HydroGeoSphere

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

R. Therrien, Université Laval
R.G. McLaren, University of Waterloo
E.A. Sudicky, University of Waterloo
S.M. Panday, AMEC Inc./University of Waterloo

DRAFT
October 26, 2010

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 surface 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 surface 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 **HydroGeo-Sphere**, 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 **HydroGeoSphere** 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 surface and subsurface flow domains such that solute concentrations are also solved for *simultaneously* at each timestep in both regimes. This makes **HydroGeoSphere** a unique and ideal tool to simulate the movement of water and solutes within watersheds in a realistic, physically-based manner.

HydroGeoSphere uses the control volume finite element approach to simulate coupled surface and subsurface flow and transport. Fully 3-D simulations of variably-saturated fractured or granular aquifers may be performed. **HydroGeoSphere** 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. **HydroGeoSphere** 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. **HydroGeoSphere** 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.

This manual describes the physical and mathematical concepts underlying **HydroGeo-Sphere** 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.

Acknowledgements

The authors would like to thank:

- Peter Forsyth of the University of Waterloo, for graciously allowing us the use of his matrix solver package.

- George Matanga and Lisa Gessford of USBR and Don Demarco of Hydrogeologic Inc. for their careful review of the manuscript.

- Thomas Graf and Young-Jin Park for their contributions to various sections of the code, including the variable-density, temperature and surface flow and transport options.

- The countless students, consultants and colleagues who, over the years, have provided us with valuable feedback and suggestions that have improved the code immeasurably.

- And last but not least, Peter Huyakorn of Hydrogeologic Inc. for many years of hospitality, support and friendly advice.

iv

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 surface 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, drainage water reuse 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 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 **HydroGeoSphere**, 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.

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

1.2   Integrated Hydrologic Model Conceptualization

HydroGeoSphere is based on a rigorous conceptualization of the hydrologic system comprising surface 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 surface and subsurface flow and mass transport equations simultaneously and provides complete water balance and solute budgets. Referring to Figure 1.1, the surface 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:

$$I = (Q_{G2} - Q_{G1}) + Q_{GS} + ET_G + Q_G^W + \Delta S_G/\Delta t \tag{1.2}$$
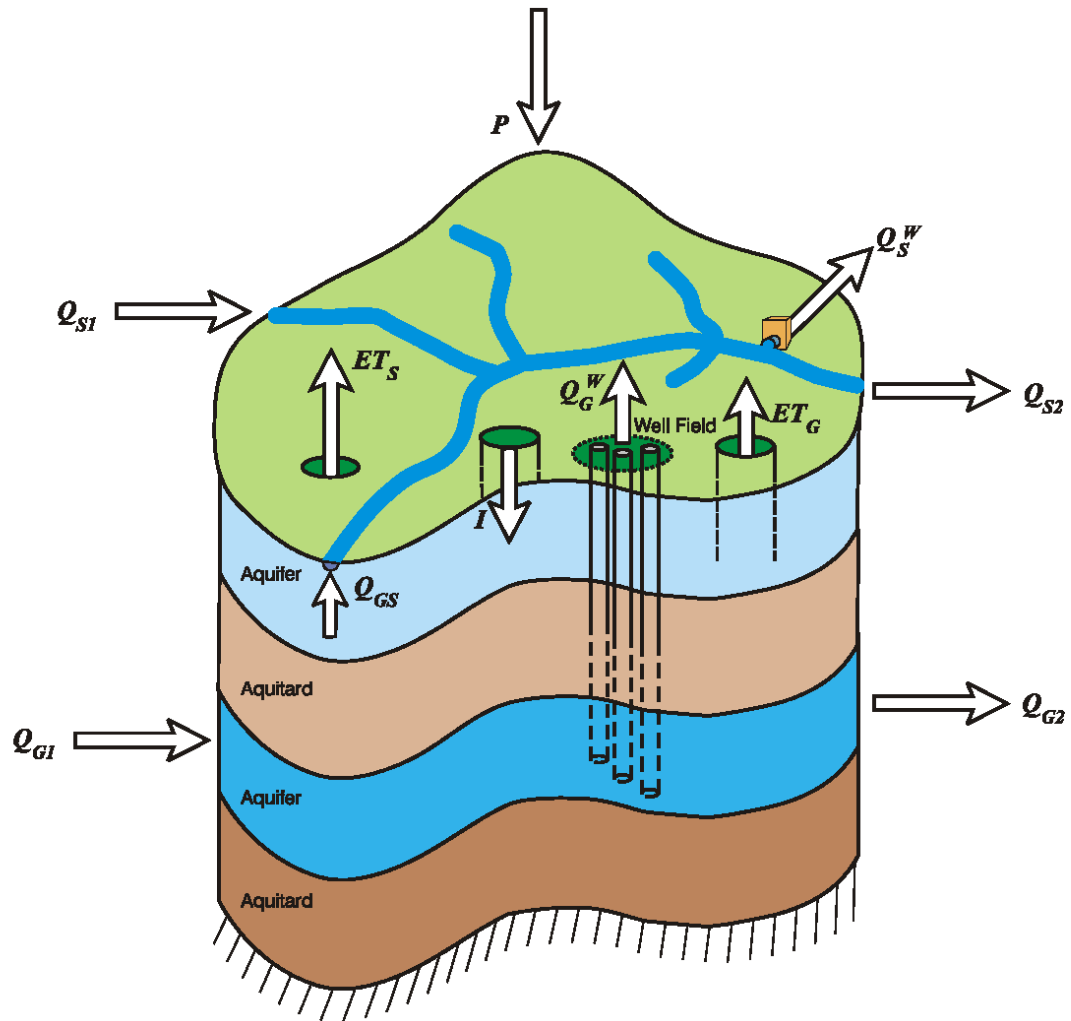
Figure 1.1: Regional Hydrologic Cycle [Adapted from *Viessman and Lewis*, 1996].

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 surface water inflow and outflow, Q_{GS} is the surface/subsurface water interactive flow, I is the net infiltration, ET_S is the evapotranspiration from the surface flow system, Q_S^W is the overland water withdrawal, ΔS_S is the surface 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, **HydroGeoSphere** utilizes a rigorous, mass conservative modeling approach that fully couples the surface 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

HydroGeoSphere 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) surface water flow and the Richards' equation governing 3-D unsaturated/saturated subsurface flow. For problems that also involve solute transport, the classical advection-dispersion equation is used in all domains.

The surface and subsurface flow and transport domains are discretized simultaneously as shown in Figure 1.2. The gridding options for 2-D 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 surface and subsurface domains consist of: (1) a single layer of surface 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 line elements, shown by squares, may represent surficial channels or subsurface tile-drains. The 2-D surface flow grid is draped over the subsurface 3-D mesh to maintain the areal topography of the land surface and to ensure that the nodes in the surface grid are coincident with those at the top of the subsurface mesh.

The 3-D 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 surface 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
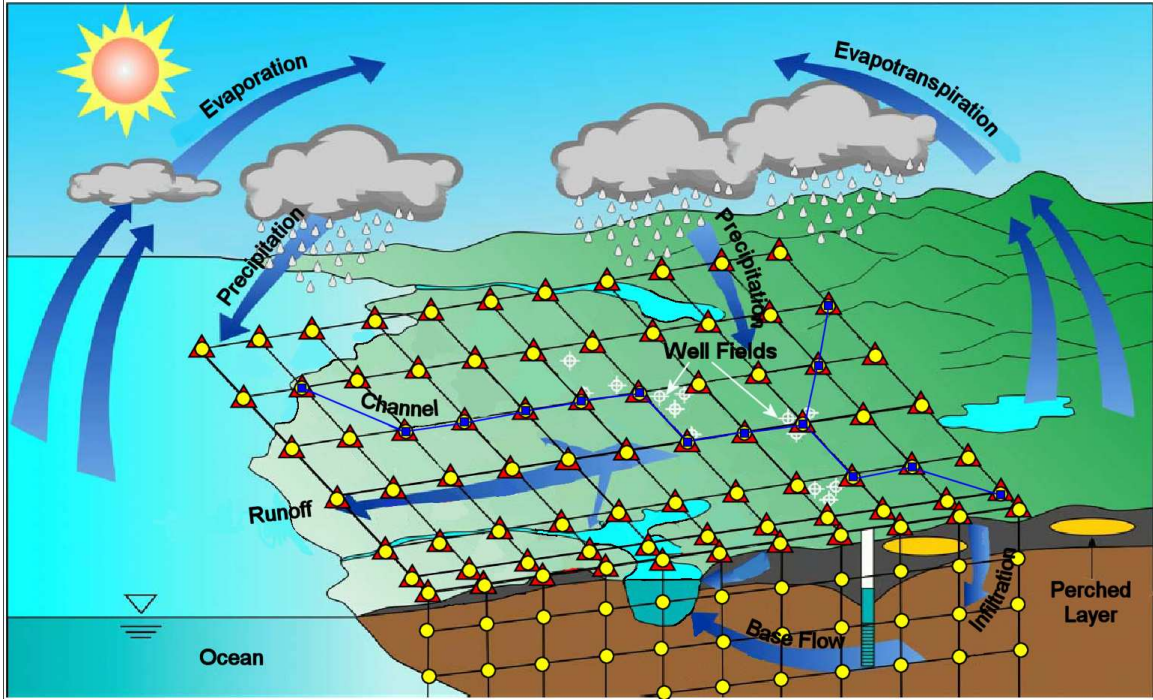
Figure 1.2: Integrated Numerical Simulation of Hydrologic System.

numerical superposition principle whereby the top layer of nodes represents both surface 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 HydroGeoSphere

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

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

Fluid flow

- Complete hydrologic cycle modeling using detailed physics of surface and subsurface flow in one integrated code. The surface regime can be represented as 2-D areal flow for the entire surface or as 2-D runoff into 1-D channels. The subsurface regime consists of 3-D unsaturated/saturated flow. Both regimes naturally interact with each other through considerations of the physics of flow between them.

- Physically-based accounting of all components of the hydrologic cycle water budget.

- 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.

- Accommodation of wellbore storage.

- Arbitrary combinations of porous, discretely-fractured, dual-porosity and dual-permeability media for the subsurface.

Mass transport

- Capability of modeling non-reactive and reactive chemical species transport in the associated surface and subsurface flow fields.

- Accurate handling of fluid and mass exchanges between fractures and matrix including matrix diffusion effects and solute advection in the matrix.

- Chain-reactions of radionuclide components.

Numerical methods

- 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.

- 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 surface water, 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, **HydroGeoSphere** 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 **HydroGeoSphere** include:

- Integrated water resource assessment.

- Watershed hydrologic analysis, including impacts of land-use or climate-change impacts on both surface and subsurface water.

- Floodplain hydrologic analysis.

- Fluvial hydraulic analysis.

- Contaminant migration and fate in both surface and subsurface water.

1.5 Operation and Input Options

HydroGeoSphere computational modules are built upon the widely popular FRAC3DVS code. Thus, the **HydroGeoSphere** conjunctive surface-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 **HydroGeoSphere** follow the original FRAC3DVS code. There are four steps involved in solving a given problem using **HydroGeoSphere**.

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 **HydroGeoSphere**.

3. Run **HydroGeoSphere** 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 that contain information for discretizing the problem domain, defining material properties for each element and specifying flow boundary conditions.

Simulation output pertaining to surface flow regime calculations is reported to the main **HydroGeoSphere** 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. These chapters and their purposes are outlined below.

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

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

Chapter 4 contains descriptions of several verification problems that were designed to test and demonstrate the capabilities of **HydroGeoSphere** 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, surface flow and solute transport problems.

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

Mathematical notation is summarized in Appendix A and Appendices F and G provide a summary of GMS and Grid Builder file formats, respectively.

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 **HydroGeoSphere** 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 **HydroGeoSphere**. This basic equation can be expanded to incorporate, among other features, discrete fractures, a second interacting porous continuum (e.g. fractures or macropores), wells, tile drains and surface flow. 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:

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

where w_m [dimensionless] is the volumetric fraction of the total porosity occupied by the porous medium (or primary continuum). This volumetric fraction is always 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. The fluid flux $\mathbf{q}$ [L T^{-1}] is given by:

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

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 [L^3 L^{-3} T^{-1}], which is a volumetric fluid 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:

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

In Equation 2.1, Γ_{ex} represents the volumetric fluid exchange rate [L^3 L^{-3} T^{-1}] between the subsurface domain and all other types of domains supported by the model and it is expressed per unit volume of the other domain types. Currently, these additional domains are surface, 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, ex=c, for the fracture, dual continuum, well, tile drain, surface and channel 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].

