition file and extracts pertinent flow boundary condition information for a unit-thickness cross-section. It requires the following input:

1. **gbfile** The name of the file which contains the GRID BUILDER boundary condition data. Currently, only Dirichlet and Neumann flow boundary conditions are recognized by **HydroSphere**.

This instruction can only be used with unit-thickness cross-sections.

Read gb first-type boundary conditions

Reads a GRID BUILDER boundary condition file and extracts first-type flow boundary condition information for a 3D domain. It requires the following input:

1. **gbfile** The name of the file which contains the GRID BUILDER first-type boundary condition data.
2. **nsheet_bot,nsheet_top** The bottom and top sheet numbers which are used to limit the selection process. Nodes which are between the bottom and top sheet (inclusive) and correspond to those listed in gbfile will be made first-type head nodes.

This instruction can only be used with unit-thickness cross-sections.

Distributed recharge from gb

Reads a file created by GRID BUILDER which contains a spatially-variable specified flux (recharge) boundary condition and applies it to the top of the 3D domain. It requires the following input:

1. **fname** The name of the file which contains the GRID BUILDER distributed recharge data.
2. **rfac** A multiplication factor to apply to the recharge function. For example, a value of 1.0 would apply the values read from the file while a value of 2.0 would double the recharge values read.

This instruction can only be used with 3D domains created using a Grid Builder slice and the instruction Generate layers from grid builder slice

5.6.6 Solver Parameters

The following instructions may be used to modify the default values for the solver parameters.

Flow solver maximum iterations

Assigns a new value for the maximum number of iterations allowed for the flow solver. The default value is 2000. It requires the following input:

1. **maxfit** The desired value.

Flow solver convergence criteria

Assigns a new value for the flow solver convergence criteria (RMAXTOL). Covergence is

obtained when this criteria is less than the absolute maximum value of the residual (error) of the latest flow solution.

For the matrix equation,

$$[A]x = b$$

where $[A]$ is the coefficient matrix, x the vector of unknowns and b the vector of knowns, we can calculate the residual r, given a solution for x as:

$$r = b - [A]x$$

The default value is 1×10^{-10}. It requires the following input:

1. **rmaxtol** The desired value.

Flow solver detail

Assigns a new flow solver detail level value, which controls the level of detail of solver performance information printed to the listing file. The default value is 1. It requires the following input:

1. **isolv_info** The desired detail level value. Set ISOLV_INFO to 0 for no information, 1 for summary information and 2 for full information.

Flow solver order2

Changes the default level of factorization (ie. first-order) to second-order.

5.6.7 Observation wells and points

The observation well data serves the purpose of outputting the computed head along an imaginary passive well in the domain. It does not affect the results of the computation.

Make observation well

Chooses a continuous set of element face edges and makes them into line elements which act as an observation well. The line elements are chosen to fall on or close to a line defined by the user.

It requires the following input:

1. **x1, y1, z1** The coordinates of one end of the well, which will be located at the node closest to this point.
2. **x2, y2, z2** The coordinates of the other end of the well, which will be located at the node closest to this point. The set of element edges which form the shortest path between the two nodes will become 1D line elements forming the well.

Make observation point

Chooses a node and makes it into an observation point. It requires the following input:

1. **x1, y1, z1** The coordinates of the observation point. The node closest to this point will be chosen as the observation point.

Using either of these instructions causes **HydroSphere** to create a file called `prefixo.obs`. For each timestep in the flow solution, one line per observation well node or observation point will be written to the file. Each line contains the well number, node number, time, hydraulic head, fluid flux and nodal coordinates. *Note that fluid flux at internal nodes will be zero unless they are constant head or specified flux nodes.* Such output can be imported into an editor (e.g. Microsoft Excel) and sorted by column to facilitate, for example, the creation of a plot of fluid flux versus time at a node.

5.6.8 Transient Flow

As mentioned previously, **grok** assumes steady-state flow when generating default data. The following instructions can be used to simulate transient behaviour, and to modify the defaults assumed in that case.

Transient flow

In cases where a transport solution is being done, the pre-processor assumes that any time-related information applies to it, and still defaults the flow solution to steady-state. In order to over-ride this behaviour, and make the flow soulution transient, it is necessary to issue the transient flow instruction.

5.6.8.1 Timestep control

Before we discuss the instructions which are available for controlling the behaviour of the transient flow solution, some background information is required. The pre-processor **grok** generates an array of target times, which are derived from the following sources:

- times specified by the user to meet timestep constraints.
- times specified by the user to meet output requirements.
- times at which transient boundary condition values change.

This target time array is passed to **HydroSphere** which uses it to produce timestep values. As well, if the adaptive timesteps instruction is used, **HydroSphere** will adjust the timestep values based on changes in head, saturation and/or concentration as the solution progresses.

The following instructions can be used to modify the behaviour of the transient flow solution:

Initial time

Assigns a new value for the initial time. This is useful if you are restarting the simulation and want to index the times used to an earlier run. The default initial time value is zero. It requires the following input:

1. **tinit** The desired initial time value.

Initial timestep

Assigns a new value for the initial timestep. The default initial timestep value is 0.01 time units. It requires the following input:

1. **val** The desired inital timestep value.

Maximum timestep

Assigns a new value for the maximum timestep. The default maximum timestep value is 10^{25} time units. It requires the following input:

1. **val** The desired maximum timestep value.

Flow time weighting

Assigns a new value for the time-weighting factor for the flow solution. The default is 1.0. It requires the following input:

1. **tw** The desired time-weighting factor for the flow solution. Values must be greater than 0.0 and less than or equal to 1.0.

Target times

Adds new target times to the current set. It requires the following input:

1. **nts_new** The number of new target times desired. Read the following nts_new times:
   (a) **target_time(i)** Target times.

Generate target times

Generates new target times based on the following input data:

1. **delta, tinc, dtmax, tm** delta is the initial time step size [T], tinc is the time step increment, dtmax is the maximum time step size allowed [T], and tm is the time for the last target time [T]. The new target times are computed by starting from the initial time tinit and by increasing the time step size by a factor tinc until the last target time tm is reached.

Output times

Adds new output times (ie. times for which you want detailed output) to the output time array. Note that these values will automatically become part of the target time array. It requires the following input:

1. **new_nts_out** The desired number of new output times. Read the following new_nts_out times:
   (a) **ouput_time(i)** Output time.

5.6.8.2 Adaptive timesteps

Adaptive timesteps

Causes the model to modify the timestep values as the solution proceeds, based on the transient flow behaviour (see Equation 3.86). Note that if a transport solution is also being done, then the timestep size could also be affected by the parameters specified in the concentration control instruction (see section 5.8.8) as well.

Saturation control

This is to indicate that the transient behaviour of the nodal saturations is to be used to control the adaptive timestep procedure if the system is variably saturated.

It requires the following input:

1. **control_sat, dsat_allowed, dsat_min_allowed** You can specify true or false for the logical variable CONTROL_SAT which, if true, causes the maximum change in sat for one time step to be used to determine the next time step size. By default, CONTROL_SAT is set to false.

   The variable DSAT_ALLOWED is the maximum allowed *percent* change in nodal saturation during any time step, and is used to determine the next time step size [L]. The percent change is calculated based on the range of the saturation function at the start of the time step. The default is 5.0 percent.

   The variable DSAT_MIN_ALLOWED is read but not used.

Head control

This is to indicate that the transient behaviour of the hydraulic heads is to be used to control the adaptive timestep procedure. It requires the following input:

1. **control_head, dhead_allowed, dhead_min_allowed** You can specify true or false for the logical variable CONTROL_HEAD which, if true, causes the maximum change in head for one time step to be used to determine the next time step size. By default, CONTROL_HEAD is set to true.

The variable DHEAD_ALLOWED is the maximum allowed *percent* change in nodal head during any time step, and is used to determine the next time step size [L]. The percent change is calculated based on the range of the head function at the start of the time step, including first-type nodes. The default is 5.0 percent.

The variable DHEAD_MIN_ALLOWED specifies an *absolute* lower limit on the head changes which will result in time step reduction. This eliminates any uneccessary reduction in the timestep if the range in the head function becomes very small. In such cases the percent change could be greater than DHEAD_ALLOWED while the absolute change in head was negligible. The default is 0.01.

5.6.9 Variably-saturated flow

The following instruction tells **HydroSphere** to perform a variably-saturated flow simulation.

Unsaturated

Unless you modify the default values, all zones (and elements) in the domain will be assigned default variably-saturated porous media constitutive properties. These are to assume pseudo-soil relations, as developed by *Huyakorn et al.* [1994]. Essentially, in the pseudo soil relation, the porous medium is assigned a nodal saturation of 1 above the water table and 0 (zero) below it. Relative permeability is applied to horizontal flow only and water travels vertically under saturated hydraulic conductivity conditions.

You can use the methods and instructions outlined in section 5.1.4 to modify the default distribution of the variably-saturated porous media properties.

The following instructions can be used in the `.grok` file or a porous media material properties file to modify the porous media constitutive relationships for the current set of chosen zones.

Unsaturated functions

Instructs **grok** to use functions to describe the constitutive relationships for the porous media and to begin reading instructions which can be used to modify the function parameters that define the relationships. If no further instructions are issued, the default function parameter values listed in Table 5.2 will be used.

The Unsaturated functions instruction overrides the pseudo-soil default so that relative permeability is applied to horizontal and vertical flow.

The following instructions are all optional, with the exception of End, which must be present to signal the end of the instruction list for the unsaturated functions

Residual saturation

Assigns a residual saturation to the currently chosen set of zones. It requires the following input:

Parameter	Value	
Residual water saturation, S_{wr}	0.053	
Power index (alpha), α	3.5237	1/m
Power index (beta), β	3.1768	
Power index (gamma, computed), $\|gamma$	$1 - 1/\beta$	
Exponent (krw-pressure equation)	0.0d0	
Air-entry pressure	0.0d0	m

Table 5.2: Default values for functions defining the porous media constitutive relationships

1. **val** Residual saturation

Alpha

Assigns a power index alpha value to the currently chosen set of zones. It requires the following input:

1. **val** Power index alpha [1/L]

Beta

Assigns a power beta value to the currently chosen set of zones. It requires the following input:

1. **val** Power index beta

Exponent

Assigns an exponent (krw-pressure equation) to the currently chosen set of zones. It requires the following input:

1. **val** Exponent (krw-pressure equation). Set the exponent to zero for van Genuchten functions and to 1 for Brooks-Corey functions.

Air entry pressure

Assigns an air entry pressure to the currently chosen set of zones. It requires the following input:

1. **val** Air entry pressure [L].

End

This instruction signals the end of the unsaturated function input.

Unsaturated tables

Instructs **grok** to use functions to describe the constitutive relationships for the porous

Pressure(m)	*Saturation*
-10.0	0.053
0.0	1.0

Table 5.3: Default pressure-saturation table for variably-saturated porous media

S_w	K_{rw}
0.053	0.053
1.0	1.0

Table 5.4: Default saturation-relative permeability table for variably-saturated porous media

media and to begin reading instructions which describe the pressure-saturation and/or saturation-relative k tables which will define the constitutive relationships for the porous media. If no further instructions are issued, the default values of water saturation versus pressure head listed in Table 5.3 and saturation versus relative permeability listed in table 5.4 will be used.

The Unsaturated tables instruction overrides the pseudo-soil default so that relative permeability is applied to horizontal and vertical flow.

The following instructions are all optional, with the exception of End, which must be present to signal the end of the instruction list for the unsaturated tables.

Pressure-saturation

Assigns the following pressure-saturation table to zone iz. It requires the following input:

pstab_pw(1,iz), pstab_sw(1,iz) Pressure and saturation for first entry

pstab_pw(2,iz), pstab_sw(2,iz) Pressure and saturation for second entry

⋮ etc.

pstab_pw(n,iz), pstab_sw(n,iz) Pressure and saturation for nth entry

end_card The string 'end'

Note that these numbers must be input sequentially from the lowest (i.e. largest negative) pressure value to the highest pressure value, usually zero. The number of entries in the list are counted automatically to determine the table size.

Saturation-relative k

Assigns the following saturation-relative k relationship to zone iz. It requires the following input:

sktab_sw(1,iz), sktab_krw(1,iz) Saturation and relative hydraulic conductivity for first entry

sktab_sw(2,iz), sktab_krw(2,iz) Saturation and relative hydraulic conductivity second entry

⋮ etc.

sktab_sw(n,iz), sktab_krw(n,iz) Saturation and relative hydraulic conductivity for n^{th} entry

end_card The string 'end'

The number of entries in the list are counted automatically to determine the table size.

End

This instruction signals the end of the unsaturated table input.

You can modify the variably-saturated parameters of the current set of chosen zones using the following instructions. Setting the chosen zones is described in section 5.5.6.

relative permeability xy

Modify only the x-, y-components of the relative permeability function for the current set of chosen zones.

pressure relative permeability

rgm currently has no effect. Use relative permeability function of pressure instead of saturation for the current set of chosen zones.

5.6.9.1 Newton iteration parameters

The following parameters can be used to control the Newton-Raphson iteration scheme for solution of the variably-saturated flow problem.

Newton maximum iterations

Assigns a new value for the maximum number of Newton iterations allowed for solution, i.e. for any one time step. If this number is exceeded, the current time step is reduced by half and the simulation is restarted at the new time value. The default value is 15. It requires the following input:

1. **nval** The desired maximum number of newton iterations.

Jacobian epsilon

Assigns a new value for the shift in pressure head used to compute the derivatives in the Jacobian matrix numerically. As a rule of thumb, a value equal to 10^{-5} times the average pressure head in the domain should be used. The default value is 1×10^{-4}. It requires the following input:

1. **val** The desired Jacobian epsilon value.

Newton absolute convergence criteria

Assigns a new value for the absolute convergence criteria DELNEWT. Convergence of the solution occurs when the maximum absolute nodal change in pressure head over the domain for one Newton iteration is less than this value. The default value is 1×10^{-5}. It requires the following input:

1. **val** The desired Newton delnewt value.

Newton residual convergence criteria

Assigns a new value for the residual convergence criteria RESNEWT. Convergence of the solution occurs when the maximum absolute nodal residual (see section 5.6.6) in the domain for one Newton iteration exceeds this value. The default value is 1×10^{-8}. It requires the following input:

1. **val** The desired newton resnewt value.

Underrelaxation factor

Assigns a new value for the underrelaxation factor for the Newton iteration. Set it to a number between 0 and 1, with 1 being no underrelaxation. The default value is 1. It requires the following input:

1. **val** The desired underrelaxation factor value.

Compute underrelaxation factor

Causes the underrelaxation factor ω to be computed according to the following method described by *Cooley* [1983].

$$\begin{aligned} \omega_{r+1} = & \quad \frac{3+s}{3+|s|} \quad if\, s \geq -1 \\ = & \quad \frac{1}{2|s|} \quad if\, s < -1 \end{aligned} \tag{5.1}$$

where

$$\begin{aligned} s = & \quad \frac{e_{r+1}}{e_r \omega_r} \quad if\, r > 1 \\ = & \quad 1 \quad if\, r = 1 \end{aligned} \tag{5.2}$$

In the equations presented above, r and $r+1$ represent the previous and current iteration level, ω_r and ω_{r+1} represent the underrelaxation factor for the previous and current iteration levels, and e represents the maximum value of the largest difference between head values for 2 successive iterations, $e_r = Max_I \mid \psi_I^r - \psi_I^{r-1} \mid$.

Compute underrelaxation factor limit

Specify an upper limit on the computed underrelaxation factor. A suggested value is 10 times the system domain thickness. Default value is 1000.

Newton information

Causes **HydroSphere** to write more detailed information to the listing file about the performance of the Newton iteration process.

5.6.10 Fractures

Physical parameters for a variety of fracture zones or sets can be defined in a file called `default.fprops` (ie. fracture properties) which should be located in the same directory as the `.grok` file. Any line in the `default.fprops` file which is completely blank or which begins with an exclamation point(!) will be ignored. This allows you to include comments whenever required. This is similar to the approach described in section 5.1.4 for defining material properties.

You can create your own fracture property files and give them more meaningful names. In this case you should instruct **grok** to read these files rather than the defaults using the following instruction:

Fracture properties file

It requires the following input:

1. **fprops_file_name** Name of the file which contains the fracture properties to be used.

Each fracture material in the `default.fprops` file is identified by a unique label and requires the following parameters for saturated flow, unsaturated flow and transport:

frac_name A string variable up to 50 characters which identifies the name of the fracture material whose properties are to follow.

stfrac Specific storage coefficient for a fluid-filled fracture [1/L]

fracyes You can specify true or false for this logical variable which controls whether to treat the current set as fractures or as high-conductivity planes. If true, they are treated as fractures and the cubic law is used to determine fracture conductivity. If false the set refers to a high-conductivity plane and the hydraulic conductivity is given by the user.

frack Hydraulic conductivity [1/L] for a high-conductivity plane if fracyes above is false. Not used if fracyes is true.

aperture Fracture aperture [L]. This variable is not used if random fractures were generated.

tab_dataf You can specify true or false for this logical variable which controls whether to use tabular data or a function to describe the unsaturated hydraulic properties for the current fracture zone. If true, read the following:

npstab_sizef Number of entries for the pressure-saturation relationship.

pstab_pwf(i),pstab_swf(i),i=1,npstab_sizef Pressure head and corresponding saturation for each of the npstab_sizef entries. Note that these numbers must be input sequentially from the lowest pressure value to the highest pressure value. This means that for negative pressure values, input must start with the biggest negative value.

nsktab_sizef Number of entries for the saturation- relative permeability relationship.

sktab_swf(i),sktab_krwf(i),i=1,nkstab_sizef Saturation and corresponding relative permeability for each of the nsktab_sizef entries. Note that these numbers must be input from low to high saturation.

If false read the following:

swrf, alphaf, betaf, gammaf, expnf, aentryf Residual saturation (S_{wr}), fitting parameters α[1/L], β and γ, for the van Genuchten function (note that $\gamma = 1-1/\beta$), exponent for the relative permeability-saturation function and air entry pressure. If expnf is set greater than 0, the Brooks-Corey expression, instead of the van Genuchten, will be used for the relative permeability vs. saturation function.

ncarea_sizef Number of entries for the pressure-contact area relationship.

patab_pwf(i),patab_caf(i),i=1,ncarea_sizef Pressure head and corresponding contact area for each of the ncarea_sizef entries. These values are only used if the instruction Contact area (see below) is issued in the `.grok` file.

alfrac Longitudinal dispersivity [L]

atfrac Transverse dispersivity [L]

Currently, retardation factors for each mobile species are set to 1.0 (i.e. no retardation in fractures).

As was the case for material properties, the parameters needed to simulate unsaturated flow and solute transport are both given, in spite of the fact that the default problem is saturated and transport is not simulated. These are included so the user does not have to modify the `fprops` file if he decides to change the default problem to include unsaturated flow or solute transport.

Table 5.5 is an example `fprops` entry for a material called fracture. In this case the unsaturated behaviour is defined by tabular input.

You can put data sets for as many different material properties as you like in the mprops file as long as each one has a unique name. You can use the material name later in conjunction

```
fracture
0.0                Fracture storage coefficient
.true.             True if fracture, false if high-K plane
1.0                Hydraulic conductivity if high-K plane
.0001              Aperture if fracture, thickness if higk-K plane
.true.             True for tabular data
2                  Pressure-saturation table
0.0 1.0
-1.0 1.0
2                  Saturation-relative K table
0.0 1.0
1.0 1.0
1                  Pressure-contact area table
1.0 1.0
1.0                Longitudinal dispersivity
1.0                Transverse dispersivity
```

Table 5.5: FPROPS parameters for a material called fracture

with a pre-processor instruction to assign different material properties to specific zones or arbitrary subsets of elements.

Normally, fracture sets are defined after grid generation is complete using the instruction Make fractures. In this case it is up to the user to indicate which set of fracture properties are to be assigned to the current set of chosen faces. However, if the random fracture generation option is used (see section 5.3.3) the three fracture sets, which are oriented parallel to the XY, XZ and YZ planes will be assigned fracture zone numbers 1,2 and 3 respectively. By default, the first set of data encountered in the file `fprops` will be used as the fracture properties for all faces with fracture zone number 1, the second set for faces with fracture zone number 2, etc... until the end of the data file is reached. If there are still faces which have not received data (ie. more fracture zone numbers than data sets in the file) they will be assigned fracture properties using the first set of data encountered in the file. It is possible to superimpose other sets of fractures on the random ones using the instruction Make fractures but bear in mind that if you do so, you should set up `fprops` so that the first three sets of data are for the random fractures and the new set is in the fourth position.

Echo fracture incidences

Writes the current fracture element incidences to the `.eco` file.

Make fractures

Creates a new set of fractures using all of the currently chosen faces (see section 5.5.3) and reads fracture properties from the file `fprops`. It requires the following input:

mat_name The name of the fracture material (as identified in the `fprops` file) whose properties are to be read and assigned.

Impermeable matrix

This command causes the matrix to be considered impermeable and so flow and transport will only be computed for the fractures. This overrides the values in the mprops file and so you do not have to alter them.

Contact area

Reduces the available area (contact area) for flow normal to the fracture surface under unsaturated conditions according to the function provided by *Wang and Narasimhan* [1985].

5.6.10.1 Recharge Spreading Layers

A recharge spreading layer is a zone of relatively high hydraulic conductivity which allows recharge water to infiltrate preferentially into zones with high hydraulic conductivity (e.g. fractures).

Make recharge spreading layer

Creates a new set of 2D planar elements using the current set of chosen elements. It requires the following input:

1. **frack** Hydraulic conductivity [L/T] of the recharge spreading layer.
2. **aperture** Thickness [L] of the recharge spreading layer.

NOTE: A recharge spreading layer will allow water to bypass the rest of the system if it is connected to or in close proximity to a constant head boundary which is acting as a discharge point for the system. In such cases you may have to choose a subset of faces and assign properties using the Make fractures instruction to achieve the desired results. If your intent is to distribute water preferentially between fractures and low K matrix, experience has shown that recharge spreading layer transmissivities two orders of magnitude higher than the matrix are usually sufficient.

5.6.11 Wells

Make well

Chooses a set of element face edges (ie. segments) and makes them into 1D line elements which act as an open (ie. fluid-filled) well. The line elements are chosen to fall on or close to a line defined by the user. Currently, in cases where the flow rate is non-zero, the water will be extracted from the uppermost node in the well screen. The reader is referred to *Sudicky et al.*, [1995] for details on how wells are handled in **HydroSphere**. It requires the following input:

1. **well_name()** A descriptive name for the well. Up to 40 characters.

2. **x1, y1, z1** The coordinates of one end of the well, which will be located at the node closest to this point.