Using the Brooks and Corey functions, the saturation is given by:

$$\begin{aligned}
S_w &= \ S_{wr} + (1 - S_{wr}) \mid \alpha\psi \mid^{-\beta} && \text{for} \quad \psi < -1/\alpha \\
S_w &= \ 1 && \text{for} \quad \psi \geq -1/\alpha
\end{aligned} \tag{2.5}$$

and the relative permeability is obtained from:

$$k_r = S_e^{(2/\beta + l_p + 2)} \tag{2.6}$$

where α [L^{-1}] is the inverse of the air-entry pressure head, β [dimensionless] is the pore-size distribution index, l_p is the pore-connectivity parameter assumed equal to 2.0 in *Brooks and Corey* [1964], and S_e is often called an effective saturation, given by $S_e = (S_w - S_{wr})/(1 - S_{wr})$, with S_{wr} being the residual water saturation [dimensionless].

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

$$\begin{aligned} S_w &= S_{wr} + (1 - S_{wr}) \left[1 + \mid \alpha\psi \mid^\beta\right]^{-\nu} && \text{for} \quad \psi < 0 \\ S_w &= 1 && \text{for} \quad \psi \geq 0 \end{aligned} \tag{2.7}$$

with the relative permeability given by:

$$k_r = S_e^{(l_p)} \left[1 - \left(1 - S_e^{1/\nu}\right)^\nu\right]^2 \tag{2.8}$$

where

$$\left(\nu = 1 - \frac{1}{\beta}\right), \quad \beta > 1 \tag{2.9}$$

and where α and β are usually obtained from a fit of Equations 2.7 and 2.8 to experimental results. For his relative permeability model, [*Mualem*, 1976] has estimated that the pore-connectivity parameter l_p is equal to 0.5 for most soils.

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 **HydroGeoSphere**.

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 Equation 2.1 [*Cooley*, 1971; *Neuman*, 1973]:

$$\frac{\partial}{\partial t}\left(\theta_s S_w\right) \approx S_w S_s \frac{\partial \psi}{\partial t} + \theta_s \frac{\partial S_w}{\partial t} \tag{2.10}$$

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 Richard's Equation (Equation 2.1) for the porous matrix. With this extension, the governing two-dimensional flow equation in a fracture has the form:

$$-\overline{\overline{\nabla}} \cdot (w_f \mathbf{q_f}) - w_f \Gamma_f = w_f \frac{\partial S_{wf}}{\partial t} \tag{2.11}$$

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

$$\mathbf{q_f} = -\mathbf{K_f} \cdot k_{rf} \overline{\overline{\nabla}}(\psi_f + z_f) \tag{2.12}$$

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]:

$$\mathbf{K_f} = \frac{\rho g \, w_f^2}{12\mu} \tag{2.13}$$

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 **HydroGeoSphere**, relative permeability and saturation-pressure head relationships for the fractures are given by either the Brooks-Corey model (Equations 2.5 and 2.6), the Van Genuchten model (Equations 2.7 and 2.8) 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 be considered when describing flow in 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.

Using arguments similar to those invoked for the porous medium equation, the storage term in Equation 2.11 describing variably-saturated flow in the fractures becomes:

$$\frac{\partial}{\partial t} S_{wf} \approx S_{wf} S_{sf} \frac{\partial \psi_f}{\partial t} + \frac{\partial S_{wf}}{\partial t} \tag{2.14}$$

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:

$$S_{sf} = \rho g \alpha_w \tag{2.15}$$

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 **HydroGeoSphere** model can simulate variably-saturated fluid flow in a dual continuum based on the formulation presented by *Gerke and Van Genuchten* [1993]. The dual continuum formulation in **HydroGeoSphere** involves 2 separate continua, with the first continuum represented by the porous medium. We present here the formulation for the second continuum, which is linked to the porous medium continuum by a fluid exchange term. 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:

$$-\nabla \cdot (w_d \mathbf{q_d}) - \Gamma_d \pm Q_d = w_d \frac{\partial}{\partial t} (\theta_{sd} S_{wd}) \tag{2.16}$$

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

$$\mathbf{q_d} = -\mathbf{K_d} \cdot k_{rd} \nabla (\psi_d + z_d) \tag{2.17}$$

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 fluid flux per unit volume Q_d [L^3 L^{-3} 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:

$$\mathbf{K_d} = \frac{\rho g}{\mu}\mathbf{k_d} \tag{2.18}$$

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:

$$\mathbf{k_d} = \frac{w_f^2}{12} \tag{2.19}$$

For the second continuum the storage term in Equation 2.16 is expanded in a manner similar to Equation 2.10 for the porous medium, using the specific storage S_{sd} [L^{-1}] for the second continuum.

2.1.2.4   1D Hydromechanical coupling

The **HydroGeoSphere** code can be used to simulate transient flow affected by surface loading conditions. Examples of surface loading conditions include: advent or retreat of glaciation, erosion, and deposition.

Under hydromechanical equilibrium conditions, the governing equations are [*Neuzil*, 2003]:

$$\frac{\partial \sigma_{ij}}{\partial x_j} = 0 \tag{2.20a}$$

$$\sigma_{ij} = 2G\varepsilon_{ij} + 2G\frac{\nu}{1-2\nu}\varepsilon_{kk}\delta_{ij} + \alpha p\delta_{ij} \tag{2.20b}$$

$$\frac{\partial}{\partial x_i}\left(K_{ij}\frac{\partial h}{\partial x_j}\right) = S_{s3}\frac{\partial h}{\partial t} + \beta S_{s3}\frac{1}{\rho g}\frac{\partial \sigma_t}{\partial t} \pm Q \tag{2.21}$$

where σ_{ij} is the total stress [M T^{-2} L^{-1}] defined by Equation 2.20b, ε_{ij} is the mechanical strain [dimensionless], G is the shear modulus [M T^{-2} L^{-1}], ν is Poisson's ratio [dimensionless], α is the effective-stress hydroelastic constant [dimensionless], p is fluid pressure [M T^{-2} L^{-1}], S_{s3} is the three-dimensional specific storage coefficient [L^{-1}] of the porous medium originally defined in Equation 2.10, β is the three-dimensional loading efficiency [dimensionless], and σ_t is the mean normal total stress [M T^{-2} L^{-1}]. The loading efficiency is the fraction of stress that is transferred to fluid pressure. The ratio $\sigma_t/\rho g$ may be interpreted as equivalent fresh water height of the mean normal total stress at a given point. The three-dimensional loading efficiency, α, shear modulus, mean normal total stress, and specific storage are defined below:

$$\beta = \frac{\left(\frac{1}{K} - \frac{1}{K_s}\right)}{\left(\frac{1}{K} - \frac{1}{K_s}\right) + \left(\frac{\theta_s}{K_f} - \frac{\theta_s}{K_s}\right)} \tag{2.22}$$

$$\alpha = \left(1 - \frac{K}{K_s}\right) \tag{2.23}$$

$$G = \frac{E}{2\left(1 + \nu\right)} \tag{2.24}$$

$$\sigma_t = \frac{\sigma_{kk}}{3} \tag{2.25}$$

$$S_{s3} = \left(\frac{1}{K} - \frac{1}{K_s}\right) + \rho g \left(\frac{\theta_s}{K_f} - \frac{\theta_s}{K_s}\right) \tag{2.26}$$

where K is the bulk modulus of porous media [M T^{-2} L^{-1}], K_s is the solid bulk modulus [M T^{-2} L^{-1}], K_f is the fluid bulk modulus [M T^{-2} L^{-1}], and E is Young's elastic modulus [M T^{-2} L^{-1}]. The three-dimensional loading efficiency is the ratio of change in fluid pressure to change in mean total stress under undrained conditions. For highly compressible media, β approaches unity while in stiff media it can be zero [*Neuzil*, 2003]. As porosity approaches zero, K approaches K_s and the effects of water pressure on total stress (Equation 2.20b) and the effects of change of total stress on water storage (Equations 2.22 and 2.26) also vanish.

However, the current version of **HydroGeoSphere** does not have a mechanical or an equilibrium module that is coupled with the flow module. The mean normal total stress, which may be spatially distributed and vary with time, must be estimated externally to **HydroGeoSphere**. However, under the condition of purely vertical strain, the flow component can be decoupled or partially decoupled from the mechanical component.

Mechanical and hydraulic behaviour can also be partially coupled under the conditions of relatively homogeneous and areally extensive vertical loading. Under these conditions, Equation 2.21 is written as

$$\frac{\partial}{\partial x_i}\left(K_{ij}\frac{\partial h}{\partial x_j}\right) = S_{s1}\frac{\partial h}{\partial t} + \zeta S_{s1}\frac{1}{\rho g}\frac{\partial \sigma_{zz}}{\partial t} \pm Q \tag{2.27}$$

where σ_{zz} is the vertical total stress [M T^{-2} L^{-1}], S_{s1} is the modified one-dimensional specific storage [L^{-1}] [*Neuzil*, 2003] and ζ is the one-dimensional loading efficiency [dimensionless]. S_{s1} and ζ are defined below:

$$S_{s1} = \left(\frac{1}{K} - \frac{1}{K_s} \right) \left(1 - 2\alpha \frac{1-2\nu}{3(1-\nu)} \right) + \rho g \left(\frac{\theta_s}{K_f} - \frac{\theta_s}{K_s} \right) \tag{2.28a}$$

$$\zeta = \frac{\beta(1+\nu)}{3(1-\nu) - 2\alpha\beta(1-2\nu)} \tag{2.28b}$$

It can be seen that Equation 2.27 is a subset of Equation 2.21 which is more general.

Equations 2.20 to 2.26 are developed for isotropic materials, for transversely anisotropic materials (isotropic along horizontal planes and anisotropic in the vertical direction), β , σ_t, and S_{s3} in Equation 2.21 must be replaced by:

$$\beta' = \frac{\frac{1}{K_H} - \frac{1}{K_s}}{\left(\frac{2}{3K_H} - \frac{1}{3K_V} - \frac{1}{K_s} \right) + \left(\frac{\theta_s}{K_f} - \frac{\theta_s}{K_s} \right)} \tag{2.29}$$

$$\sigma_t' = \frac{\sigma_{xx}}{3} + \frac{\sigma_{yy}}{3} + \frac{\sigma_{zz}}{3} \left(\frac{\frac{1}{K_V} - \frac{1}{K_s}}{\frac{1}{K_H} - \frac{1}{K_s}} \right) \tag{2.30}$$

$$S_{s3}' = \left(\frac{2}{3K_H} + \frac{1}{3K_V} - \frac{1}{K_s} \right) + \left(\frac{\theta_s}{K_f} - \frac{\theta_s}{K_s} \right) \tag{2.31}$$

where σ_{xx}, and σ_{yy} are total stresses [M T^{-2} L^{-1}] in the horizontal x and y directions, respectively, σ_{zz} is the total stress [M T^{-2} L^{-1}] in the vertical direction, K_H and K_V are bulk moduli in the horizontal and vertical directions, respectively. It is implicitly assumed that solid grain compressibility is isotropic. A general formulation for anisotropic materials is given by *Guvanasen and Chan* [2000].

2.1.2.5   Wells

The equation describing 1-D 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]:

$$-\overline{\nabla} \cdot (\pi r_s^2 \mathbf{q_w}) \pm Q_w \delta(l - l') - P_w \Gamma_w = \frac{\partial}{\partial t} \left[\left(\frac{\pi r_c^2}{L_s} + \pi r_s^2 \rho g \alpha_w \right) S_{ww} \psi_w \right] \tag{2.32}$$

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

$$\mathbf{q_w} = -K_w k_{rw} \overline{\nabla}(\psi_w + z_w) \tag{2.33}$$

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], P_w is the wetted perimeter of the well [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]:

$$K_w = \frac{r_s^2 \rho g}{8\mu} \tag{2.34}$$

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 Equation 2.32 represents the storage coefficient of the well bore and has two components. The first component represents storage resulting from the variation of the water level in the casing. The second storage component results from the compressibility of the fluid. Although it can be assumed that the contribution to storage from fluid compressibility 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 2.33.

2.1.2.6 Tile Drains

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

$$-\overline{\nabla} \cdot (A\mathbf{q_t}) + Q_t\delta(l - l') - P_t\Gamma_t = \frac{\partial A}{\partial t} \tag{2.35}$$

where $\overline{\nabla}$ is the one-dimensional gradient operator along the length direction, l, of the tile drain, P_t is the wetted perimeter of the tile drain section [L], Q_t is the externally applied or withdrawn fluid flow rate at drain location l' [L^3 T^{-1}]. This is typically zero for most tile drain applications. $\delta(l - l')$ 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].

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

$$\mathbf{q_t} = -K_t k_{rt} \nabla(\psi_t + z_t) \tag{2.36}$$

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]:

$$K_t = \frac{\rho g \psi_t^2}{3\mu} \tag{2.37}$$

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 tile drain 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 2.36, with K_t now being a function of the friction factor and the hydraulic radius R_t [L] of the tile drain with:

$$R_t = \frac{A}{P_t} \tag{2.38}$$

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

$$K_t = \frac{R_t^{2/3}}{n} \left[\frac{\partial \psi_t}{\partial l} + \frac{\partial z_t}{\partial l} \right]^{-1/2} \tag{2.39}$$

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

$$K_t = C_h R_t^{1/2} \left[\frac{\partial \psi_t}{\partial l} + \frac{\partial z_t}{\partial l} \right]^{-1/2} \tag{2.40}$$

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

$$K_t = \sqrt{8g/f}\, R_t^{1/2} \left[\frac{\partial \psi_t}{\partial l} + \frac{\partial z_t}{\partial l} \right]^{-1/2} \tag{2.41}$$

2.2   Surface Flow

2.2.1   General

This section describes the mathematical theory of the surface water flow package of the **HydroGeoSphere** simulator. Surface 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 surface flow processes.

Areal surface flow is represented in **HydroGeoSphere** by a two-dimensional 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 **HydroGeoSphere** we present the simplifications needed to obtain this equation from the full two-dimensional Saint Venant equations.

2.2.2 Governing Equations

2.2.2.1 Surface Runoff

The two-dimensional Saint Venant equations for unsteady shallow water flow consist of 3 equations, which are given by the following mass balance equation:

$$\frac{\partial \phi_o h_o}{\partial t} + \frac{\partial(\overline{v}_{xo}d_o)}{\partial x} + \frac{\partial(\overline{v}_{yo}d_o)}{\partial y} + d_o\Gamma_o \pm Q_o = 0 \tag{2.42}$$

coupled with the momentum equation for the x-direction:

$$\frac{\partial}{\partial t}(\overline{v}_{xo}d_o) + \frac{\partial}{\partial x}(\overline{v}_{xo}^2 d_o) + \frac{\partial}{\partial y}(\overline{v}_{xo}\overline{v}_{yo}d_o) + gd_o\frac{\partial d_o}{\partial x} = gd_o(S_{ox} - S_{fx}) \tag{2.43}$$

and the momentum equation for the y-direction:

$$\frac{\partial}{\partial t}(\overline{v}_{yo}d_o) + \frac{\partial}{\partial y}(\overline{v}_{yo}^2 d_o) + \frac{\partial}{\partial x}(\overline{v}_{xo}\overline{v}_{yo}d_o) + gd_o\frac{\partial d_o}{\partial x} = gd_o(S_{oy} - S_{fy}) \tag{2.44}$$

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 a surface flow domain 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:

$$S_{fx} = \frac{\overline{v}_{xo}\overline{v}_{so}n_x^2}{d_o^{4/3}} \tag{2.45}$$

and:

$$S_{fy} = \frac{\overline{v}_{yo}\overline{v}_{so}n_y^2}{d_o^{4/3}} \tag{2.46}$$

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:

$$S_{fx} = \frac{1}{C_x^2}\frac{\overline{v}_{xo}\overline{v}_{so}}{d_o} \tag{2.45}$$

and:

$$S_{fy} = \frac{1}{C_y^2}\frac{\overline{v}_{yo}\overline{v}_{so}}{d_o} \tag{2.46}$$

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:

$$S_{fx} = \frac{f_x}{8g} \frac{\overline{v}_{xo} \overline{v}_{so}}{d_o} \tag{2.45}$$

and:

$$S_{fy} = \frac{f_y}{8g} \frac{\overline{v}_{yo} \overline{v}_{so}}{d_o} \tag{2.46}$$

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 for laminar flow as [*Akan and Yen*, 1981]:

$$f_i = \frac{C_L}{Re_i} \tag{2.47}$$

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

$$C_L = 24 + 27.162 r^{0.407} \tag{2.48}$$

where r is the rainfall intensity in inches hour^{-1}, and Re_i is the Reynolds number in direction i given as:

$$Re_i = \frac{\overline{v}_{io} d_o}{\gamma} \tag{2.49}$$

where γ is the kinematic viscosity [L^2 T^{-1}].

Momentum Equations 2.43 and 2.44 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 2.45 and 2.46) to give:

$$\overline{v}_{ox} = -K_{ox} \frac{\partial h_o}{\partial x} \tag{2.50}$$

and:

$$\overline{v}_{oy} = -K_{oy} \frac{\partial h_o}{\partial y} \tag{2.51}$$

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

$$K_{ox} = \frac{d_o^{2/3}}{n_x} \frac{1}{[\partial h_o/\partial s]^{1/2}} \tag{2.52}$$

and:

$$K_{oy} = \frac{d_o^{2/3}}{n_y} \frac{1}{[\partial h_o/\partial s]^{1/2}} \tag{2.53}$$

where s is taken in the direction of maximum slope.

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

$$K_{ox} = C_x d_o^{1/2} \frac{1}{[\partial h_o/\partial s]^{1/2}} \qquad (2.52)$$

and:

$$K_{oy} = C_y d_o^{1/2} \frac{1}{[\partial h_o/\partial s]^{1/2}} \qquad (2.53)$$

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

$$K_{ox} = \sqrt{\frac{8g}{f_x}} \frac{1}{[\partial h_o/\partial s]^{1/2}} \qquad (2.52)$$

and:

$$K_{oy} = \sqrt{\frac{8g}{f_x}} \frac{1}{[\partial h_o/\partial s]^{1/2}} \qquad (2.53)$$

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

$$C_x = \frac{d_o^{1/6}}{n_x} = \sqrt{\frac{8g}{f_x}} \quad \text{and} \quad C_y = \frac{d_o^{1/6}}{n_y} = \sqrt{\frac{8g}{f_y}} \qquad (2.54)$$

The surface flow equation solved by **HydroGeoSphere** is finally obtained by substituting Equations 2.50 and 2.51 into continuity Equation 2.42, which gives the following diffusion wave approximation for surface flow:

$$\frac{\partial \phi_o h_o}{\partial t} - \frac{\partial}{\partial x}\left(d_o K_{ox} \frac{\partial h_o}{\partial x}\right) - \frac{\partial}{\partial y}\left(d_o K_{oy} \frac{\partial h_o}{\partial y}\right) + d_o \Gamma_o \pm Q_o = 0 \qquad (2.55)$$

with Equations 2.52 and 2.53 providing expressions for conductances K_{ox} and K_{oy}.

In addition to neglecting the inertial terms, the assumptions associated with 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.

To simplify the presentation of the discretized surface flow equation in the next chapter, Equation 2.55 is rewritten in vectorial notation:

$$-\nabla \cdot (d_o \mathbf{q_o}) - d_o \Gamma_o \pm Q_o = \frac{\partial \phi_o h_o}{\partial t} \qquad (2.56)$$

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

$$\mathbf{q_o} = -\mathbf{K_o} \cdot k_{ro} \nabla(d_o + z_o) \qquad (2.57)$$

where k_{ro} is a dimensionless factor that accounts for the reduction in horizontal conductance from obstruction storage exclusion, which is discussed in Section 2.2.2.2.

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

For investigations of urban runoff or agricultural settings, the storage and flow terms of Equation 2.55 have been further enhanced to include rill storage, also known as depression storage, and exclusion effects from obstruction storage. Figure 2.1 shows various physical settings that can be simulated by the model, given an appropriate formulation of the storage and flow terms.

It can be assumed that flow occurs over a flat plane, as shown in Figure 2.1a, without any rill storage or obstruction storage effects. In that case, the surface flow domain porosity ϕ_o is always unity. 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 can be an important factor in several urban as well as rural settings as shown in Figure 2.1c. It represents the amount of storage that must be filled before any lateral surface flow can occur. Microtopographic relief, relative to the scale of the finite elements in the grid, is included in rill 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 and both rill and obstruction storage need to be included in the model. In addition, horizontal flow term conductances may also be affected over the obstruction heights.

Rill storage and obstruction storage exclusion are both modeled by assuming that the combined depressions and exclusions have 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 2.55, when depression storage is modeled for a system. Thus, surface 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}
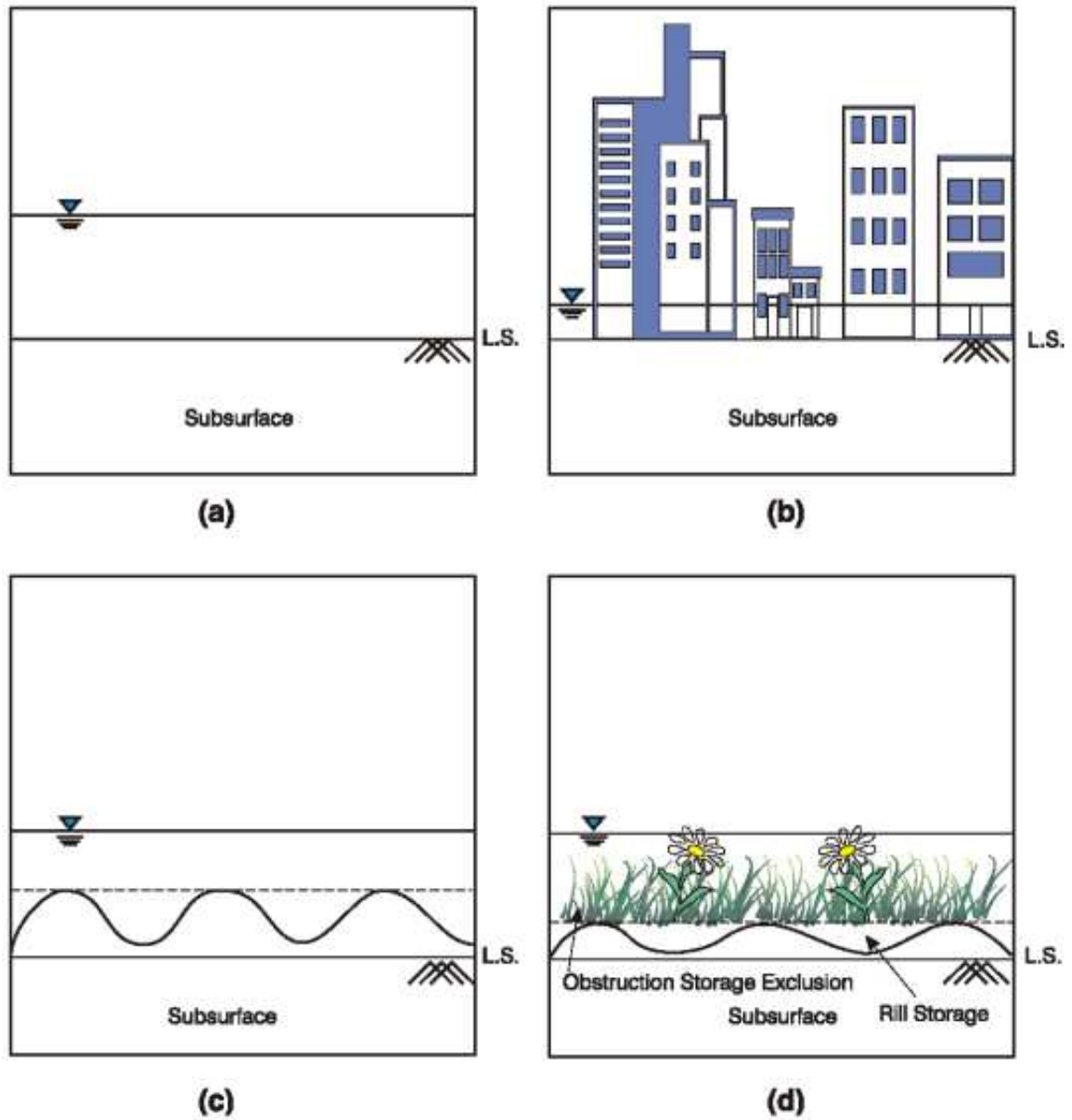
Note that Rill storage, obstruction storage exclusion as well as friction coefficients may be varied on a stress period basis to account for crop growth or changes in land use.

Figure 2.1: Treatment of Storage Terms for Various Settings.
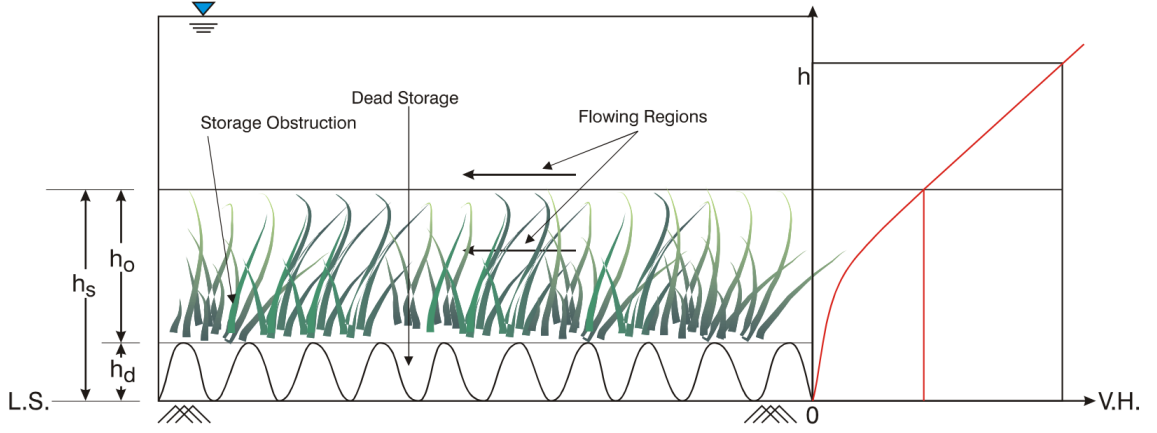
L.S.   =   Land surface = elevation of top of underlying subsurface node.