3. **x2, y2, z2** The coordinates of the other end of the well, which will be located at the node closest to this point. The set of element edges which form the shortest path between the two nodes will become 1D line elements forming the well.

4. **npanel** Number of time panels for which a well flowrate is specified.

5. **ton_val(), flowrate()** Ton_val() is the time at which the flowrate [L^3/T] takes effect. If the flowrate is set to zero, the well is passive but can still transmit water vertically through its screen.

6. **xf, yf, zf** The spatial coordinates corresponding to the location of injection-withdrawal in the well. The well node closest to these coordinates will be selected.

7. **radw()** Screen radius [L]

8. **radc()** Casing radius [L]

The hydraulic conductivity of the well [L/T] is calculated (assuming laminar flow) as:

$$K_w = \frac{r_s^2 \rho g}{8\mu}$$

where r_s is the well screen radius, ρ is the fluid density, g is the gravitational acceleration constant and μ is the fluid viscosity.

The specific storage coefficient of the well [1/L] is calculated as:

$$S_s = \rho g \beta$$

where β is the compressibility of water.

Make infilled well

Chooses a set of element face edges and makes them into 1D line elements which act as an infilled well. The line elements are chosen to fall on or close to a line defined by the user. Currently, in cases where the flow rate is non-zero, the water will be extracted from the uppermost node in the well screen. The reader is referred to *Sudicky et al.*, [1995] for details on how wells are handled in **HydroSphere**. It requires the following input:

1. **well_name()** A descriptive name for the infilled well. Up to 40 characters.

2. **x1, y1, z1** The coordinates of one end of the well, which will be located at the node closest to this point.

3. **x2, y2, z2** The coordinates of the other end of the well, which will be located at the node closest to this point. The set of element edges which form the shortest path between the two nodes will become 1D line elements forming the well.

4. **npanel** Number of time panels for which a well flowrate is specified.
5. **ton_val(), flowrate()** Ton_val() is the time at which the flowrate [L^3/T] takes effect. If the flowrate is set to zero, the well is passive but can still transmit water vertically through its screen.
6. **xf, yf, zf** The spatial coordinates corresponding to the location of injection-withdrawal in the well. The well node closest to these coordinates will be selected.
7. **radw()** Screen radius [L]
8. **radc()** Casing radius [L]
9. **mat_name** The name of the material (as identified in the `mprops` file) whose vertical hydraulic conductivity ($K_z z$) and storage coefficient are to be used for the infill material.

Make well node

Chooses a single node and makes it an open (ie. fluid-filled) well. The reader is referred to *Sudicky et al.*, [1995] for details on how wells are handled in **HydroSphere**. It requires the following input:

1. **well_name()** A descriptive name for the well. Up to 40 characters.
2. **x1, y1, z1** The coordinates of the node which will be defined as a well. The nearest node to this point is chosen.
3. **npanel** Number of time panels for which a well flowrate is specified.
4. **ton_val(), flowrate()** Ton_val() is the time at which the flowrate [L^3/T] takes effect. If the flowrate is set to zero, the well is passive but can still transmit water vertically through its screen.
5. **radw()** Screen radius [L]
6. **radc()** Casing radius [L]

The hydraulic conductivity and storage coefficient of the well are calculated as above in the Make well instruction.

5.6.12 Tile drains

Make tile drain

Chooses element face edges and makes them into 1D line elements which act as a tile drain. The line elements are chosen to fall on or close to a polyline defined by the user. Currently, this option requires that the the system be variably-saturated. It requires the following input:

1. **tile_name()** A descriptive name for the tile drain. Up to 40 characters.
2. **npts** The number of points defining the polyline. Read the following npts times:
   (a) **x1, y1, z1** The coordinates of a point on the polyline. The points should be given in order from one end of the polyline to the other. The routine finds the nodes closest to the coordinates of the points given and then finds the segments forming the shortest path between the nodes.
3. **wid()** Tile width [L]
4. **qtile()** Tile discharge rate [L^3/T]. If the flowrate is set to zero, the tile is passive but can still collect or exfiltrate water.
5. **xd, yd, zd** The coordinates of the tile discharge point. This point should correspond to one of the points given for the polyline and is normally coincident with one end of the tile. *If the point is not located at the end of the tile, the total tile discharge rate will be twice that specified above.*

5.6.13 Cutoff Walls

Cutoff Walls

Defines impermeable cutoff walls which are currently used to represent internal impermeable boundaries as is the case for funnel-and-gate simulations. Currently, because of certain assumptions inherent in the grid numbering scheme, it must be used in conjunction with rectangular block element grids which have been generated by using the Group 4 instruction. It requires the following input:

1. **walls** A logical switch which, if true, cause the preprocessor to read the following information.
2. **nwalls** The number of impermeable cutoff walls to be defined. Read the following nwalls times:
   (a) **iorient()** Set iorient to 1 if the cutoff wall is in the XZ plane. Set iorient to 2 if the cutoff wall is in the YZ plane.
   (b) **avalue(),afrom(),ato(),verfrom(),verto()** The values input here depend on the variable IORIENT. If IORIENT is 1 (wall in XZ plane) then avalue() is the Y-coordinate of the wall and afrom() and ato() are the X-coordinates of the ends of the wall. If IORIENT is 2 (wall in YZ plane) then avalue() is the X-coordinate of the wall and afrom() and ato() are the Y-coordinates of the ends of the wall. In either case verfrom() and verto() are the Z-coordinates of the bottom and top of the wall respectively.

Parameter	Value	
Name	Overland flow defaults	
X friction factor	0.0548	
Y friction factor	0.0548	
Rill storage height h_{ds}	0.0	m
Obstruction storage height h_o	0.0	m
Node coupling scheme	shared	
Dual scheme recharge	0.0	m/s
Dual scheme bottom recharge	0.0	m/s

Table 5.6: Default properties for overland flow.

5.7 Overland Flow

5.7.1 General

By default, **HydroSphere** does not simulate overland flow unless an overland flow zone is created using the methods and instructions outlined in section 5.1.4. Unless you modify the default values, all zones (and elements) in the the overland flow domain will be assigned the default properties listed in Table 5.6. .

The following instructions can be used in the `.grok` file or an overland flow properties file to define material parameter values for the current set of chosen zones. Setting the chosen zones is described in section 5.5.6.

X friction

Assigns an x friction factor to the currently chosen set of zones. It requires the following input:

1. **val** x friction factor

Y friction

Assigns a y friction factor to the currently chosen set of zones. It requires the following input:

1. **val** y friction factor

Rill storage height

Assigns a rill storage height to the currently chosen set of zones. It requires the following input:

1. **val** rill storage height [L]

Parameter	Value	
Longitudinal dispersivity α_L	1.0	m
Transverse horizontal dispersivity α_{TH}	0.1	m
Transverse vertical dispersivity α_{TV}	0.1	m
Bulk density ρ	2650.0	kg/m^3
Tortuosity τ	0.1	
Immobile zone porosity θ_{Imm}	0.0	
Immobile zone mass transfer coefficient α	0.0	1/s

Table 5.7: Default values for porous media transport properties

Obstruction storage height

Assigns an obstruction storage height to the currently chosen set of zones. It requires the following input:

1. **val** obstructions storage height [L]

rgm The file ./surface/olf1_input_specifics.tex contains the mod-hms general input discussion which describes functionality which needs to be implemented in HydroSphere

5.8 Solute Transport

5.8.1 General

The following instructions tells **HydroSphere** to perform a transport simulation.

Do transport

5.8.2 Transport properties

Unless you modify the default values, all zones (and elements) in the domain will be assigned default porous media transport properties which are listed in Table 5.7.

Furthermore, the default mode does not consider isotopic fractionation.

You can use the methods and instructions outlined in section 5.1.4 to modify the default distribution of the porous media transport properties.

The following instructions can be used in the `.grok` file or a porous media material properties file to define material parameter values for the current set of chosen zones. Setting the chosen zones is described in section 5.5.6.

Longitudinal dispersivity

Assigns a new value for the longitudinal dispersivity. It requires the following input:

1. **val** Longitudinal dispersivity [L]

Transverse horizontal dispersivity

Assigns a new value for the horizontal transverse dispersivity. it requires the following input:

1. **val** Horizontal transverse dispersivity [L]

transverse vertical dispersivity

Assigns a new value for the vertical transverse dispersivity. It requires the following input:

1. **val** Vertical transverse dispersivity [L]

Tortuosity

Assigns a new value for the tortuosity. It requires the following input:

1. **val** Tortuosity value.

Bulk density

Assigns a new value for the bulk density. It requires the following input:

1. **val** Bulk density [M/L^3]

Immobile zone porosity

Assigns a new value for the immobile zone porosity θ_{Imm}. It requires the following input:

1. **val** Immobile zone porosity

Immobile zone mass transfer coefficient

Assigns a new value for the mass transfer coefficient describing diffusion between the mobile and immobile zones. It requires the following input:

1. **val** Mass transfer coefficient [1/T]. It is possible to relate the mass transfer coefficient α to the physical properties and geometry of the immobile zones. If they are assumed to be comprised of spheres, then

$$\alpha = \frac{15\theta_{\text{Imm}} D^*_{\text{Imm}}}{r_0^2}$$

Parameter	Value
Forward fractionation rate, k_r	0.0 1/s
Fractionation factor, α_r	0.0
Mass ration, solid to water phases, x_r	0.0

Table 5.8: Default values for parameters defining the isotopic fractionation

where r_0 is the radius of an anonymous sphere and D^*_{Imm} is the effective diffusion coefficient. If the immobile zone is represented as prismatic slabs, then

$$\alpha = \frac{3\theta_{\text{Imm}} D^*_{\text{Imm}}}{B^2}$$

where $2B$ is the thickness of the slabs, being equivalent to the fracture spacing. For more details, the reader is referred to *Sudicky* [1990]. If the dual porosity option is not desired then simply set α and θ_{Imm} to 0.0.

Isotope fractionation data

Instructs **grok** to use read parameters that describes isotopic exchange between the water (mobile) and solid (immobile) phases. If no further instructions are issued, the default function parameter values listed in Table 5.8 will be used.

The following instructions are all optional, with the exception of End, which must be present to signal the end of the instruction list for the isotopic fractionation.

rock-water mass ratio

Forward rate

Assigns a forward fractionation rate to the currently chosen set of zones. It requires the following input:

1. **val** Forward rate [1/L].

Fractionation factor

Assigns a fractionation factor to the currently chosen set of zones. It requires the following input:

1. **val** Fractionation factor.

Rock-water mass ratio

Assigns an isotopic rock-water mass ratio to the currently chosen set of zones. It requires the following input:

1. **val** Rock-water mass ratio.

End

This instruction signals the end of the isotopic fractionation input.

5.8.3 Initial conditions

Currently the initial condition (for both mobile and immobile zones if dual porosity) defaults to 0.0 unless one of the following instructions are included.

Initial concentration

Assigns an initial concentration value to the current set of chosen nodes. It requires the following input:

1. **val** Initial concentration to be assigned.

Initial concentration from file

Reads an initial concentration for each node from a free-format ascii file. The concentrations should be written to the file in order from node 1 to node NN and each line can contain one or more values. It requires the following input:

1. **fname** Name of the file which contains the initial concentrations to be assigned.

Initial immobile zone concentration

Assigns an initial immobile zone concentration value to the current set of chosen nodes. It requires the following input:

1. **val** Initial concentration to be assigned.

Initial immobile zone concentration from file

Reads an initial immobile zone concentration for each node from a free-format ascii file. The concentrations should be written to the file in order from node 1 to node NN and each line can contain one or more values. It requires the following input:

1. **fname** Name of the file which contains the initial concentrations to be assigned.

Restart file for concentrations

Restarts the simulation with initial concentrations being read from a file. It requires the following input:

1. **transport_restart_file_name** The name of the file which contains the results of the previous flow solution.

This has the same effect as setting the variable KRESTARC to true. This switch is used in conjunction with the KWRITHC flag (see above). If KWRITHC was set to 1 during a previous simulation, the results from this previous simulation are saved in a file called, for example, `previous_run.cen`. This file can be used as initial condition when KRESTARC is true. It is recommended that the file be renamed in order to avoid overwriting it and changing the restart conditions.

5.8.4 Boundary conditions

5.8.4.1 Specified concentration

This is also known as a first-type, Dirichlet, or constant concentration boundary condition. It is a nodal property so you should first choose the subset of nodes for which you want to apply the condition and then issue one of the following instructions.

If the node was assigned a specified concentration, mass flux or third-type value by a previous instruction then it will not be modified by subsequent specified concentration instructions.

Specified concentration

Assigns a time-variable concentration value to all currently chosen nodes. If the node was previously assigned a specified concentration, It requires the following input:

1. **npanel** Number of panels in the time-variable concentration function. A panel is a point in time at which the specified concentration is set to a new value. The first panel would normally start at time zero. The concentration given for the last panel will be maintained until the end of the simulation. For each panel, enter the following:
2. **ton_val(i),toff_val(i), (bc_val(i,j),j=1,nspeciesmob)** Ton_val() and toff_val(i) are the start and end times at which bc_val(), the concentration, is applied. If more than one species is being simulated, additional values of bc_val() should be included on the line.

You can assign a concentration for the duration of the simulation by setting npanel to 1, ton_val() to 0.0 and toff_val() to a large number (ie. greater than the duration of the simulation).

Specified well concentration

Assigns a time-variable first-type concentration value to a well, which is uniform along the length of the well. It requires the following input:

1. **iw** The number of the well to which the concentration is to be applied. For each species defined, enter the following:
   (a) **ninjc()** The number of panels in the injection concentration history for this well and species. For each panel ninjc(), enter the following:
      i. **cinjc(),toninjc(),toffinjc()** Toninjc() and toffinjc(i) are the start and end times at which cinjc(), the concentration, is applied.

NOTE: Since this instruction loops over all defined species, you must supply input even if it is a zero concentration.

Specified tile concentration

Assigns a time-variable first-type concentration value to a tile drain, which is uniform along the length of the tile. It requires the following input:

1. **iw** The number of the tile to which the concentration is to be applied. For each species defined, enter the following:
   (a) **ntdc()** The number of panels in the injection concentration history for this well and species. For each panel ntdc(), enter the following:
      i. **c_tile(),ton_c_tile(),toff_c_tile()** Ton_c_tile() and toff_c_tile(i) are the start and end times at which c_tile(), the concentration, is applied.

NOTE: Since this instruction loops over all defined species, you must supply input even if it is a zero concentration.

5.8.4.2 Specified mass flux

This is also known as a second-type, Neumann, or constant mass flux boundary condition. It is a distributed property so you should first choose the subset of faces for which you want to apply the condition and then issue one of the following instructions.

If the node was assigned a specified concentration or third-type concentration by a previous instruction then it will not be modified by further specified mass flux instructions.

If the node was assigned a specified mass flux value by a previous instruction then mass fluxes assigned in by subsequent instructions will be cumulative. This is because mass fluxes are applied to faces, and any node common to two such faces requires a contribution from each face

Specified mass flux

Assigns a time-variable mass flux per unit time value to all currently chosen nodes. This is a passive injection of solute mass which has no effect on the flow solution. It requires the following input:

1. **npanel** Number of panels in the time-variable mass flux function. A panel is a point in time at which the specified mass flux is set to a new value. The first panel would normally start at time zero. The mass flux given for the last panel will be maintained until the end of the simulation. For each panel, enter the following:
2. **ton_val(i),toff_val(i), (bc_val(i,j),j=1,nspeciesmob)** Ton_val() and toff_val(i) are the start and end times at which bc_val(), the mass flux per unit time, is applied. If more than one species is being simulated, additional values of bc_val() should be included on the line.

You can assign a mass flux for the duration of the simulation by setting npanel to 1, ton_val() to 0.0 and toff_val() to a large number (ie. greater than the duration of the simulation).

Note that the mass flux values are per unit time and the total mass which will be injected for a given timestep can be calculated by multiplying the value given here by the timestep length and the number of chosen nodes.

5.8.4.3 Specified third-type concentration

This is also known as a third-type or Cauchy boundary condition. It is a distributed property so you should first choose the subset of faces for which you want to apply the condition and then issue one of the following instructions.

If the node was assigned a specified concentration by a previous instruction then it will not be modified by further specified mass flux instructions.

If the node was assigned a third-type concentration value by a previous instruction then third-type concentration fluxes assigned in subsequent instructions will be cumulative. This is because third-type concentrations are applied to faces, and any node common to two such faces requires a contribution from each face

Specified third-type concentration

Assigns a time-variable third-type concentration value to all currently chosen faces. It requires the following input:

1. **calcflux** This is a logical variable which controls how fluid fluxes are handled for this third-type boundary condition. If true, the fluxes are calculated by **HydroSphere** from the flow solution. Only positive fluxes (ie. flowing into domain) are used. If false, read the following:
   (a) **userflux** The fluid flux value to be used in conjunction with the third-type concentration given below.
2. **npanel** Number of panels in the time-variable concentration function. A panel is a point in time at which the specified concentration is set to a new value. The first panel would normally start at time zero. The concentration given for the last panel will be maintained until the end of the simulation. For each panel, enter the following:
3. **ton_val(i),toff_val(i), (bc_val(i,j),j=1,nspeciesmob)** Ton_val() and toff_val(i) are the start and end times at which bc_val(), the concentration, is applied. If more than one species is being simulated, additional values of bc_val() should be included on the line.

You can assign a concentration for the duration of the simulation by setting npanel to 1, ton_val() to 0.0 and toff_val() to a large number (ie. greater than the duration of the simulation).

Nonuniform third-type concentration

Reads a unique time-variable third-type concentration value for each currently chosen face from a file. It requires the following input:

1. **fname** The name of the file which contains the list of time-variable third-type concentration values to be assigned. The input format required is identical to that given above for the instruction Specified third-type concentration.

Note that it is the responsibility of the user to insure that there are enough time-variable third-type concentration values supplied in the file to satisfy the current number of chosen faces, and that the order of fluxes given matches the order of numbering of the faces in the 3D mesh. This instruction would normally be used to assign spatially variable third-type concentration values to the top face, and in this case, the number of faces chosen would be equal to the number of elements in a 2D slice, and the file of flux values could be generated by a program such as GRID BUILDER.

5.8.4.4 Imported from GMS

Read gms transport boundary conditions

Reads a GMS boundary condition file which has been produced by the FEMWATER module and extracts pertinent transport boundary condition information. It requires the following input:

1. **gmsfile** The name of the file which contains the GMS boundary condition data.

Currently, only Dirichlet and Cauchy transport boundary conditions are recognized by **HydroSphere**. Also, this feature only works for a single species. If more than one species is present in the simulation, use the instructions specified concentration and specified third-type concentration to define the input data.

Dirichlet (first-type) boundary conditions can be assigned in GMS using the *Select Boundary Nodes* tool in the *3D mesh* module. Once the approprate nodes are chosen, you can assign the boundary condition using the *Assign Node/Face BC...* tool under the *FEMWATER* menu option. In the GMS Node BC dialog, you can select either Constant or Variable concentration to produce a static or time-variable concentration function.

Cauchy (third-type) boundary conditions can be assigned in GMS using the *Select Boundary Faces* tool in the *3D mesh* module. Once the approprate faces are chosen, you can assign the boundary condition using the *Assign Node/Face BC...* tool under the *FEMWATER* menu option. In the GMS Node BC dialog, you should select Flux then Constant or Variable to produce a static or time-variable mass flux function.

5.8.4.5 Immiscible phase dissolution source

Immiscible phase dissolution data

Makes all currently chosen nodes dissolution nodes. It is assumed that there is dissolution of the immiscible phase (that can be liquid or solid) into the subsurface water until all

dissolvable material is exhausted. For the dissolution, first-type boundary conditions are applied at the node. When all the material associated with a node has dissolved, the node reverts back to unconstrained conditions. It requires the following input:

1. **diss_mass** Grams of dissolvable material per unit volume of porous medium. It is assumed that the total mass of dissolvable material is located in the matrix only. The total mass available for dissolution associated with a given node is then calculated by multiplying contributing volume (from all shared elements) times the porosity times the variable diss_mass.

2. **diss_conc** The aqueous solubility value for the immiscible phase in mass of solute per unit volume of aqueous solution. The node is maintained as a first-type node of concentration diss_conc.

NOTE: Do not put dissolution nodes on a domain boundary. When the mass is exhausted the node reverts to being unconstrained and any water flowing in will generate mass due to the nonzero concentration.

5.8.4.6 Zero-order source

Zero-order source

Makes all nodes in the set of currently chosen zones time-varying, zero-order source nodes. A zero-order source is one in which the medium itself produces a solute. For example, some soils produce radon gas as a result of radioactive decay. It requires the following input:

1. **npanel** Number of panels in the time-variable, zero-order source term function. A panel is a point in time at which the zero-order source term is set to a new value. The first panel would normally start at time zero. The zero-order source term given for the last panel will be maintained until the end of the simulation. For each panel, enter the following:

2. **ton_val(i),toff_val(i), (bc_val(i,j),j=1,nspeciesmob)** Ton_val() and toff_val(i) are the start and end times at which bc_val(), the zero-order source term, is applied. The units of the source term are: Mass of solute produced per unit volume of porous medium solids per unit time $[M/L^3/T]$. If more than one species is being simulated, additional values of bc_val() should be included on the line.

For an example, see verification problem ZERO **rgm cross-reference**.

5.8.5 Solver Parameters

The following instructions may be used to modify the default values for the solver parameters.

Transport solver convergence criteria

Assigns a new value for the transport solver convergence criteria (RMAXTOLC). Covergence is obtained when this criteria is less than the absolute maximum value of the residual (error) of the latest transport solution. The default value is 1×10^{-10}. It requires the following input:

1. **rmaxtolc** The desired value.

Transport solver output detail

Assigns a new transport solver detail level value, which controls how much solver performance information is printed to the listing file during the transport solution. A value of 0 means that no information will be written, a value of 1 causes summary information to be written, while a value of 2 causes detailed information to be written. The default value is 1. It requires the following input:

1. **isolv_info** The desired detail level value.

Upstream weighting of velocities

Causes upstream-weighting of velocities to be used, as opposed to the default, central weighting of velocities. It requires the following input:

1. **almax,btmax,gammax** Upstream weighting factors for the x-direction (almax), the y-direction (btmax) and the z-direction (gammax). Range from 0.0i for no upstream-weighting to 1.0 for full upstream weighting. Note that these variables do not apply for the control volume case, where full upstream weighting is always applied when the switch upstrvel is true.