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.

may be further reduced by a factor k_{ro} above this elevation, up to the obstruction height of storage exclusion to account for the additional resistance losses. The factor k_{ro} varies from zero to unity as the obstruction heights vary from 0 to H_o.

2.2.2.3   Channel flow

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

$$-\overline{\nabla} \cdot (A\mathbf{q_c}) + Q_c \delta(l - l') - \Gamma_c = \frac{\partial A}{\partial t} \qquad (2.58)$$

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

The fluid flux, $\mathbf{q_c}$ [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:

$$\mathbf{q_c} = -K_c k_{rc} \nabla(\psi_c + z_c) \tag{2.59}$$

where k_{rc} is the relative permeability of the channel, ψ_c is the depth of fluid, z_c is the channel elevation [L] and K_c is the channel conductivity [L T^{-1}] approximated for shallow laminar flow as [*Dingman*, 1994]:

$$K_c = \frac{\rho g \psi_c^2}{3\mu} \tag{2.60}$$

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

The discharge $\mathbf{q_c}$ along the channel may alternatively be formulated from friction formulas and the diffusion-wave approximation to the shallow water flow equations. For this case, $\mathbf{q_c}$ is again given by Equation 2.59, with K_c now being a function of the friction factor and the hydraulic radius R_c [L] of the channel with:

$$R_c = \frac{A}{P_c} \tag{2.61}$$

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

$$K_c = \frac{R_c^{2/3}}{n} \left[\frac{\partial \psi_c}{\partial l} + \frac{\partial z_c}{\partial l} \right]^{-1/2} \tag{2.62}$$

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

$$K_c = C_h R_c^{1/2} \left[\frac{\partial \psi_c}{\partial l} + \frac{\partial z_c}{\partial l} \right]^{-1/2} \tag{2.63}$$

where C_h is the Chezy coefficient [L$^{1/2}$ T^{-1}]. For the Darcy-Weisbach equation, K_c is given by:

$$K_c = \sqrt{8g/f}\, R_c^{1/2} \left[\frac{\partial \psi_c}{\partial l} + \frac{\partial z_c}{\partial l} \right]^{-1/2} \tag{2.64}$$

Finally, the unit discharge, q_c, along the channel 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, **HydroGeoSphere** allows the use of rating curves, which are tabulated input of q_c vs ψ_c relationships. These rating curves may be experimentally determined or calculated from formulas readily available for the various structures considered.

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,

Table 2.1: Types of Coupling and Dimensionality for Fluid Flow.

Domains	Coupling	
	Common	Dual
Porous medium - Discrete fractures (2-D)	$\checkmark$	
Porous medium - Second continuum (3-D)		$\checkmark$
Porous medium - Wells (1-D)	$\checkmark$	
Porous medium - Tile drains (1-D)	$\checkmark$	
Porous medium - Surface (2-D)	$\checkmark$	$\checkmark$
Porous medium - Channel (1-D)	$\checkmark$	$\checkmark$

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.

Table 2.1 summarizes the types of fluid flow coupling currently available in **HydroGeo-Sphere**. 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 surface 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   Dual-continuum Subsurface 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):

$$\Gamma_d = \alpha_{wd} K_a k_{ra} (\psi_d - \psi) \tag{2.65}$$

where:

$$\alpha_{wd} = \frac{\beta_d}{a} \left[\frac{\gamma_w}{a} \right] \tag{2.66}$$

where β_d / a is the macropore surface area per unit total volume of the medium [L^{-1}], β_d is a dimensionless geometrical shape factor, a is the fracture-matrix skin thickness [L] over which

the flow exchange occurs, 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 Surface - Subsurface Coupling

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

$$d_o\Gamma_o = \frac{k_r\mathbf{K_{zz}}}{l_{exch}}(h - h_o) \tag{2.67}$$

where a positive Γ_o represents flow from the subsurface system to the surface system as determined by Equation 2.42, h_o is the surface water head, h is the subsurface water head, k_r is the relative permeability for the exchange flux, $\mathbf{K_{zz}}$ is the vertical saturated hydraulic conductivty of the underlying porous medium and l_{exch} is the coupling length. The relative permeability term k_r is same as the relative permeability of the porous medium when water flows from subsurface to surface, while, when water flows from surface to subsurface it is determined by the ratio of the water depth in the surface to the total obstruction height (H_s) such that

$$k_r = \begin{cases} S_{exch}^{2(1-S_{exch})} & \text{when } d_o < H_s \\ 1 & \text{when } d_o > H_s \end{cases} \tag{2.68}$$

where $S_{exch} = d_o/H_s$.

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 or pressure head, areal infiltration or recharge, source/sinks, evaporation, seepage faces, free-drainage and drain nodes. 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.

When **HydroGeoSphere** is used to solve fully coupled subsurface and surface flow, the areal recharge boundary for subsurface flow is not required since the solution to the interacting system determines the subsurface recharge.

2.4.2 Surface Flow

Boundary conditions to the surface 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, as discussed in the next chapter.

2.4.3   Interception and Evapotranspiration

Interception and comprehensive evapotranspiration [*Panday and Huyakorn*, 2004] are simulated as mechanistic processes governed by plant and climatic conditions as noted by *Kristensen and Jensen* [1975], and *Wigmosta et al.* [1994].

2.4.3.1   Interception

Interception is the process involving retention of a certain amount of precipitation on the leaves, branches, and stems of vegetation or on buildings and structures in urban areas. The interception process is simulated by the bucket model, wherein precipitation in excess of interception storage and evaporation from interception reaches the ground surface. The interception storage varies between zero and S_{int}^{Max}, the interception storage capacity [L], which depends on the vegetation type and its stage of development. It is calculated from [*Kristensen and Jensen*, 1975]

$$S_{int}^{Max} = c_{int} LAI \tag{2.69}$$

where LAI is the dimensionless leaf area index and c_{int} is the canopy storage parameter [L].

Note that LAI represents the cover of leaves over a unit area of the ground surface, and may be prescribed in a time-dependent manner. The interception storage is filled by rainfall and depleted by evaporation.

2.4.3.2   Evapotranspiration

Evapotranspiration is rigorously modeled as a combination of plant transpiration and of evaporation, and affects both surface and subsurface flow domains. Transpiration from vegetation occurs within the root zone of the subsurface which may be above or below the water table. The rate of transpiration (T_p) is estimated using the following relationship that distributes the net capacity for transpiration among various factors [*Kristensen and Jensen*, 1975]:

$$T_p = f_1(LAI)\, f_2(\theta)\, RDF[E_p - E_{can}] \tag{2.70}$$

where $f_1(LAI)$ is a function of leaf area index [dimensionless], $f_2(\theta)$ is a function of nodal water content [dimensionless] and RDF is the time-varying root distribution function. The vegetation term is expressed as:

$$f_1(LAI) = \max\{0, \min[1, (C_2 + C_1\, LAI)]\} \tag{2.71}$$

The root zone term is defined by the relationship:

$$RDF = \frac{\int_{z_1'}^{z_2'} r_F(z')dz'}{\int_0^{L_r} r_F(z')dz'}$$ (2.72)

and the moisture content dependence term is expressed as

$$f_2(\theta) = \begin{cases} 0 & \text{for } 0 \leq \theta \leq \theta_{wp} \\ f_3 & \text{for } \theta_{wp} \leq \theta \leq \theta_{fc} \\ 1 & \text{for } \theta_{fc} \leq \theta \leq \theta_o \\ f_4 & \text{for } \theta_o \leq \theta \leq \theta_{an} \\ 0 & \text{for } \theta_{an} \leq \theta \end{cases}$$ (2.73)

where:

$$f_3 = 1 - \left[\frac{\theta_{fc} - \theta}{\theta_{fc} - \theta_{wp}} \right]^{C_3}$$ (2.74)

$$f_4 = 1 - \left[\frac{\theta_{an} - \theta}{\theta_{an} - \theta_o} \right]^{C_3}$$ (2.75)

and where C_1, C_2, and C_3 are dimensionless fitting parameters, L_r is the effective root length [L], z' is the depth coordinate from the soil surface [L], θ_{fc} is the moisture content at field capacity, θ_{wp} is the moisture content at the wilting point, θ_o is the moisture content at the oxic limit, θ_{an} is moisture content at the anoxic limit and $r_F(z')$ is the root extraction function [L^3 T^{-1}], which typically varies logarithmically with depth.

The function f_1 correlates the transpiration (T_p) with the leaf area index (LAI) in a linear fashion. The function f_2 correlates T_p with the moisture state of the roots and is an extension of the function of *Kristensen and Jensen* [1985] to account for root processes in greater detail. Below the wilting-point moisture content, transpiration is zero; transpiration then increases to a maximum at the field-capacity moisture content. This maximum is maintained up to the oxic moisture content, beyond which the transpiration decreases to zero at the anoxic moisture content. When available moisture is larger than the anoxic moisture content, the roots become inactive due to lack of aeration [*Feddes et al.*, 1978]. In general, $f_2(\theta)$ is a nonlinear function of θ, though the ramping function is linear when $C_3 = 1$.

Two models are provided for evaporation. The first model assumes that evaporation occurs if the reference evapotranspiration E_p has not been removed by the above processes of canopy evaporation E_{can} and plant transpiration T_p. Therefore, evaporation from the soil surface and subsurface soil layers is estimated as follows:

$$E_s = \alpha^*(E_p - E_{can} - T_p)EDF$$ (2.76)

The second model assumes that evaporation occurs along with transpiration, resulting from energy that penetrates the vegetation cover and is expressed as

$$E_s = \alpha^*(E_p - E_{can})[1 - f_1(LAI)]EDF$$ (2.77)

where α^* is a wetness factor given by

$$\alpha^* = \begin{cases} \frac{\theta-\theta_{e2}}{\theta_{e1}-\theta_{e2}} & \text{for } \theta_{e2} \leq \theta \leq \theta_{e1} \\ 1 & \text{for } \theta > \theta_{e1} \\ 0 & \text{for } \theta < \theta_{e2} \end{cases} \qquad (2.78)$$

where θ_{e1} is the moisture content at the end of the energy-limiting stage (above which full evaporation can occur) and θ_{e2} is the limiting moisture content below which evaporation is zero [*Allen et al.*, 1998].

Equation 2.78 expresses the moisture availability term for the subsurface domain. For the overland flow domain, α^* is calculated as varying between unity when the elevation of flow is at or above depression storage $z_o + H_D$ and zero for a flow elevation at the land surface (z_o), thus representing the reduced evaporative area of available water in the overland flow domain within the depressions.

The term EDF in Equation 2.76 or 2.77 is the evaporation distribution function that includes the overland and subsurface flow domains. Two alternative conceptualizations are provided for EDF. For the first model, it is assumed that the capacity for evaporation ($(E_p - E_{can} - T_p)$ in Equation 2.76 or $(E_p - E_{can} - T_p)(1 - f_1(LAI))$ in Equation 2.77) decreases with depth below the surface (subject to available moisture) due to the reduction of energy penetration in the soil. Therefore, an appropriate EDF may be prescribed as a function of its depth from land surface. For the second model, the capacity for evaporation is met from the land surface downward to a prescribed extinction depth (B_{soil}).

2.5   Solute Transport

2.5.1   Governing Equations

We present here the basic subsurface transport equation solved by **HydroGeoSphere** which is expanded in its use to incorporate, among other features, discrete fractures, a second porous continuum, wells, tile drains, and the surface flow domain.

2.5.1.1   Porous Medium

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

$$-\nabla \cdot w_m \left(\mathbf{q}C - \theta_s S_w \mathbf{D}\nabla C\right) + [w_m \theta_s S_w R\lambda C]_{par} + \sum \Omega_{\text{ex}} \pm Q_c = w_m \left[\frac{\partial(\theta_s S_w RC)}{\partial t} + \theta_s S_w R\,\lambda\,C\right]$$
$$(2.79)$$

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; however, 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 Equation 2.79. The dimensionless retardation factor, R, is given by [*Freeze and Cherry*, 1979]:

$$R = 1 + \frac{\rho_b}{\theta_s S_w} K'$$ (2.80)

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]:

$$\theta_s S_w \mathbf{D} = (\alpha_l - \alpha_t)\frac{\mathbf{qq}}{|\mathbf{q}|} + \alpha_t |\mathbf{q}|\mathbf{I} + \theta_s S_w \tau D_{free} \mathbf{I}$$ (2.81)

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:

$$\tau = (S_w \theta_s)^{7/3}/\theta_s^2$$ (2.82)

In Equation 2.79, Ω_{ex} represents the mass exchange rate of solutes per unit volume [M L^{-3} T^{-1}] between the subsurface domain and all other types of domains supported by the model. Currently, these additional domains are surface, wells, tile drains, channels, 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=c ex=o, for the fracture, immobile continuum, dual continuum, well, tile drain, channel and surface domains, respectively.