5.8.6 Solute Definition

These instruction can be used to add a new solute (ie. species) to the system. **HydroSphere** is able to handle more than one solute per simulation, and straight and branching decay chains are also supported. An example of a straight decay chain is the following system:

$$\text{Uranium}^{234} \rightarrow \text{Thorium}^{230} \rightarrow \text{Radium}^{226}$$

which indicates that the decay of the radioactive isotope Uranium^{234} produces the daughter product Thorium^{230}, which in turn decays to form Radium^{226}. For an example, see verification problem PM_CD **rgm cross-reference**

Branching decay chains can have a single isotope which decays into one or more daughter products, or daughter products which have one or more parents.

Solute

Instructs **grok** to begin reading instructions which describe the behaviour of the new

solute. The following instructions are all optional, with the exception of End, which must be present to signal the end of the instruction list for the new solute.

Name

If this instruction is not present in the input block, then the default name **Species n**, where n is the current solute number, will be assigned. It requires the following input:

1. **spname** Species name.

Free-solution diffusion coefficient

If this instruction is not present in the input block, then a default value of 0.0 (zero) will be assigned to the solute free-solution diffusion coefficient. It requires the following input:

1. **diffrac** Free solution diffusion coefficient [L^2/T].

Parents

If this instruction is not present in the input block, then a default value of 0 (zero) will be assigned to be the number of parents for this solute. It requires the following input:

1. **npa** Number of parent species for the current species. If the current species has one or more parents, enter the following two instruction npa times (ie. once for each parent):
   (a) **kparen(i)** List of all the NPA parents for the current species.
   (b) **aparen(i)** Mass fraction of the parent species (between 0.0 and 1.0) transforming into the daughter species (i.e. the current species).

Decay constant

If either this instruction or Zoned decay constant (below) are not present in the input block, then a default value of 0.0 (no decay) will be assigned to the solute first-order decay constant. It requires the following input:

1. **clambda** First-order decay constant [1/T]. This decay constant will be used in all material zones in the domain.

Zoned decay constant

It requires the following input:

1. **clambda(i,j),j=1,nzones_prop** First-order decay constant [1/T] for species i for each zone j.

Distribution coefficient

If either this instruction or Zoned distribution coefficient (below) are not present in the

input block, then a default value of 0.0 (no attenuation) will be assigned to the solute distribution coefficient. It requires the following input:

1. **dkd** Distribution coefficient, which will be used in all material zones in the domain.

Zoned distribution coefficient

It requires the following input:

1. **dkd(i,j),j=1,nzones_prop** Distribution coefficient for species i for each zone j.

Fracture retardation factor

If either this instruction or Zoned fracture retardation factor (below) are not present in the input block, then a default value of 1.0 (no attenuation) will be assigned to the solute fracture retardation factor. It requires the following input:

1. **rfrac** Fracture retardation factor, which will be used in

Zoned fracture retardation factor

It requires the following input:

1. **rfrac(i,j),j=1,nzones_prop** Retardation factor for species i for each zone j.

End solute

This instruction signals the end of the solute parameter input, and control is then passed back to the pre-processor.

Note that instructions decay constant and zoned decay constant are mutually exclusive for a given solute, and should not appear in the same Solute ... End solute block. This also applies to distrubution coefficient and fracture retardation factor defininitions. You can however, define a solute with decay or attenuation properties which are uniform throughout the domain while a second solute has a zoned behaviour.

Since a new species is created each time the instruction solute is used, any instructions which depend on it should be placed after it in the `.grok` file. Currently, this applies to the instructions make fractures, specified concentration and specified third-type concentration.

The following simple example shows how to define a single, conservative, non-decaying solute called 'Species 1' with a free-solution diffusion coefficient of (0.0) zero:

```
Solute
End solute
```

Here is an example from a more complex system with two solutes and 7 material zones. Figure 5.3 shows how to define a solute, called DCB, which only decays in zone 1, and has distribution coefficients which vary from zone to zone. Figure 5.4 shows how to define a second solute, called BAM, which is a daughter product of DCB, and does not decay. This solute has the same zoned distribution coefficients as the first solute.

```
solute

    name
    DCB

    free-solution diffusion coefficient
    3.689e-5            ! free solution diffusion coefficient (m2/d)

    zoned decay constant
    0.693           ! 1  first-order decay constant (1/d)
    0.0             ! 2
    0.0             ! 3
    0.0             ! 4
    0.0             ! 5
    0.0             ! 6
    0.0             ! 7

    zoned distribution coefficient
    0.0005              ! 1   distribution coefficient (kg/m3)
    0.0005              ! 2
    0.0005              ! 3
    0.0013              ! 4
    0.005               ! 5
    0.014               ! 6
    0.020               ! 7

end solute
```

Figure 5.3: Definition of a parent solute with zoned properties

```
solute

    name
    BAM

    free-solution diffusion coefficient
    3.7295e-5           ! free solution diffusion coefficient (m2/d)

    parents
    1               ! ie. DCB
    ! parent #          ! mass ratio
    !==========      =============
        1                 1.0

    decay constant
    0.0             ! first-order decay constant (1/d)

    zoned distribution coefficient
    0.0005              ! 1   distribution coefficient (kg/m3)
    0.0005              ! 2
    0.0005              ! 3
    0.0013              ! 4
    0.005               ! 5
    0.014               ! 6
    0.020               ! 7

end solute
```

Figure 5.4: Definition of a daughter solute with zoned properties

5.8.7 Input/Output Options

The following instructions affect the I/O for the transport solution.

Echo transport boundary conditions

Causes the current boundary conditions to be written to the `o.eco` file.

Set kpconc

Assign a value to the variable KPCONC, which controls the output of nodal concentrations in binary format to a file which will automatically be assigned the suffix `o.c01`. It requires the following input:

1. **kpconc** Set KPCONC equal to 0 if no output of concentrations is desired. Set KPCONC equal to 1 to output concentrations at the output times specified in instruction output times. The default value is 1.

If a dual-porosity medium is being simulated, then concentration in the immobile zone will be written to output in binary format to a file which will automatically be assigned the suffix `o.i01`.

Set kwrithc

Assigns a value to the variable KWRITHC, which controls whether concentration values for the last timestep are output in binary format. This enables the use of the last time step concentrations as initial conditions for a subsequent simulation, for example, if one wants to carry on the simulation further in time without restarting from time zero. The file will automatically be assigned the suffix `o.cen`. It requires the following input:

1. **kwrithc** Set KWRITHC equal to 1 to write the concentration values for the last time step to a file. Set KWRITHC equal to 0 if no output of final concentrations is desired. The default value is 0.

Flux output nodes

This instruction allows you to generate detailed mass flux information at specific nodes for each timestep. It requires the following input:

1. **new_noutfc** The number of new output nodes desired. Read the following new_noutfc times:
   (a) **ioutfc(i)** Flux output node number. These values should be entered one per line.

Specifying flux output nodes causes **HydroSphere** to create a file called `prefixo.flm`. For each timestep in the transport solution, one line per flux output node per species will be written to the file which contains the species number, node number, time, concentration,

mass flux and nodal coordinates. Such output can be imported into an editor (e.g. Microsoft Excel) and sorted by column to facilitate, for example, the creation of a plot of concentration or flux versus time for a given species at a node.

For an example which uses flux output nodes, see verification problem PM_CD **rgm cross-reference**

5.8.7.1 Observation wells and points

The observation well data serves the purpose of outputting the computed concentration along an imaginary passive well in the domain. It does not affect the results of the computation.

Make observation well

Chooses a continuous set of element face edges and makes them into line elements which act as an observation well. The line elements are chosen to fall on or close to a line defined by the user.

It requires the following input:

1. **x1, y1, z1** The coordinates of one end of the well, which will be located at the node closest to this point.
2. **x2, y2, z2** The coordinates of the other end of the well, which will be located at the node closest to this point. The set of element edges which form the shortest path between the two nodes will become 1D line elements forming the well.

Make observation point

Chooses a node and makes it into an observation point. It requires the following input:

1. **x1, y1, z1** The coordinates of the observation point. The node closest to this point will be chosen as the observation point.

Using either of these instructions causes **HydroSphere** to create a file called `prefixo.obc`. For each timestep in the flow solution, one line per observation well node or observation point per species will be written to the file. Each line contains the well number, node number, time, concentration and nodal coordinates. Such output can be imported into an editor (e.g. Microsoft Excel) and sorted by column to facilitate, for example, the creation of a plot of fluid flux versus time at a node.

5.8.7.2 Solute mass balance

By default, solute mass balance information for each species is computed at each time step and written to the `prefixo.lst` file.

A summary is also written to the file `prefixo.mbg`. For each timestep in the transport solution, one line per species will be written to the file which contains the species number, time, total mass in the system ($Tmass$), change in mass due to sources and sinks ($dBoundary$), change in mass stored ($dStored$), error ($dBoundary - dStored$) and normalized error ($(dBoundary - dStored)/Tmass * 100$)) Such output can be imported into an editor (e.g. Microsoft Excel) and sorted by column to facilitate, for example, the creation of a plot of mass balance error versus time for a given species.

For an example see verification problem PM_CD **rgm cross-reference**.

No solute mass balance

This instruction prevents solute mass balance information from being computed or written.

Set kpmasbc

Assigns a value to the variable KPMASBC, which controls whether solute mass balance information is output to a file which will automatically be assigned the suffix `o.bal`. It requires the following input:

1. **kpmasbc** Set KPMASBC equal to 1 to write the solute mass balance information to a file. Set KPMASBC equal to 0 if no output of solute mass balance information is desired. The default value is 1.

5.8.7.3 Flux-averaged concentration at a well

Using either of the instructions Make well or Make infilled well causes **HydroSphere** to create a file called `prefixo.wco`. For each timestep in the flow solution, one line per well per species will be written to the file. Each line contains the species number, well number, time, flux-averaged concentration and nodal coordinates of the first node in the well. Such output can be imported into an editor (e.g. Microsoft Excel) and sorted by column to facilitate, for example, the creation of a plot of flux-averaged concentration versus time at a well.

5.8.8 Timestep control

Most of the instructions discussed in section 5.6.8.1 apply as well to the transport simulation and will not be repeated here. Here are some instructions which apply specifically to the transport simulation:

Transport time weighting

Assigns a new value for the time-weighting factor for the transport solution. The default is 0.5, which is for central (ie. Crank-Nicholson) time-weighting. A value of 1.0 would be specified for fully implicit time-weighting. Fully implicit time-weighting is less prone to exhibit oscillations but more prone to numerical smearing than central time-weighting. It requires the following input:

1. **twc** The desired time-weighting factor for the transport solution. Values must be greater than 0.0 and less than or equal to 1.0.

Peclet number

Assigns a new value for the Peclet number used to generate warning messages. This value does not influence the solution in any way, but is merely present for the convenience of the user. The default is 1×10^{20}. This large value suppresses the generation of warning messages. It requires the following input:

1. **pectol** The desired number.

Courant number

Assigns a new value for the maximum Courant number used to generate warning messages. This value does not influence the solution in any way, but is merely present for the convenience of the user. The default is 1×10^{20}. This large value suppresses the generation of warning messages. It requires the following input:

1. **courtol** The desired number.

If the simulation is to use the adaptive timestep procedure, then the following instruction causes the model to modify the timestep values as the solution proceeds, based on the transient transport behaviour.

Concentration control

Changes the way the transient behaviour of the concentration can be used to control the adaptive timestep procedure. It requires the following input:

1. **control_conc, dconc_allowed, dconc_min_allowed** You can specify true or false for the logical variable CONTROL_CONC which, if true, causes the maximum change in concentration for one time step to be used to determine the next time step size. By default, CONTROL_CONC is set to false.

   The variable DCONC_ALLOWED is the maximum allowed *percent* change in nodal conc during any time step, and is used to determine the next time step size [L]. The percent change is calculated based on the range of the concentration function at the start of the time step, including first-type nodes. The default is 5.0 percent.

   The variable DCONC_MIN_ALLOWED specifies an *absolute* lower limit on the concentration changes which will result in time step reduction. This eliminates any uneccessary reduction in the timestep if the range in the concentration function becomes very small. In such cases the percent change could be greater than DCONCV_ALLOWED while the absolute change in concentration was negligible. The default is 0.01.

5.8.9 Finite-difference options

The following instructions affect the transport simulation in a general way when finite difference method is being used.

Compute fd cross terms

Causes the model to compute the dispersive cross terms explicitly when a finite difference representation is chosen. By default, the dispersive cross terms are ignored when a finite difference representation is used.

Control volume

Causes the control-volume finite difference approach (as opposed to a standard finite difference approach) to be used. This instruction has no effect when the finite element approach is being used because the control volume approach is always used for the finite element flow solution.

5.8.10 Density-dependent flow and transport solution

This instruction should only be used for fully-saturated flow conditions in which there are NO fractures present. The following instruction can be used to enable a density-dependent flow and transport solution:

Density-dependent transport

1. **rhomax** The maximum fluid density (which corresponds to the maximum solute concentration).
2. **toldens** The minimum absolute difference in head and concentration for convergence of the non linear Picard loop.
3. **maxitdens** The maximum number of iterations for the Picard loop. If the maximum number of iterations is reached with obtaining convergence, the time step is cut in half and the loop is started over.

Note that you must set up the problem with *RELATIVE CONCENTRATIONS*. Therefore, rhomax is the fluid density for a relative concentration equal to 1.

As an example:

```
1200.0   rhomax
0.02     tolerance
100      maximum iterations
```

In this case, the maximum density is 1200.0 kg/m^3, and the reference density is 1000 kg/m^3, the tolerance on absolute head and concentration change to terminate the non-linear loop is equal to 0.02, and the maximum number of non linear iterations is 100.

Chapter 6

Mathematical Notation

The following general rules can be applied throughout this document:

- Subscripts i and j are used to denote a property of node i or j respectively.
- Subscript *par* is used to denote a property of the parent species in the case of a decay chain.
- Symbol$\hat{}$is used to denote an approximating function e.g. $\hat{h}$ for approximate hydraulic head.
- Superscript L denotes a time level in a time stepping procedure.
- Superscript r denotes an iteration level in an iterative procedure.

a	Porous medium-macropore coupling, distance from block centre to fracture [L].
A	Tile drain, cross-sectional area in the wetted portion [L^2].
B	Boundary of finite-element volume v [L].
C	Porous medium, solute concentration [M L^{-3}].
	Variants:
	C_d Dual continuum
	C_{Imm} Double-porosity immobile region
	C_f Fracture
	C_o Overland flow
	C_t Tile drain
	$C_{t_{\text{Inj}}}$ Tile drain, injected water
	C_w Well
	$C_{w_{\text{Inj}}}$ Well, injected water
C_c	Chezy coefficient [$L^{1/2}$ T^{-1}].
C_{dwn}	Concentration of downstream node between i and j [M L^{-3}].
	Variants:
	C_{ups} Upstream node

C_{i2ups} Second upstream node

C_L A constant which depends on rainfall intensity r [dimensionless].

C_x Chezy coefficient in the $x-$direction [$L^{1/2}$ T^{-1}].

C_y Chezy coefficient in the $y-$direction [$L^{1/2}$ T^{-1}].

$\mathbf{D}$ Porous medium, hydrodynamic dispersion tensor [L^2 T^{-1}].

Variants:

$\mathbf{D_d}$ Dual continuum

$\mathbf{D_f}$ Fracture

$\mathbf{D_o}$ Overland flow

D_{free} Solute, free-solution diffusion coefficient [L^2 T^{-1}].

D^*_{Imm} Double-porosity, effective diffusion coefficient in the immobile region [L^2 T^{-1}].

d_o Overland flow, water depth [L].

D_t Tile drain, dispersion coefficient [L^2 T^{-1}].

D_w Well, dispersion coefficient [L^2 T^{-1}].

ET_S Surface water, evapotranspiration [L^3 T^{-1}].

ET_G Subsurface water, evapotranspiration [L^3 T^{-1}].

F Jacobian matrix.

f_s Fracture, spacing [L].

f_x Darcy-Weisbach friction factor in the $x-$direction [dimensionless].

f_y Darcy-Weisbach friction factor in the $y-$direction [dimensionless].

g Gravitational acceleration [L T^{-2}].

h Porous medium, hydraulic head [L].

h_d Dual continuum

h_f Fracture

h_o Overland flow, water surface elevation

h_t Tile drain

h_w Well

H_d Overland flow, depression storage height [L].

H_o Overland flow, obstruction storage height [L].

H_s Overland flow, maximum height over which area covered by surface water goes from 0 to unity [L].

I Net infiltration [L^3 T^{-1}].

IT_{max} The maximum number of iterations allowed during a single time level.

$\mathbf{I}$ Identity tensor.

$\mathbf{k}$ Porous medium, permeability tensor [L^2]

$\mathbf{K}$ Porous medium, saturated hydraulic conductivity tensor [L T^{-1}].

K^* Component of hydraulic conductivity tensor normal to a seepage face.

K' Porous medium, equilibrium distribution coefficient [L^{-3} M].

Variants:

K'_d Dual continuum

K'_f Fracture

K_a Porous medium-macropore coupling, interface hydraulic conductivity [L T^{-1}].

$\mathbf{k_d}$ Dual continuum, permeability tensor [L^2]

$\mathbf{K_d}$ Dual continuum, saturated hydraulic conductivity tensor [L T^{-1}].

$\mathbf{K_f}$ Fracture, saturated hydraulic conductivity tensor[L T^{-1}].

$\mathbf{K_o}$	Overland flow, conductance tensor[L T^{-1}].
k_r	Porous medium, relative permeability [dimensionless].
	Variants:
	k_{rd} Dual continuum
	k_{rf} Fracture
	k_{ro} Overland flow
	k_{rt} Tile drain
	k_{rw} Well
k_{ra}	Porous medium-macropore coupling, interface relative permeability [dimensionless].
k_{rso}	Porous medium-overland flow coupling, rill effect term [dimensionless].
K_s	Overland flow, conductance term reduction factor [dimensionless].
K_{so}	Porous medium-overland flow coupling, leakance term [??].
K_t	Tile drain, hydraulic conductivity [L T^{-1}].
K_w	Well, saturated hydraulic conductivity [L T^{-1}].
K_{ox}	Overland flow, conductance in $x-$ direction [L T^{-1}].
K_{oy}	Overland flow, conductance in $y-$ direction [L T^{-1}].
l	Well or tile drain, length coordinate along the axis [L].
l'	Location at which specified well or tile discharge (or recharge) is applied [L].
L_s	Well, screen length [L].
N	Finite element basis function [dimensionless].
n	Manning roughness coefficient [$TL^{-1/3}$].
n^*	Brooks-Corey exponent equal to $2 + 3\lambda^*$ [dimensionless].
n_x	Manning roughness coefficient in the $x-$direction [$L^{-1/3}$ T].
n_y	Manning roughness coefficient in the $y-$direction [$L^{-1/3}$ T].
P	Net precipitation [L^3 T^{-1}].
$\mathbf{q}$	Porous medium, fluid flux [L T^{-1}].
	Variants:
	$\mathbf{q_d}$ Dual continuum
	$\mathbf{q_f}$ Fracture
	$\mathbf{q_o}$ Overland flow
	$\mathbf{q_t}$ Tile drain
	$\mathbf{q_w}$ Well
P_c	Capillary pressure head [L]
P_{dwn}	Position vector of downstream node between i and j.
	Variants:
	P_{ups} Upstream node
	P_{i2ups} Second upstream node
P_t	Tile drain, wetted perimeter of channel [L].
Q	Porous medium, fluid source or sink [T^{-1}]
Q_c	Porous medium, solute source or sink [T^{-1}].
	rgm Rene, not mentioned in text see also dual, frac etc
Q_{cd}	Dual continuum, solute source or sink [T^{-1}].
	rgm Rene, not mentioned in text see also dual, frac etc
Q_d	Dual continuum, fluid source or sink [T^{-1}]

Q_G^W	Subsurface water, withdrawal [L^3 T^{-1}].
Q_{G1}	Subsurface water, inflow [L^3 T^{-1}].
Q_{G2}	Subsurface water, outflow [L^3 T^{-1}].
Q_{GS}	Surface/subsurface water interactive flow [L^3 T^{-1}].
Q_o	Overland flow, volumetric flow rate per unit area representing external source and sinks [L T^{-1}].
Q_S^W	Surface water, withdrawal [L^3 T^{-1}].
Q_{S1}	Surface water, inflow [L^3 T^{-1}].
Q_{S2}	Surface water, outflow [L^3 T^{-1}].
Q_t	Tile drain, specified fluid flow rate in or out [L^3 T^{-1}], applied at location l'.
Q_w	Well, discharge (or recharge) per unit length [L^2 T^{-1}] applied at location l'.
r	Rainfall intensity [L T^{-1}].
R	Porous medium, retardation factor [dimensionless]. Variants: R_d Dual continuum; R_f Fracture; R_o Overland flow; R_t Tile drain
r_0	Double porosity, radius of a representative sphere [L].
r_c	Well, casing radius [L].
r_s	Well, screen radius [L].
Re_i	Reynolds number in coordinate direction i.
s	Overland flow, coordinate along direction of maximum ground surface slope [L].
S_e	Effective saturation [dimensionless].
S_{fx}	Overland flow, friction slope in the $x-$direction [dimensionless].
S_{fy}	Overland flow, friction slope in the $y-$direction [dimensionless].
S_{ox}	Overland flow, bed slope in the $x-$direction [dimensionless].
S_{oy}	Overland flow, bed slope in the $y-$direction [dimensionless].
S_s	Porous medium, specific storage [L^{-1}].
S_{sf}	Fracture, specific storage [L^{-1}].
S_w	Porous medium, water saturation [dimensionless]. Variants: S_{wd} Dual continuum; S_{wf} Fracture; S_{wt} Tile drain; S_{ww} Well
S_{wmax}	The maximum change in water saturation allowed during a single time-step.
S_{wr}	Residual water saturation [dimensionless].
t	Time [T].
tol_b, tol_f	Switching parameters for primary variable substitution [dimensionless].
v	Region or control volume associated with a node [L^3].
V	Volume of the finite-element domain [L^3].
$\overline{v}_{io}$	Overland flow, vertically averaged flow velocity in the coordinate direction i [L T^{-1}].
$\overline{v}_{xo}$	Overland flow, vertically averaged flow velocity in the $x-$direction [L T^{-1}].