2.5.1.2 Discrete 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:

$$-\nabla \cdot (w_f \mathbf{q_f} C_f - w_f S_{wf} \mathbf{D_f} \nabla C_f) + w_f [R_f \lambda_f C_f]_{par} - w_f \Omega_f =$$

$$w_f \left[\frac{\partial (S_{wf} R_f C_f)}{\partial t} + S_{wf} R_f \lambda_f C_f \right]$$ (2.83)

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 Equation 2.81 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]:

$$R_f = 1 + \frac{2K'_f}{w_f} \tag{2.84}$$

where K'_f is a fracture-surface distribution coefficient [L], which is defined as the ratio between the mass of solute on the solid phase per unit area of an assumed planar fracture surface over the concentration of a solute in solution.

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 2.79. 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:

$$\frac{\partial(\theta_{\text{Imm}}C_{\text{Imm}})}{\partial t} - \Omega_{\text{Imm}} = 0 \tag{2.85}$$

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 represents the water phase and the immobile domain is associated with 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 (2.79). 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:

$$\frac{\partial C_{\text{Imm}}}{\partial t} - \frac{\Omega_{\text{Imm}}}{x_r} = 0 \tag{2.86}$$

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:

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

$$w_d \left[\frac{\partial(\theta_{sd}S_{wd}R_d C_d)}{\partial t} + \theta_{sd}S_{wd}R_d\lambda_d C_d\right] \tag{2.87}$$

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]:

$$R_d = 1 + \frac{\rho_{bd}}{\theta_{sd}S_{wd}}K'_d \tag{2.88}$$

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]:

$$\theta_{sd}S_{wd}\mathbf{D_d} = (\alpha_{ld} - \alpha_{td})\frac{\mathbf{q_d q_d}}{|\mathbf{q_d}|} + \alpha_{td}|\mathbf{q_d}|\mathbf{I} + \theta_{sd}S_{wd}\tau_d D_{free}\mathbf{I} \tag{2.89}$$

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 2.82.

2.5.1.6   Wells

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

$$-\overline{\nabla} \cdot \pi r_s^2 \left(\mathbf{q_w} C_w - S_{ww} D_w \overline{\nabla} C_w \right) + \pi r_s^2 \left[\lambda C_w \right]_{par} -$$

$$Q_w (C_w - C_{w_{\text{Inj}}}) \delta(l - l') - \pi r_s^2 \Omega_w = \pi r_s^2 \frac{\partial C_w}{\partial t} \tag{2.90}$$

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]:

$$D_w = \frac{r_s^2 \mathbf{q_w}^2}{48 D_{free}} + D_{free} \tag{2.91}$$

2.5.1.7   Tile Drains

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

$$-\overline{\nabla} \cdot A \left(\mathbf{q_t} C_t - S_{wt} D_t \overline{\nabla} C_t \right) + A \left[\lambda C_t \right]_{par} - A Q_t (C_t - C_{t_{\text{Inj}}}) \delta(l - l') - A \Omega_t = \frac{\partial A C_t}{\partial t} \tag{2.92}$$

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:

$$D_t = \frac{r_t^2 \mathbf{q_t}^2}{48 D_{free}} + D_{free} \tag{2.93}$$

where r_t is an equivalent radius for a tile drain of cross-section A.

2.5.1.8   Surface runoff

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

$$-\overline{\overline{\nabla}}(q_o C_o - \mathbf{D_o} \phi_o h_o \overline{\nabla} C_o) + [\phi_o h_o R_o \lambda C_o]_{par} - d_o \Omega_o = \frac{\partial}{\partial t}(\phi_o h_o R_o C_o) + \phi_o h_o R_o \lambda C_o \tag{2.94}$$

where C_o is the concentration in water on the surface domain [M L^{-3}], $\mathbf{D_o}$ is the hydrodynamic dispersion tensor of the surface flow domain [L^2 T^{-1}] and $\overline{\overline{\nabla}}$ is the vertically integrated two-dimensional gradient operator. An expression similar to Equation 2.81 is used to represent the dispersion coefficient D_o and the retardation factor R_o is represented by an expression similar to Equation 2.84.

2.5.1.9   Channels

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

$$-\overline{\nabla} \cdot A \left(\mathbf{q_c} C_c - S_{wc} D_c \overline{\nabla} C_c \right) + A \left[\lambda C_c \right]_{par} - A Q_c (C_c - C_{c_{\mathrm{Inj}}}) \delta(l - l') - A \Omega_c = \frac{\partial A C_c}{\partial t} \quad (2.95)$$

where C_c is the solute concentration in the channel [M L^{-3}], $\mathbf{q_c}$ is the fluid flux along the channel axis [L T^{-1}], and $C_{c_{\mathrm{Inj}}}$ is the concentration of injected water.

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

$$D_c = \frac{r_c^2 \mathbf{q_c}^2}{48 D_{free}} + D_{free} \quad (2.96)$$

where r_c is an equivalent radius for a channel of cross-section A.

2.5.1.10   Thermal Energy Transport

Recent research has highlighted the importance of quantifying the impacts of climate change on the hydrological and ecological regimes, prompting the need for a model capable of assessing thermal energy transport in a holistic manner. The fully-integrated framework of **HydroGeoSphere** made it an ideal choice for integrating the thermal energy transport system. **HydroGeoSphere** is a fully-integrated three-dimensional surface/subsurface flow and transport model. It is capable of two-dimensional overland/stream flow and transport and three-dimensional variably-saturated flow and transport in the subsurface including macropores, fractures and karst conduits. These media are fully-coupled and employ a simultaneous solution of the surface/subsurface flow and transport equations using a Control-Volume Finite Element method [*Therrien et al.*, 2007]. Solving the surface and subsurface equations simultaneously for both flow and transport allows for a complete coupling of the interactions within and between the domains. These interactions can play a very important role in assessing the impacts of hydrological, chemical or thermal stressors. Without the integrated approach, these interactions cannot be properly quantified because some form of abstraction of the physical processes would be necessary.

Graf [2005] incorporated heat transport within the saturated-zone flow regime into **HydroGeoSphere** together with temperature-dependent fluid properties, such as viscosity and density. The model's capability was successfully demonstrated for the case of thermohaline flow and transport in porous and fractured porous media [*Graf and Therrien*, 2007]. This work extends the model's capability to include thermal energy transport in the unsaturated zone and in the surface water, which is considered a key step in the linkage between the atmospheric and hydrologic systems. Surface heat fluxes from atmospheric inputs are an important source/sink of thermal energy, especially to the surface water system. As such, surface heat fluxes across the land surface were also incorporated into **HydroGeoSphere**. The following sections will give a brief overview of the equations used to simulate the thermal energy transport in **HydroGeoSphere**. A complete description of the physical

processes and governing flow and solute transport equations that form the basis of **Hydro-GeoSphere** can be found in *Therrien et al.* [2007] and therefore will not be presented here.

The equation describing thermal energy transport in the unsaturated zone is similar to that for the saturated zone, with the inclusion of a saturation term in the bulk transport parameters. The general equation for variably-saturated subsurface thermal energy transport following *Molson et al.* [1992] is given by:

$$[(\partial \rho_b c_b T)/\partial t] = -\nabla[\mathbf{q}\rho_w c_w T - (k_b + c_b \rho_b \mathbf{D})\nabla T] \pm Q_T + \Omega_o \qquad (2.97)$$

where ρ is the density, c is the heat capacity, T is the temperature of the bulk subsurface, t is time, $\mathbf{q}$ is the Darcy flux in the subsurface, k is the thermal conductivity term, $\mathbf{D}$ is the thermal dispersion term, Q_T is a thermal source/sink and Ω_o is the thermal surface/subsurface interaction term, which will be discussed in a following section. The subscript b denotes a bulk term, whereas w represents the aqueous phase. In equation 2.97 1 the Darcy flux ($\mathbf{q}$) in the subsurface is calculated from a numerical solution of the mixed form of Richards' equation, which is valid both above and below the water table, and the water saturation term is included in each of the bulk property terms. The thermal dispersion term (D) is calculated for each flow direction, and for cross-terms to account for anisotropic systems, following [*Burnett and Frind,* 1987]:

$$D_{xx} = \alpha_l(q_x^2)/|q| + \alpha_{th}(q_y^2)/|q| + \alpha_{tv}(q_z^2)/|q| + D^* \qquad (2.98a)$$

$$D_{yy} = \alpha_{th}(q_x^2)/|q| + \alpha_l(q_y^2)/|q| + \alpha_{tv}(q_z^2)/|q| + D^* \qquad (2.98b)$$

$$D_{zz} = \alpha_{tv}(q_x^2)/|q| + \alpha_{tv}(q_y^2)/|q| + \alpha_l(q_z^2)/|q| + D^* \qquad (2.98c)$$

$$D_{xy} = D_{yx} = (\alpha_l - \alpha_{th})(q_x q_y)/|q| \qquad (2.98d)$$

$$D_{xz} = D_{zx} = (\alpha_l - \alpha_{tv})(q_x q_z)/|q| \qquad (2.98e)$$

$$D_{yz} = D_{zy} = (\alpha_l - \alpha_{tv})(q_y q_z)/|q| \qquad (2.98f)$$

where α_l, α_{th}, and α_{tv} are the longitudinal, horizontal transverse and vertical transverse dispersivities, $|q|$ is the magnitude of the Darcy flux, and $D*$ is the effective molecular diffusion coefficient. The bulk parameters, with the exception of thermal conductivity, are calculated by a volumetric average approximation; for example:

$$\rho_b = (1 - \theta)\rho_s + S_w \theta \rho_w + (1 - S_w)\theta \rho_a \qquad (2.99)$$

where θ is porosity, S_w represents the water saturation and the subscripts s and a represent the matrix solids and the air phase, respectively.

The bulk thermal conductivity term (k_b) can either be specified, or calculated. It represents the thermal conductivity of the entire cell, which can include the matrix solids, the aqueous phase and air. Specifying a bulk thermal conductivity may be appropriate for saturated flow conditions, when the phase composition of the subsurface does not vary with space or time; however under variably-saturated flow conditions, the saturations of air and water in the subsurface may change with time, and thus the thermal conductivity may also change.

Calculating a three-phase thermal conductivity is not straightforward; several approaches have been presented, and no one method has proven to consistently provide more representative solutions than others [*Chaudhary and Bhandari,* 1968, *Markle et al.,* 2006]. Two methods of calculating the bulk thermal conductivity are available in **HydroGeoSphere**, allowing the user to determine which method is appropriate for their simulations. The first method uses a volumetric average approximation similar to that employed in SUTRA [*Voss,* 1984]:

$$k_b = (1 - \theta)k_s + S_w\theta k_w + (1 - S_w)\theta k_a \quad (2.100)$$

The second approximation extends the accepted two-phase thermal conductivity calculation provided by *Sass et al.,* [1977] by calculating both a dry and saturated thermal conductivity (kdry and ksat, respectively), and then determining the bulk thermal conductivity using a linear interpolation between the dry and saturated thermal conductivities based on the degree of water saturation:

$$k_{dry} = k_s^{(1-\theta)}k_a^{\theta} \quad (2.101)$$

$$k_{sat} = k_s^{(1-\theta)}k_w^{\theta} \quad (2.102)$$

$$k_b = S_w k_{sat} + (1 - S_w)k_{dry} \quad (2.103)$$

The equation describing thermal energy transport in the surface water is similar to that for solute transport in the surface water. Following the solute transport equation by *Therrien et al.* [2007], it is given by:

$$\frac{\partial \rho_w c_w d_o T_o}{\partial t} = -\nabla[\mathbf{q}_o \rho_w c_w T_o - (k_w + D_o \rho_w c_w)d_o \nabla T_o] + E_{atm} \pm Q_{T_o} - d_o \Omega_o \quad (2.104)$$

where d is the depth of the surface water, E_{atm} represents the atmospheric inputs to the surface thermal energy system, and the subscript o denotes overland (or stream) flow. Both solute and thermal energy transport are depth averaged in the surface domain within **HydroGeoSphere**. In equation 2.104, the overland water flux ($\mathbf{q}_o$) is calculated from the numerical solution of the Diffusion-wave equation, together with Manning's equation, which is fully coupled to the Richards' equation describing subsurface flow. These derivations for thermal energy transport are hydrodynamically, and not thermodynamically based, and are not applicable to high temperature, high pressure hydrothermal conditions, but are valid for most shallow groundwater/surface water systems. In addition, the surface thermal regime is depth-averaged and cannot represent thermal stratification in surface water bodies.