$\overline{v}_{yo}$	Overland flow, vertically averaged flow velocity in the $y-$direction [L T^{-1}].
w_d	Dual continuum, volumetric fraction of the total porosity [dimensionless].
w_f	Fracture, aperture or width [L].
w_m	Porous medium, volumetric fraction of the total porosity [dimensionless].
W	A large number e.g. 10^{20}.
x, y, z	Global Cartesian coordinates [L].
x_i	Component of the Cartesian coordinate system [L].
x_r	Mass ratio of a unit volume of solid phase to that in the water phase, [dimensionless]
z	Porous medium, elevation head [L].
	Variants:
	z_d Dual continuum
	z_f Fracture
	z_t Tile drain
	z_w Well
z_o	Overland flow, bed (land surface) elevation [L].
α	Van Genuchten parameter [L^{-1}].
α_{Imm}	Double-porosity, first-order mass transfer coefficient between the mobile and immobile regions [L^{-1}].
α_l	Porous medium, longitudinal dispersivity [L].
	Variants:
	α_{ld} Dual continuum
α_t	Porous medium, transverse dispersivity [L].
	Variants:
	α_{td} Dual continuum
α_w	Water, compressibility [L T^2 M^{-1}].
α_{wd}	Porous medium-macropore coupling, first-order transfer coefficient for water [L^{-1} T^{-1}].
α^*_{wd}	Porous medium-macropore coupling, geometric factor [L^{-2}].
β	Van Genuchten parameter [dimensionless].
β_d	Porous medium-macropore coupling, geometrical factor [dimensionless].
γ	Water, kinematic viscosity [L^2 T^{-1}].
γ_{ij}	Term describing fluid flow between nodes i and j. [??]
Γ_{ex}	Porous medium, fluid exchange rate with all other domains [T^{-1}]
Γ_d	Dual continnum, fluid exchange rate with subsurface domain [T^{-1}]
	Variants:
	Γ_f Fracture
	Γ_o Overland flow
	Γ_t Tile drain
	Γ_w Well
γ_w	Porous medium-macropore coupling, empirical constant [dimensionless].
δ	Dirac delta function.
ΔS_G	Subsurface, change in water storage. [L^3]
ΔS_S	Overland flow, change in water storage. [L^3]
Δt	Time step [T].
ϵ	A small numerical shift in the pressure head value used in the Newton-

	Raphson method.
η_i	Porous medium, a set of nodes connected to node i.
	Variants:
	η_{di} Dual continuum
	η_{fi} Fracture
	η_{wi} Well
	η_{ti} Tile drain
θ	Porous medium, water content [dimensionless].
θ_{Imm}	Double-porosity, porosity of the immobile region [dimensionless]
θ_s	Porous medium, saturated water content [dimensionless].
	Variants:
	θ_{sd} Dual continuum
λ	Porous medium, solute first-order decay constant [L^{-1}].
	Variants:
	λ_d Dual continuum
	λ_f Fracture
	λ_o Overland flow
	λ_t Tile drain
	λ_w Well
λ^*	Brooks-Corey pore-size index [dimensionless].
μ	Water, viscosity [M L^{-1} T^{-1}].
ν	Van Genuchten parameter equal to $1 - \frac{1}{\beta}$ [dimensionless]
π	The constant $3.1415\cdots$. Would you prefer apple or cherry? Ice cream with that?
ψ	Porous medium, pressure head [L].
	Variants:
	ψ_d Dual continuum
	ψ_f Fracture
	ψ_t Tile drain
	ψ_w Well
ψ_{atm}	Atmospheric pressure [L].
ψ_b	Brooks-Corey **rgm need this??** [L].
ψ_b	An assigned pressure head. [L]
ρ	Water, density [M L^{-3}].
ρ_b	Porous medium, bulk density [M L^{-3}].
	Variants:
	ρ_{bd} Dual continuum
$\sigma(r)$	Van Leer flux limiter [dimensionless] with smoothness sensor r.
τ	Porous medium, matrix tortuosity [dimensionless].
	Variants:
	τ_d Dual continuum
ϕ_o	Overland flow, surface porosity [dimensionless].
Ω_{ex}	Porous medium, solute exchange rate with all other domains [T^{-1}]
Ω_d	Dual continuum, solute exchange rate with subsurface domain [T^{-1}]
	Variants:

Ω_{Imm}	Double-porosity immobile zone
Ω_f	Fracture
Ω_o	Overland flow
Ω_t	Tile drain
Ω_w	Well

$\overline{\nabla}$ One-dimensional gradient operator.

$\overline{\overline{\nabla}}$ Two-dimensional gradient operator.

∇ Three-dimensional gradient operator.

Chapter 7

References

Abdul, A.S., 1985. Experimental and Numerical studies of the effect of the capillary fringe on streamflow generation, Ph.D. Thesis, University of Waterloo, Waterloo, Ontario, Canada, 210 pp.

Akan, A.O. and B.C. Yen, 1981. Mathematical Model of shallow water flow over porous media, Journal of Hydrology, Division of ASCE, H14, 479–494.

Bear, J., 1972. Dynamics of fluids in porous media, American Elsevier, New York, NY, 764 pp.

Behie, G.A., and P.A. Forsyth, 1984. Incomplete factorization methods for fully implicit simulation of enhanced oil recovery, SIAM J. Sci. Stat. Comput., 5(3), 543–561.

Berkowitz, B., J. Bear, and C. Braester, 1988. Continuum models for contaminant transport in fractured porous formations, Water Resour. Res., 24(8), 1225–1236.

Beven, K.J., 1985. Distributed Models in Hydrological Forecasting, edited by M.G. Anderson, and T.P. Burt, John Wiley, NY, 425–435.

Brooks, R.J., and A. T. Corey, 1964. Hydraulic properties of porous media. Hydrology paper 3, Colorado State university, Fort Collins, CO.

Celia, M.A., E.T. Bouloutas, and R.L. Zarba, 1990. A general mass-conservative numerical solution for the unsaturated flow equation, Water Resour. Res., 26(7), 1483–1496.

Cooley, R.L., 1971. A finite difference method for unsteady flow in variably saturated porous media: Application to a single pumping well, Water Resour. Res., 7(6), 1607–1625.

Cooley, R.L., 1983. Some new procedures for numerical simulation of variably-saturated flow problems, Water Resour. Res., 19(5), 1271–1285.

diGiammarco, P., E. Todini, and P. Lamberti, 1996. A conservative finite element approach to overland flow: the control volume finite element formulation, Journal of Hydrology, 175, 267–291.

Forsyth, P.A., and P.H. Sammon, 1986. Practical considerations for adaptive implicit methods in reservoir simulation, J. Comput. Phys. 62, 265–281.

Forsyth, P.A., 1988. Comparison of the single-phase and two-phase numerical formulation for saturated-unsaturated groundwater flow, Comput. Methods Appl. Mech. Engrg., 69, 243–259.

Forsyth, P.A., 1991. A control volume finite element approach to NAPL groundwater contamination, SIAM J. Sci. Stat. Comput., 12(5), 1029–1057.

Forsyth, P.A., and R.B. Simpson, 1991. A two phase, two component model for natural convection in a porous medium, Int. J. Num. Meth. Fluids, 12, 655–682.

Freeze, R.A., and J.A. Cherry, 1979. Groundwater, Prentice-Hall Inc., New Jersey.

Frind, E.O., 1982. Simulation of long-term transient density-dependent transport in groundwater contamination problems, Adv. Water Res., 5(2), 73–88.

Gelhar, L.W., and M.A. Collins, 1971. General analysis of longitudinal dispersion in nonuniform flow, Water Resources Research, 7 (6), 1511–1521.

Gerke, H.H., and M.T. van Genuchten, 1993. A dual-porosity model for simulating the preferential movement of water and solutes in structured porous media, Water Resour. Res., 29(2), 305–319.

Gottardi, G. and M. Venutelli, 1993. A Control-Volume finite-element model for two-dimensional overland flow, Adv. Water Res., 16, 277–284.

Govindaraju, R.S. and M.L. Kavvas., 1991. Dynamics of moving overland flows over infiltrating surfaces at hillslopes, Water Resour. Res., 27(8), 1885–1898.

Govindaraju, R.S., Jones, S.E., and M.L. Kavvas, 1988a. On the Diffusion Wave Model for Overland Flow 1. Solution for steep slopes, Water Resour. Res., 24(5), 734–744.

Govindaraju, R.S., Jones, S.E., and M.L. Kavvas, 1988b. On the Diffusion Wave Model for Overland Flow 2. Steady State Analysis, Water Resour. Res., 24(5), 745–754.

Hoopes, J.A., and D.R. Harleman, 1967. Wastewater recharge and dispersion in porous media, Journal of the Hydraulics Division, ASCE, 93 (HY5), 51–71.

Huyakorn, P.S., and G.F. Pinder, 1983. Computational Methods in Subsurface Flow, Academic Press, New York.

Huyakorn, P.S., S.D. Thomas, and B.M. Thompson, 1984. Techniques for making finite elements competitive in modeling flow in variably saturated porous media, Water Resour. Res., 20(8), 1099–1115.

Huyakorn, P.S., A.G. Kretschek, R.W. Broome, J.W. Mercer, and B.H. Lester, 1984b. Testing and validation of models for simulating solute transport in groundwater: development, evaluation and comparison of benchmark techniques. International Groundwater Modeling Center. HRI Report No. 35, Nov.

Huyakorn, P.S., Y.S. Wu and N.S. Park, 1994. An improved sharp-interface model for assessing NAPL contamination and remediation of groundwater systems, Journal of Contaminant Hydrology, 16, 203-234.

Huyakorn, P.S., E.P. Springer, V. Guvanasen, and T.D. Wadsworth, 1986. A three-dimensional finite-element model for simulating water flow in variably saturated porous media, Water Resour. Res., 22(13), 1790–1808.

Kropinski, M.C.A., 1990. Numerical techniques for saturated-unsaturated groundwater flow, MSc. thesis, University of Waterloo.

Lacombe, S., E.A. Sudicky, S.K. Frape and A.J.A. Unger, 1995. Influence ofleaky boreholes on cross-formational groundwater flow and contaminant transport, Water Resour. Res., 31(8), 1871–1882.

Millington, R.J., 1959. Gas diffusion in porous media, Science, 130, 100–102.

Millington, R.J. and J.P. Quirk, 1961. Permeability of porous solids, Trans. Faraday Society, 15, 1200-1207.

Milly, P.C.D., 1985. A mass-conservative procedure for time-stepping in models of unsaturated flow, Adv. Water Resour., 8, 32–36.

Mualem, Y., 1976. A new model to predict the hydraulic conductivity of unsaturated porous media, Water Resour. Res., 12, 513–522.

Neuman, S.P., 1973. Saturated-unsaturated seepage by finite elements, ASCE J. Hydraul. Div., 99(HY12), 2233–2251.

Nielsen, D.R., M.Th. van Genuchten, and J.W. Biggar, 1986. Water flow and solute transport processes in the unsaturated zone, Water Resour. Res., 22(9), 89S–108S.

Panday, S., P.S. Huyakorn, R. Therrien, and R.L. Nichols, 1993. Improved three-dimensional finite element techniques for field simulation of variably saturated flow and transport, J. Contam. Hydrol., 12, 3–33.

Pruess, K., and Y.W. Tsang, 1990. On two-phase relative permeability and capillary pressure of rough-walled rock fractures, Water Resour. Res., 26(9), 1915–1926.

Perlmutter, N.M., and M. Lieber, 1970. Dispersal of plating wastes and sewage contaminants in groundwater and surface water, South Famingdale-Massapequa area, Nassau County, New york, U.S. Geological Survey Water Supply paper 1879-G.

Rasmussen, T.C. and D.D. Evans, 1989. Fluid flow and solute transport modeling in three-dimensional networks of variably saturated discrete fractures, U.S. Nuclear Regulatory Commission, Report NUREG/CR-5239.