The inclusion of atmospheric thermal energy inputs is necessary to properly simulate the surface and subsurface temperature regimes. In this work, the atmospheric inputs from CLASS [*Verseghy,* 1991] are used to determine the surface heat fluxes in **HydroGeoSphere**. CLASS is a well established land surface scheme, and the atmospheric inputs incorporated into CLASS have been demonstrated to be representative [*Verseghy,* 1991]. In addition to the reliability of the CLASS approach, the equations used for atmospheric inputs are also computationally inexpensive, relative to other atmospheric models, and thus are appropriate for implementing into **HydroGeoSphere**. The total atmospheric input to the surface thermal energy system can be expressed as:

$$E_{atm} = K_* + L_* + Q_H + Q_E \quad (2.105)$$

where K_* is the net shortwave radiation, L_* is the net longwave radiation, Q_H is the sensible heat flux and Q_E is the latent heat flux . All of the equations used here are taken from CLASS [*Verseghy,* 1991] with the exception of the incoming longwave radiation calculation, which is adapted from *Fassnacht et al.* [2001]. All of the atmospheric thermal inputs are calculated explicitly, and are treated as a source/sink term in the thermal energy transport equation for the surface regime. A limitation to this approach is that the model does not provide feedback to the atmosphere; however, given the scale of the atmospheric regime, it is assumed that the feedback from the smaller hydrologic domains is negligible.

Shortwave radiation is the radiant energy in the visible, near-ultraviolet and near-infrared wavelengths from the atmosphere to the Earth's surface, broadly defined as between 0.1 and 5.0 micrometers. The equation used to calculate the thermal energy input to the surface from net shortwave radiation follows *Verseghy* [1991] and is:

$$K_* = (1 - \alpha_g)K^{\downarrow} \tag{2.106}$$

where α_g is the ground surface albedo, and $K^{\downarrow}$ is the incoming shortwave radiation. Ground surface albedo is dependent on the water content of the surficial soil, as formulated by *Idso et al.*[1975], and is given by:

$$\begin{array}{llll}
\alpha_g = & \frac{S_w(1-\theta)(\alpha_{sat}-\alpha_{dry})}{0.20} + \alpha_{dry} & \text{for} & S_w(1-\theta) < 0.20 \\
\alpha_g = & \alpha_{sat} & \text{for} & S_w(1-\theta) \geq 0.20
\end{array} \tag{2.107}$$

where α_{sat} and α_{dry} are the limiting wet and dry soil albedoes.

A variation of this equation is also available in **HydroGeoSphere** to account for cloud and canopy cover (given by C_c):

$$K_* = (1 - C_c)(1 - \alpha_g)K^{\downarrow} \tag{2.108}$$

The cloud and canopy cover term (C_c) varies between 0 and 1 and represents the fraction of the sky that is blocked by either clouds or vegetation from the ground surface. Net Longwave Radiation (L_*) Longwave radiation is the infrared energy emitted by the earth and atmosphere at wavelengths between about 5 and 25 micrometers. The equation for net longwave radiation to the surface following *Verseghy* [1992] is:

$$L_* = L^{\downarrow} - \sigma T_g^4 \tag{2.109}$$

where $L^{\downarrow}$ is the incoming longwave radiation, σ is the Steffan-Boltzmann constant, and T_g is the ground surface temperature (temperature of the surface regime). In the original CLASS formulation, the incoming radiation term is specified as an input parameter. However, due to the lack of longwave radiation data (incoming longwave radiation is not routinely measured), incoming longwave radiation can also be calculated in **HydroGeoSphere** using the formulation given by *Fassnacht et al.* [2001]:

$$L^{\downarrow} = \varepsilon_{at}\sigma T_a^4 \tag{2.110}$$

where T_a is the air temperature, ε_{at} is the integrated emissivity of the atmosphere and canopy, calculated by:

$$\varepsilon_{at} = (0.53 + 0.2055e_a^{0.5})(1 + 0.40C_c) \tag{2.111}$$

where e_a is the near surface vapour pressure. Emissivity is limited so that it cannot exceed 1.0, ensuring that the incoming longwave radiation is not overestimated.

Sensible heat flux represents the movement of energy from the Earth to the air above typically via conduction. The equation used for sensible heat flux follows *Verseghy* [1991] and is given by:

$$Q_H = \rho_a c_a V_a c_D [T_a - T_g] \tag{2.112}$$

where ρ_a is the density of the air, c_a is the specific heat of the air, V_a is the wind speed and c_D is the drag coefficient.

Latent heat flux represents the energy transfer during the evaporation/condensation process. There are three methods of accounting for evaporation in the flow solution. The quantity of water evaporated/condensed in the flow solution provides the basis for the amount of energy transferred between the atmospheric and hydrologic regimes for latent heat flux. As such, the evaporation rate used in the flow solution must be linked to the latent heat flux term used to calculate the atmospheric thermal energy inputs. The first, and most simplistic method of accounting for evaporation, is to simply reduce the precipitation rate applied to the surface of the domain by the evaporation rate. By not explicitly specifying or calculating the evaporation rate in **HydroGeoSphere**, the latent heat flux must be determined independently. In this case the equation used by CLASS [*Verseghy, 1991*] is used, given by:

$$Q_E = L_V \rho_a V_a c_D [SH_a - SH_g] \tag{2.113}$$

Where L_V is the latent heat of vaporization, SH_a is the specific humidity of the air, and SH_g is the specific humidity of the ground surface. Specific humidity of the ground surface is calculated using the same formulation given by Verseghy (1991):

$$SH_g = h SH_{sat}[T_g] \tag{2.114}$$

Where h is the relative humidity of the air in the surface soils, calculated by:

$$h = exp \left[\frac{-g\psi_g}{R_w T_g} \right] \tag{2.115}$$

and $SH_{sat}[T_g]$ is the saturation specific humidity at T_g, given by:

$$SH_{sat}[T_g] = \frac{0.622 e_{sat}[T_g]}{p_a - 0.378 e_{sat}[T_g]} \tag{2.116}$$

In these equations, g represents the acceleration due to gravity, ψ_g is the soil-water suction at the surface, R_w is the gas constant for water vapour, $e_{sat}[T_g]$ is the saturation vapour pressure at the ground surface and p_a is the air pressure. Evaporation can also be specified or calculated in **HydroGeoSphere**. When evaporation is specified, the evaporation rate is input to **HydroGeoSphere** and is subtracted from the incoming precipitation throughout the simulation. Internally calculating the evaporation rate is a more complex approach, based on the empirical Hargreaves equation for determining the potential evapotranspiration, and then calculating the actual evapotranspiration as a combination of plant transpiration and evaporation from the surface and the subsurface domains. The equations

for this approach are given in detail by Li et al., (2008). Whether the evaporation is speci-
fied or calculated, the latent heat flux is then determined from the evaporation rate used in
the flow solution, ensuring continuity between the flow and thermal transport simulations.
The latent heat flux equation for a specified or calculated evaporation rates is:

$$Q_E = L_V E_{flux} \rho_w \tag{2.117}$$

where E_{flux} is the specified or calculated evaporation rate.

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 superposi-
tion principle (see *Therrien and Sudicky*, 1996), where continuity of solute concentration is
assumed between the two domains concerned, which corresponds to instantaneous equilib-
rium 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.

Table 2.2 summarizes the types of solute transport coupling currently available in **Hydro-
GeoSphere**. The common node approach is currently the only one available in **Hydro-
GeoSphere** 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 be-
tween the surface 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:

$$\Omega_{\text{Imm}} = \alpha_{\text{Imm}}(C - C_{\text{Imm}}) \tag{2.118}$$

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

Table 2.2: Types of Coupling and Dimensionality Solute Transport.

Domains	Coupling	
	Common	Dual
Porous medium - Discrete fractures (2-D)	$\checkmark$	
Porous medium - Second continuum (3-D)		$\checkmark$
Porous medium - Double porosity (3-D)		$\checkmark$
Porous medium - Wells (1-D)	$\checkmark$	
Porous medium - Tile drains (1-D)	$\checkmark$	
Porous medium - Surface (2-D)	$\checkmark$	$\checkmark$
Porous medium - Channel (1-D)	$\checkmark$	$\checkmark$

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:

$$\alpha_{\text{Imm}} = \frac{15\theta_{\text{Imm}}D_{\text{Imm}}^*}{r_0^2} \tag{2.119}$$

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 B_f and where the porous matrix blocks are prismatic slabs:

$$\alpha_{\text{Imm}} = \frac{3\theta_{\text{Imm}}D_{\text{Imm}}^*}{(B_f/2)^2} \tag{2.120}$$

2.6.2 Dual-continuum Subsurface 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):

$$\Omega_d = -u_m C - u_d C_d \tag{2.121}$$

where:

$$u_m = d^*\Gamma_d\phi - \alpha_s w_m \theta_s S_w \tag{2.122}$$

and:

$$u_d = (1 - d^*)\Gamma_d\phi^* + \alpha_s w_m \theta_s S_w \tag{2.123}$$

In Equation 2.122 and 2.123, α_s is a mass transfer coefficient [T^{-1}] resembling that defined for the double porosity option. We further define the following variables:

$$d^* = 0.5\left(1 - \frac{\Gamma_d}{|\Gamma_d|}\right) \tag{2.124}$$

and

$$\phi = w_m \frac{\theta_s S_w}{\theta_{tot}}; \quad \phi^* = (1 - w_m) \frac{\theta_{sd} S_{wd}}{\theta_{tot}}; \quad \theta_{tot} = w_m \theta_s S_w + (1 - w_m) \theta_{sd} S_{wd} \qquad (2.125)$$

For the case where the hydraulic heads of the porous medium and the second continuum are equivalent, then, according to Equation 2.65, the term Γ_d above becomes equal to 0 and the solute exchange given by term Ω_d reduces to a diffusion-type exchange between the two domains, similar to that for the double porosity medium (see Equation 2.118).

2.6.3   Surface - Subsurface 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.

2.6.4   Surface - Subsurface 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.

2.6.5   Thermal Energy Coupling

The coupling of the surface and subsurface thermal continua is similar to that used for advective-dispersive contaminant transport in **HydroGeoSphere**. There are two methods of coupling the surface and subsurface continua, the common node and the dual node approaches. The common node approach is based on the assumption of continuity of temperature at the surface/subsurface interface. The dual node approach on the other hand uses a first-order flux relation to transfer heat from one domain to the other. The equation for the dual-node coupling of the surface and subsurface thermal equations follows Therrien et al. (2007) and is given by:

$$\Omega_o = \rho_w c_w T_{ups} \Gamma_o + \alpha_o \rho_{dwn} c_{dwn} (T - T_o) \qquad (2.126)$$

where Γ_o represents the aqueous exchange flux between the surface and subsurface (the amount of water flowing between the two regimes) and α_o is an energy transfer coefficient determined by the thermal dispersivity over the depth of the surface/subsurface exchange zone. The subscript *ups* represents the "upstream" direction, and the subscript *dwn* represents the "downstream" direction (i.e. $T_{ups} = T, \rho_{dwn} c_{dwn} = \rho_w c_w$ when $\Gamma_o > 0$ and $T_{ups} = T_o, \rho_{dwn} c_{dwn} = \rho_b c_b$ when $\Gamma_o < 0$). The most significant difference between the solute and the thermal transport coupling equations is the treatment of mass/energy transfer between the surface and subsurface. The downstream parameters are used to differentiate between the amount of thermal energy required to change the temperature of the surface

and the subsurface domains, respectively. When the diffusive gradient is transferring thermal energy to the subsurface, the bulk heat capacity and density parameters regulate how much the temperature of the bulk subsurface changes given the amount of thermal energy added. Conversely, when the diffusive gradient is transporting thermal energy to the surface, the aqueous heat capacity and density terms regulate how much the surface water temperature increases given the thermal energy inputs. As the bulk and aqueous heat capacity and density terms can be significantly different, the amount of energy required to change the temperature of the surface and subsurface regimes can also be different. For this formulation, the downstream location (where the energy is diffusing to) determines how much thermal energy is required to reduce the thermal gradient between regimes. The movement of a solute between the two regimes is different from heat because a non-sorbing solute has a tendency to remain in the aqueous phase, whereas thermal energy tends to preferentially transfer into the solid phase of the porous medium, thus affecting the bulk temperature.