Reitsma, S, and B.H. Kueper, 1994. Laboratory measurement of capillary pressure-saturation relationships in a rock fracture, Water Resour. Res., 30(4), 865–878.

Robin, M.J.L., A.L. Gutjahr, E.A. Sudicky, and J.L. Wilson, 1993. Cross-correlated random field generation with the direct Fourier Transform method, Water Resour. Res., 29(7), 2385–2397.

Sammon, P.H., 1988. An analysis of upstream differencing, Soc. Pet. Engrg. J. Res. Engrg., 3, 1053–1056.

Sharika, U., S. Senarath, F.L. Ogdon, C.W. Downer and H.O. Sharif, 2000. On the Calibration and Verification of Two-Dimensional, Distributed, Hortonian, Continuous Watershed Models, Water Resour. Res., 36, 6, 1495–1510.

Singh, V. and S.M. Bhallamudi, 1998. Conjunctive surface-subsurface modeling of overland flow, Adv. Water Res., 21, 567–579.

Smith, R.E. and D.A. Woolhiser, 1971. Overland Flow on an Infiltrating Surface, Water Resour. Res., 7(4), 899–913.

Sudicky, E.A., 1990. The Laplace transform Galerkin technique for efficient time-continuous solution of solute transport in double-porosity media. Geoderma, 46, 209–232.

Sudicky, E.A., and R.G. McLaren, 1992. The Laplace transform Galerkin technique for large-scale simulation of mass transport in discretely- fractured porous formations, Water Resour. Res., 28(2), 499–514.

Sudicky, E.A., A.J.A. Unger and S. Lacombe, 1995. A noniterative technique for the direct implementation of well bore boundary conditions in three-dimensional heterogeneous formations, Water Resour. Res., 31(2), 411–415.

Tang, D.H., E.O. Frind, and E.A. Sudicky, 1981. Contaminant transport in fractured porous media: Analytical solution for a single fracture, Water Resour. Res., 17(3), 555–564.

Therrien, R., 1992. Three-dimensional analysis of variably-saturated flow and solute transport in discretely-fractured porous media, Ph.D. thesis, Univ. of Waterloo, Waterloo, Ontario, Canada.

Therrien, R., and E.A. Sudicky, 1996. Three-dimensional analysis of variably-saturated flow and solute transport in discretely-fractured porous media. J. Contam. Hydrol., 23(1-2), 1–44.

Therrien, R., and E.A. Sudicky, 2000. Well bore boundary conditions for variably-saturated flow modeling, Advances in Water Resources, 24, 195-201.

Unger, A.J.A., P.A. Forsyth and E.A. Sudicky, 1996. Variable spatial and temporal weighting schemes for use in multi-phase compositional problems, Adv. Water resour., 19(1), 1–27.

van Genuchten, M.Th., 1980. A closed-form equation equation for predicting the hydraulic conductivity of unsaturated soils, Soil Sci. Soc. Am. J., 44, 892–898.

van Leer, B., 1974. Towards the ultimate conservative difference scheme II Monotonicity and conservation combined in a second order scheme, J. Comp. Phys., 14, 361–370.

VanderKwaak, J., 1999. Numerical Simulation of Flow and Chemical Transport in Integrated Surface-Subsurface Hydrologic Systems. Ph.D. Thesis in Earth Sciences, University of Waterloo, Waterloo, Ontario, Canada, 217 pp.

Viessman, W. (Jr.) and G.L. Lewis, 1996, Introduction to Hydrology, 4th Edition, Harper Collins College Publisher, New York, 760 pp.

Wang, J.S.Y., and T.N. Narasimhan, 1985. Hydrologic mechanisms governing fluid flow in a partially saturated, fractured, porous medium, Water Resour. Res., 21(12), 1861–1874.

Wilson, J.L., and P.J. Miller, 1978. Two-dimensional plume in uniform groundwater flow, Journal of the Hydraulics Division, ASCE, 104 (HY4), 503–514.

Woolhiser, D.A., 1996. Search for Physically Based Runoff Model - A Hydrologic El Dorado?, J. Hydrol. Eng., 122, 122–129.

Woolhiser, D.A., R.E. Smith, and J.V. Giraldez, 1997. Effects of spatial variability of saturated hydraulic conductivity on Hortonian overland flow, Water Resour. Res., 32(3), 671–678.

Zheng, C., 1990. A modular three-dimensional transport model for simulation of advection, dispersion and chemical reactions of contaminants in groundwater systems, prepared for USEPA Kerr Environmental Research Laboratory, Ada, OK 74820.

Appendix A

GMS file formats

The following description is taken from the GMS Reference Manual, Version 1.1. GMS divides files into logical units called cards. The first component of each card is a short name which serves as an identifier. The rest of the line contains information associated with the card. Some cards can use multiple lines.

A.1 2D meshes (ie. slices)

The instructions Read slice and 2D mesh to gms read and write 2D mesh data in GMS format respectively. The portion of the 2D mesh file format recognized by **grok** is as follows:

```
MESH2D                          ! File type identifier
E3T id n1 n2 n3 mat             ! 3-node triangle
E4Q id n1 n2 n3 n4 mat          ! 4-node quadrilateral
ND id x y z                     ! Nodal coordinates
```

grok does not recognize the cards E6T (6-node triangles) and E8Q (8-node quadrilaterals).

The card types used in the 2D mesh file are as follows.

Card Type	MESH2D
Description	File type identifier. Must be on first line of file. No Fields.
Required	YES

Card Type	E3T		
Description	Defines a 3-node (linear) triangular element.		
Required	NO		
Format	E3T id n1 n2 n3 mat		
Sample	E3T 283 13 32 27 4		
Field	*Variable*	*Value*	*Description*
1	id	+	The id of the element.
2–4	n1–n3	+	The nodal incidences of the elements ordered counterclockwise.
5	mat	+	The material id of the element.

Card Type	E4Q		
Description	Defines a 4-node (linear) quadrilateral element.		
Required	NO		
Format	E4Q id n1 n2 n3 n4 mat		
Sample	E4Q 283 13 32 27 30 4		
Field	*Variable*	*Value*	*Description*
1	id	+	The id of the element.
2–5	n1–n4	+	The nodal incidences of the elements ordered counterclockwise.
6	mat	+	The material id of the element.

Card Type	ND		
Description	Defines the coordinates of a node.		
Required	NO		
Format	ND id x y z		
Sample	ND 84 120.4 380.3 5632.0		
Field	*Variable*	*Value*	*Description*
1	id	+	The id of the node.
2–4	x,y,z	±	The nodal coordinates.

A.2 Ascii or binary scalar data set files

The instruction Generate layers from slice reads a node data set in order to define a variable surface (usually an elevation for the z-coordinate) for the base of the 3D grid or the top of a layer. This file can be written in either GMS ascii or binary format. **grok** automatically determines which format was used and reads it in the appropriate manner.

The ascii file format recognized by **grok** is as follows:

```
SCALAR                          ! File type identifier
ND n                            ! Number of data values
STAT 0                          ! Never any status flags
TS time                         ! Time step of the following data
val_1                           ! Scalar data values
```

```
val_2
.
.
val_n
```

In this case, the value n should correspond to the number of nodes. Currently, although the cards STAT and TS are recognized, **grok** does not use them internally. You should not include status flag information in files to be read by **grok**. **grok** only reads one set of scalar data values per file.

The card types used in the ascii scalar data set file are as follows.

Card Type	SCALAR
Description	File type identifier. Must be on first line of file. No Fields.
Required	YES

Card Type	ND		
Description	Defines the number of data values per time step. This number should correspond to the total number of nodes in the 2D slice being used to generate the 3D mesh.		
Required	YES		
Format	ND n		
Sample	ND 4		
Field	*Variable*	*Value*	*Description*
1	n	+	The number of nodes per 2D slice.

Card Type	STAT		
Description	Specifies whether or not status flags will be included in the file.		
Required	YES		
Format	STAT i		
Sample	STAT 0		
Field	*Variable*	*Value*	*Description*
1	i	0	Always zero for **grok** (ie./ no status flags)

Card Type	TS		
Description	Defines a set of scalar values associated with a timestep.		
Required	YES		
Format	TS time val_1 val_2 . . val_n		
Sample	TS 0.0 34.5 74.3 48.3 72.9		
Field	*Variable*	*Value*	*Description*
1	time	±	The time step value. Not used by **grok**.
2–(n+1)	val	±	The scalar values for each item.

The binary file format recognized by **grok** is as follows. One scalar value is listed per node in the 2D slice, and the status flag should always be false (ie. 0).

Item	**Size**	**Description**
version	4 byte integer	Version = 1000 for scalar file.
n	4 byte integer	Number of items (ie. nodes in 2D slice).
status data	4 byte integer	0 (ie. no status flags).
SFLT	4 byte integer	The number of bytes that will be used in the remainder of the file for each floating point value. 4 for **grok**.
SFLG	4 byte integer	The number of bytes that will be used in the remainder of the file for each floating status flag. Read but not used.
times step i	SFLT real	Time corresponding to time step I. Not used by **grok**.
val_1	SFLT real	Scalar value for item 1.
val_2	SFLT real	Scalar value for item 2.
.		
.		
val_n	SFLT real	Scalar value for item n.

Appendix B

Grid Builder file formats

B.1 2D meshes (ie. slices)

The instruction Read grid builder slice reads 2D triangular-element mesh data in Grid builder format.

The file which contains the node coordinates for the 2D triangular mesh has the file extension `.xyc` assigned by GRID BUILDER. The following fortran code segment shows how this data is read:

```
integer*4 nn2d
real*4 x2d(maxnn2d), y2d(maxnn2d)
open(44,file=coorfile,status='unknown',form='unformatted')
read(44) nn2d
read(44) (x2d(i),y2d(i),i=1,nn2d)
```

where nn2d is the number of nodes in the 2D triangular grid and x2d and y2d are the x-, y-coordinates of the nodes.

The file which contains the element incidences for the 2D triangular mesh has the file extension `.in3` assigned by GRID BUILDER. The following fortran code segment shows how this data is read:

```
integer*4 ne2d, in2d(maxne2d,4)
open(44,file=incfile,status='unknown',form='unformatted')
read(44) ne2d
read(44) ((in2d(i,j),j=1,3),i=1,ne2d)
```

where ne2d is the number of elements in the 2D triangular grid and in2d is the array containing the list of node incidences for each element. The second dimension of array in2d is set to 4 to accomodate 2D slices made up of rectangular elements in a future release.

The file which contains the element incidences for the 2D triangular mesh has the file extension `.ean` assigned by GRID BUILDER. The following fortran code segment shows how this data is read:

```
integer*4 el_area2d(maxne2d)
```

```
open(44,file=eanfile,status='unknown',form='unformatted')
read(44) (el_area2d(i),i=1,ne2d)
```

where el_area2d is the array containing the element area numbers. The GRID BUILDER program can be used to generate 2D grids with multiple areas and it automatically assigns each element the appropriate area number. The pre-processor can use these numbers to assign material properties.

B.2 Scalar data set files

The file which contains the scalar values for the nodes in a 2D triangular mesh has the file extension `.n01, .n02` etc. assigned by GRID BUILDER. The following fortran code segment shows how this data is read:

```
integer*4 nn2d
real*4 nprop(maxnn2d)
character*50 dtitle
open(8,file=fname,status='unknown',form='unformatted')
read(8) dtitle
read(8) (nprop(j),j=1,nn2d)
```

where nprop is the array containing the scalar values for each node and dtitle is a character string identifying the data in the file fname.