2.6.6 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:

$$\Omega_{\text{Imm}} = x_r k_r (\alpha_r C - C_{\text{Imm}}) \tag{2.127}$$

where k_r is a reverse fractionation rate [L^{-1}] and $k_r \alpha_r$ is the forward reaction rate with α_r being the isotope fractionation factor between water and the solid phase. Concentration C_{Imm} describes here the isotopic concentration in the solid phase.

2.7 Solute Transport Boundary Conditions

2.7.1 Subsurface

Boundary conditions for subsurface transport include the following: first-type (Dirichlet) boundaries of prescribed concentration, prescribed mass flux or third-type (Cauchy) boundary conditions. For chain-decay of solutes, Bateman's equation is used to prescribe the concentrations for the daughter products. 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.

2.7.2 Surface

Boundary conditions to the surface flow system include at the moment first-type (Dirichlet) boundaries of prescribed concentration.

2.8   Travel Time Probability

2.8.1   Definitions

Groundwater age is usually defined as a relative quantity with respect to a starting location where age is assumed to be zero. For a given spatial position in the reservoir, the age (A) relates to the time elapsed since the water particles entered the system at the recharge limits, where age is zero. For the same spatial position, the life expectancy (E) is defined as the time required for the water particles to reach an outlet limit of the system. Life expectancy is therefore zero at an outlet. The total transit time (T) finally refers to the total time required by the same water particles to migrate from an inlet zone ($T = E$) to an outlet zone ($T = A$). The three variables A, E and T are random variables, characterized by probability density functions (PDF's) $g_{U=A,E,T}$, that can be regarded as the statistical occurrence of water particles with respect to time, which could be observed in a groundwater sample if any analytical procedure would allow such measurements.

The travel time probability $g_t(t, \boldsymbol{x} \mid t_0, \boldsymbol{x}_i)$ characterizes the probability density for the amount of time (t is a random variable) required by the water particles to travel from a given position $\boldsymbol{x}_i$ (at time $t = t_0$) to the position $\boldsymbol{x}$. The location probability $g_{\boldsymbol{x}}(\boldsymbol{x}, t \mid \boldsymbol{x}_i, t_0)$ characterizes the probability density of finding water particles at the position $\boldsymbol{x}$ ($\boldsymbol{x}$ is a random variable) at a given time t after their release at the position $\boldsymbol{x}_i$. If the input zone corresponds to the entire inlet Γ_-, then travel time corresponds to age A. Similarly, if the input zone corresponds to the entire outlet Γ_+, then travel time corresponds to life expectancy E.

The age (and/or forward travel time) and life expectancy (and/or backward travel time) pdfs can be obtained as solutions of advection-dispersion equations (ADE), by making use of specific boundary conditions.

2.8.2   Basic equations

2.8.2.1   Forward model

The age PDF at a position $\boldsymbol{x}$ in an aquifer Ω can be evaluated by solving the ADE when a unit pulse of conservative tracer is uniformly applied on the recharge area Γ_- (see Fig. 2.3). The resulting breakthrough curve is the probabilistic age distribution [*Danckwerts*, 1953; *Jury*, 1990]. The pre-solution of a velocity field is performed by solving the groundwater flow equation. The age PDF is then obtained by solving the following forward boundary value problem:

$$\frac{\partial \theta g}{\partial t} = -\nabla \cdot \mathbf{q} g + \nabla \cdot \theta \mathbf{D} \nabla g + q_{\mathrm{I}} \delta(t) - q_{\mathrm{O}} g \qquad \text{in} \quad \Omega \qquad (2.128a)$$

$$g(\boldsymbol{x}, 0) = g(\boldsymbol{x}, \infty) = 0 \qquad \text{in} \quad \Omega \qquad (2.128b)$$

$$\mathbf{J}(\boldsymbol{x}, t) \cdot \boldsymbol{n} = (\mathbf{q} \cdot \boldsymbol{n}) \delta(t) \qquad \text{on} \quad \Gamma_- \qquad (2.128c)$$

$$\mathbf{J}(\boldsymbol{x}, t) \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad \Gamma_0 \tag{2.128d}$$

where $g(\boldsymbol{x}, t) = g_A(\boldsymbol{x}, t)$ denotes the transported age PDF $[\mathrm{T}^{-1}]$, $\mathbf{q}$ is the water flux vector $[\mathrm{LT}^{-1}]$, q_I and q_O are fluid source and sink terms, respectively $[\mathrm{T}^{-1}]$, $\mathbf{J}(\boldsymbol{x}, t)$ is the total age mass flux vector $[\mathrm{LT}^{-2}]$, $\mathbf{D}$ is the tensor of macro-dispersion $[\mathrm{L}^2\mathrm{T}^{-1}]$, $\boldsymbol{x} = (x, y, z)$ is the vector of Cartesian coordinates $[\mathrm{L}]$, t is time $[\mathrm{T}]$, $\theta = \theta(\boldsymbol{x})$ is porosity or mobile water content $[-]$, $\boldsymbol{n}$ is a normal outward unit vector, and $\delta(t)$ is the time-Dirac delta function $[\mathrm{T}^{-1}]$, which ensures a pure impulse on Γ_-. The total age mass flux vector $\mathbf{J}(\boldsymbol{x}, t)$ is classically defined by the sum of the convective and dispersive fluxes:

$$\mathbf{J}(\boldsymbol{x}, t) = \mathbf{q}g(\boldsymbol{x}, t) - \mathbf{D}\nabla g(\boldsymbol{x}, t) \tag{2.129}$$

The third-type (Cauchy) boundary condition (2.128c) is the most meaningful condition to simulate the age problem since it prevents backward losses by dispersion (homogeneity of the condition at $t = 0^+$).

2.8.2.2 Backward model

The life expectancy PDF satisfies the adjoint backward model of Eq. (2.128a):

$$\frac{\partial \theta g}{\partial t} = \nabla \cdot \mathbf{q}g + \nabla \cdot \theta \mathbf{D}\nabla g - q_\mathrm{I}g \qquad \text{in} \quad \Omega \tag{2.130a}$$

$$g(\boldsymbol{x}, 0) = g(\boldsymbol{x}, \infty) = 0 \qquad \text{in} \quad \Omega \tag{2.130b}$$

$$\mathbf{J}(\boldsymbol{x}, t) \cdot \boldsymbol{n} = -(\mathbf{q} \cdot \boldsymbol{n})\delta(t) \qquad \text{on} \quad \Gamma_+ \tag{2.130c}$$

$$-\mathbf{D}\nabla g(\boldsymbol{x}, t) \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad \Gamma_0 \tag{2.130d}$$

where $g(\boldsymbol{x}, t) = g_E(\boldsymbol{x}, t)$ denotes the transported life expectancy PDF, and where the total life expectancy mass flux vector $\mathbf{J}(\boldsymbol{x}, t)$ is

$$\mathbf{J}(\boldsymbol{x}, t) = -\mathbf{q}g(\boldsymbol{x}, t) - \mathbf{D}\nabla g(\boldsymbol{x}, t) \tag{2.131}$$

Eq. (2.130a) is the formal adjoint of Eq. (2.128a) [*Garabedian*, 1964; *Arnold*, 1974], known as the "backward-in-time" equation [*Uffink*, 1989; *Wilson*, 1997] or the backward Kolmogorov equation [*Kolmogorov*, 1931]. Given the forward equation, the backward equation is technically obtained by reversing the sign of the flow field, and by adapting the boundary conditions [*Neupauer*, 1999; *Neupauer*, 2001]. On the impermeable boundary Γ_0, a third-type condition (Cauchy) in the forward equation becomes a second-type condition (Neumann) in the backward equation, and vice-versa. A second-type condition in the forward model will also become a third-type condition in the backward model [*Gardiner*, 1983, p. 146]. The advection term is known to be not self-adjoint (it should be written in the form $\mathbf{q} \cdot \nabla g$ in Eq. (2.130a)) unless flow is non-divergent. However, the backward equation can still handle divergent flow fields by means of the important sink term $-q_\mathrm{I}g$ appearing in Eq. (2.130a). This sink term has been derived in *Cornaton* [2003] from the vertical averaging process of the general 3D backward ADE, and is consistent with the analysis of *Neupauer* [2001; 2002]
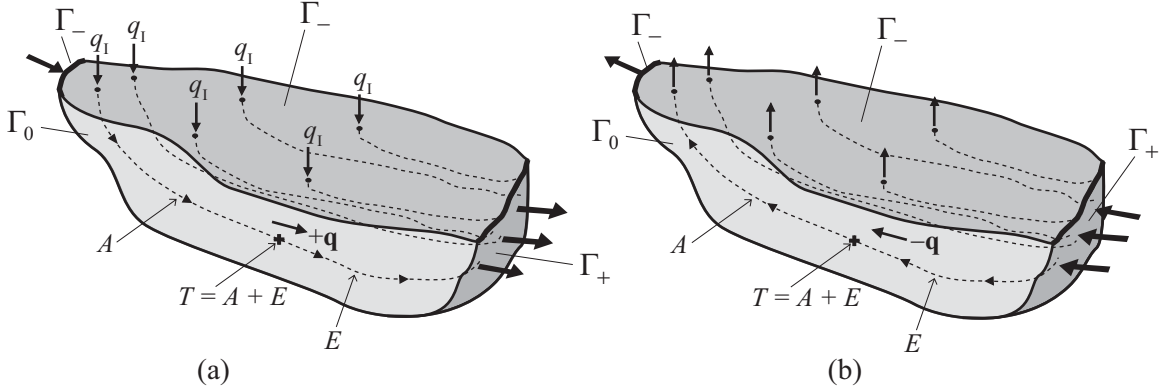
Figure 2.3: Schematic illustration of a groundwater reservoir Ω, with inlet (Γ_-) and outlet (Γ_+) boundaries: (a) Age problem with normal flow field; (b) Life expectancy problem with reversed flow field. The cross stands for a small water sample, to illustrate the random variable total transit time (T) as the sum of the two random variables age (A) and life expectancy (E).

and *Wilson* [1997]. Recharge by internal sources (3D or 2D vertical) or by areal fluxes (fluid source for 2D horizontal) is introduced by the first-order decay type term $-q_{\mathrm{I}}g$, which is a consequence of the reversed flow field. Internal sources produce a sink of life expectancy probability, while internal sinks (term $q_{\mathrm{O}}g$ in Eq. (2.128a)) do not appear in the backward model since a fluid sink may not influence the life expectancy PDF.

The life expectancy to a specific outlet $\Gamma_n \subset \Gamma_+$ can be calculated by modifying the boundary value problem (2.130) in such a way that the function g will characterize the probability density for the water particles to reach Γ_n, exclusively. This can be done by assuming a maximum intensity of probability of exit at Γ_n ($\mathbf{J}(\boldsymbol{x}, t) \cdot \boldsymbol{n} = -(\mathbf{q} \cdot \boldsymbol{n})\delta(t)$) and a minimum intensity of probability of exit at each other outlet ($\mathbf{J}(\boldsymbol{x}, t) \cdot \boldsymbol{n} = 0$).

2.8.2.3   Total transit time

Since $T = A + E$, and since A and E are independent variables, the total transit time PDF g_T is obtained by the following convolution product:

$$g_T(\boldsymbol{x}, t) = \int_0^t g_A(\boldsymbol{x}, \tau) g_E(\boldsymbol{x}, t - \tau) d\tau \tag{2.132}$$

The field of g_T characterizes the evolution of groundwater particles throughout the aquifer domain by specifying the amount of time from recharge to discharge. At a given position in the reservoir, the temporal evolution of the groundwater particles can be characterized by the three PDF's g_A, g_E and g_T. Each function contains specific information on a time of residence, the nature of which is a function of the spatial references that are chosen for evaluation. For instance, g_A is conditioned by the inlet limit Γ_-, where the variable A is nil, while g_E is conditioned by the outlet limit Γ_+, where the variable E is nil. For the

variable T, the PDF g_T is conditioned by the fact that $T = A$ at outlet, and that $T = E$ at inlet.

2.8.2.4   Outlet/Inlet Transit Time PDF

The representative transit time distribution $\varphi(t)$ of the reservoir outlet zone can be defined as a flux averaged concentration [*Rubin*, 2003], i.e. $\varphi(t)$ is evaluated as the flow rate-normalized sum on Γ_+ of the total age mass flux response function $\mathbf{J}$ resulting from a unit flux impulse on Γ_-:

$$\varphi(t) = \frac{1}{F_0} \int_{\Gamma_+} \mathbf{J} \cdot \boldsymbol{n} \, d\Gamma \tag{2.133}$$

where F_0 denotes the total flow rate through Γ_+. Similarly, the inlet transit time PDF $\varphi(t)$ is derived by enforcing Eq. (2.133) on the inlet boundary Γ_-, given that the function $\mathbf{J}$ represents the life expectancy mass flux.

2.8.2.5   Travel Time Probabilities

The forward and backward travel time probabilities are calculated as solutions of the following forward and backward initial value problems:

$$\frac{\partial \theta g}{\partial t} = -\nabla \cdot \mathbf{q}g + \nabla \cdot \theta \mathbf{D}\nabla g - q_{\mathrm{O}}g \qquad \text{in} \quad \Omega \tag{2.134a}$$

$$g(\boldsymbol{x},0) = \frac{\delta(\boldsymbol{x} - \boldsymbol{x}_i)}{\theta} \qquad \text{in} \quad \Omega \tag{2.134b}$$

$$[\mathbf{q}g(\boldsymbol{x},t) - \mathbf{D}\nabla g(\boldsymbol{x},t)] \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad \Gamma_- \cup \Gamma_0 \tag{2.134c}$$

for the forward problem, and

$$\frac{\partial \theta g}{\partial t} = \nabla \cdot \mathbf{q}g + \nabla \cdot \theta \mathbf{D}\nabla g - q_{\mathrm{I}}g \qquad \text{in} \quad \Omega \tag{2.135a}$$

$$g(\boldsymbol{x},0) = \frac{\delta(\boldsymbol{x} - \boldsymbol{x}_i)}{\theta} \qquad \text{in} \quad \Omega \tag{2.135b}$$

$$[-\mathbf{q}g(\boldsymbol{x},t) - \mathbf{D}\nabla g(\boldsymbol{x},t)] \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad \Gamma_+ \tag{2.135c}$$

$$-\mathbf{D}\nabla g(\boldsymbol{x},t) \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad \Gamma_0 \tag{2.135d}$$

for the backward problem. The variable $\boldsymbol{x}_i$ denotes the location for the release of a unit mass.

The forward and backward *location* probabilities $[\mathrm{L}^{-3}]$ are then defined by:

$$g_{\boldsymbol{x}}(\boldsymbol{x},t) = \theta(\boldsymbol{x})g(\boldsymbol{x},t) \tag{2.136}$$
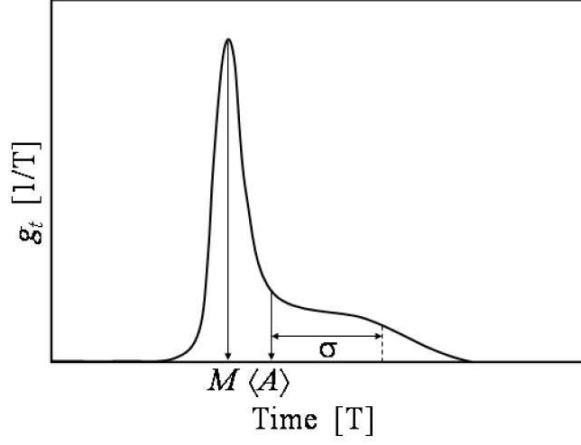
Figure 2.4: Descriptive statistics on the travel time PDF.

The forward and backward *travel time* probabilities $[\text{T}^{-1}]$ are defined by:

$$g_t(t, \boldsymbol{x}) = \|\mathbf{q}(\boldsymbol{x}, t)\| A(\boldsymbol{x}) g(t, \boldsymbol{x}) \tag{2.137}$$

where $A(\boldsymbol{x})$ denotes the area of a control plane orthogonal to velocity, through which the travel time probability is evaluated.

2.8.2.6 Age, Life Expectancy and Travel Time Statistics

Given an age/life expectancy/travel time PDF solution $g_t(t, \boldsymbol{x})$, the following descriptive statistics can be post-processed (see Fig. 2.4)):

1. Mean age/life expectancy/travel time μ:

$$\mu(\boldsymbol{x}) = \int_0^{+\infty} t g_t(t, \boldsymbol{x}) dt \tag{2.138}$$

2. Standard deviation σ:

$$\sigma(\boldsymbol{x}) = \sqrt{\int_0^{+\infty} t^2 g_t(t, \boldsymbol{x}) dt - \mu^2} \tag{2.139}$$

3. Mode $M(\boldsymbol{x})$.

2.8.2.7 Mean Age and Mean Life Expectancy Direct Solutions

Temporal moment equations can be derived from Eqs. (2.128) and (2.130). For instance, the *mean age equation* is obtained by taking the first moment form of Eq. (2.128):

$$-\nabla \cdot \mathbf{q}\langle A\rangle + \nabla \cdot \theta \mathbf{D}\nabla\langle A\rangle - q_{\mathrm{O}}\langle A\rangle + \theta = 0 \qquad \text{in} \quad \Omega \tag{2.140}$$

where $\langle A\rangle$ denotes mean age. It requires the following boundary conditions:

$$\langle A\rangle(\boldsymbol{x}) = 0 \qquad \text{on} \quad \Gamma_- \tag{2.141a}$$

$$\mathbf{J}(\boldsymbol{x}) \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad \Gamma_0 \tag{2.141b}$$

The *mean life expectancy equation* is similarly obtained by taking the first moment form of Eq. (2.130):

$$\nabla \cdot \mathbf{q}\langle E\rangle + \nabla \cdot \theta \mathbf{D}\nabla\langle E\rangle - q_{\mathrm{I}}\langle E\rangle + \theta = 0 \qquad \text{in} \quad \Omega \tag{2.142}$$

where $\langle E\rangle$ denotes mean life expectancy. It requires the following boundary conditions:

$$\langle E\rangle(\boldsymbol{x}) = 0 \qquad \text{on} \quad \Gamma_+ \tag{2.143a}$$

$$-\mathbf{D}\nabla\langle E\rangle(\boldsymbol{x}) \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad \Gamma_0 \tag{2.143b}$$

Mean age and mean life expectancy are continuously generated during groundwater flow, since porosity $\theta = \theta(\boldsymbol{x})$ acts as a source term in Eqs. (2.140) and (2.142). This source term indicates that groundwater is aging one unit per unit time, in average. Finally, mean total transit time (from inlet to outlet) can be obtained by taking the first moment form of Eq. (2.132), yielding to $\langle T\rangle = \langle A\rangle + \langle E\rangle$.

2.8.2.8 Evaluating the travel time PDF from the travel time CDF

An alternative technique for the evaluation of the travel time PDF is to solve for the CDF, which can be done by making use of the following forward and backward boundary value problems:

$$\frac{\partial \theta G}{\partial t} = -\nabla \cdot \mathbf{q}G + \nabla \cdot \theta \mathbf{D}\nabla G - q_{\mathrm{O}}G \qquad \text{in} \quad \Omega \tag{2.144a}$$

$$G(\boldsymbol{x}, 0) = 0 \qquad \text{in} \quad \Omega \tag{2.144b}$$

$$[\mathbf{q}G(\boldsymbol{x}, t) - \mathbf{D}\nabla G(\boldsymbol{x}, t)] \cdot \boldsymbol{n} = \mathbf{q} \cdot \boldsymbol{n} \qquad \text{on} \quad \Gamma_- \tag{2.144c}$$

$$[\mathbf{q}G(\boldsymbol{x}, t) - \mathbf{D}\nabla G(\boldsymbol{x}, t)] \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad \Gamma_0 \tag{2.144d}$$

for the forward problem, and

$$\frac{\partial \theta G}{\partial t} = \nabla \cdot \mathbf{q}G + \nabla \cdot \theta \mathbf{D}\nabla G - q_{\mathrm{I}}G \qquad \text{in} \quad \Omega \tag{2.145a}$$

$$G(\boldsymbol{x}, 0) = 0 \qquad \text{in} \quad \Omega \tag{2.145b}$$

$$[-\mathbf{q}G(\boldsymbol{x},t) - \mathbf{D}\nabla G(\boldsymbol{x},t)] \cdot \boldsymbol{n} = -\mathbf{q} \cdot \boldsymbol{n} \qquad \text{on} \quad \Gamma_+ \tag{2.145c}$$

$$- \mathbf{D}\nabla G(\boldsymbol{x},t) \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad \Gamma_0 \tag{2.145d}$$

for the backward problem. The function $G(\boldsymbol{x},t)$ is the forward (or backward) travel time CDF, from which the travel time PDF $g(\boldsymbol{x},t)$ can be deduced by enforcing:

$$g(\boldsymbol{x},t) = \frac{\partial G(\boldsymbol{x},t)}{\partial t} \tag{2.146}$$

This formulation is more appropriate to account for transient velocity fields. Note that the boundary conditions (2.144c) and (2.145c) may be replaced by the first-type condition $G(\boldsymbol{x},t) = 1$. Note also that the n temporal moments μ_n of the travel time PDF can also be directly calculated from the CDF by means of the following formula:

$$\mu_n(\boldsymbol{x}) = \int_0^{+\infty} n\,t^{n-1}(1 - G(t,\boldsymbol{x}))dt \tag{2.147}$$

2.8.2.9   Capture zone probability

The capture zone probability of a specific outlet can easily be calculated by adapting Eq. (2.130) for the cdf of the life-expectancy-to-outlet Γ_n, $p_n(\boldsymbol{x},t) = \int_0^t g(\boldsymbol{x},u)du$:

$$\frac{\partial \phi p_n}{\partial t} = \nabla \cdot \mathbf{q}p_n + \nabla \cdot \theta \mathbf{D}\nabla p_n - q_{\mathrm{I}}p_n \qquad \text{in} \quad \Omega \tag{2.148a}$$

$$p_n(\boldsymbol{x},0) = 0 \qquad \text{in} \quad \Omega \tag{2.148b}$$

$$[-\mathbf{q}p_n(\boldsymbol{x},t) - \mathbf{D}\nabla p_n(\boldsymbol{x},t)] \cdot \boldsymbol{n} = -\mathbf{q} \cdot \boldsymbol{n} \qquad \text{on} \quad \Gamma_n \tag{2.148c}$$

$$[-\mathbf{q}p_n(\boldsymbol{x},t) - \mathbf{D}\nabla p_n(\boldsymbol{x},t)] \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad \Gamma_+ \tag{2.148d}$$

$$- \mathbf{D}\nabla p_n(\boldsymbol{x},t) \cdot \boldsymbol{n} = 0 \qquad \text{on} \quad \Gamma_0 \tag{2.148e}$$

The field of $p_n(\boldsymbol{x},t)$ defines the probabilistic drainage basin corresponding to the outlet Γ_n, or probabilistic capture zone relative to a particular transit time t. The ultimate, or steady-state probabilistic drainage basin is the field $p_n^\infty(\boldsymbol{x}) = p_n(\boldsymbol{x},\infty)$. Because in both 2D and 3D domains there is a non-zero probability that the water particles will *not* be intercepted by the outlet Γ_n (the only location where this probability is zero), $p_n^\infty(\boldsymbol{x})$ is less than one, $p_n^\infty(\boldsymbol{x}) = \int_0^\infty g(\boldsymbol{x},t)dt < 1 \;\forall \boldsymbol{x} \in \Omega$. An iso-probability value $p_n(\boldsymbol{x},\infty)$ includes the domain for which the fraction $1 - p_n(\boldsymbol{x},\infty)$ of its water amounts will reach the outlet Γ_n, sooner or later.

Chapter 3

Numerical Implementation

3.1 General

HydroGeoSphere uses the control volume finite element method to solve the flow equations for all domains considered in a simulation, 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 3-D porous medium and dual continuum equations are rectangular prisms (8-node elements), 3-D triangular prisms (6-node elements), and 3-D tetrahedra (4-node elements). The 2-D fracture and surface equations are solved for using either rectangular (4-node elements) or triangular elements (3-node elements) and the 1-D well, tile drain and channel equations are solved for 1-D linear elements (2-node elements). For the 3-D and 2-D 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, surface flow, solute transport with a flux-limiter, including density-dependent flow and transport). To solve the non-linear equations, **HydroGeoSphere** 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 surface flow and for transport. We finally present the solution method for non-linear equations.

3.2   Control Volume Finite Element Method

The method of solution for the flow problem is based on the control volume finite element approach [e.g. *Forsyth*, 1991] 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 surface 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 surface 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 Equation 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 Equation 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:

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

The last term on the right-hand side is the integral of the fluid flux q^* normal to the boundary, B, of volume v. Let us assume for clarity that this fluid flux is zero. Use can be made of Equation 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:

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

The relationship in Equation 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 Equation 3.12 becomes:

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

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 Equation 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 Equations 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 surface 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 Equation 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 and Sudicky* [1996]. 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 for both 3-D block and prism elements. 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 and Sudicky* [1996] for details on the implementation. The finite difference form of the needed influence coefficient matrices are provided in *Huyakorn et al.* [1986]. Because of the different number of nodal connections (*i.e.* 7 for finite difference, 27 for finite element for block elements), the finite element method requires nearly four times as much memory to store the coefficient matrix compared to that for the finite difference method. For block elements, 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 surface. 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 3-D porous medium elements and where the 3-D 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:

$$\lambda_{ij+1/2} = k_{rj} \quad \text{if} \quad \gamma_{ij}(h_j - h_i) > 0$$

$$\lambda_{ij+1/2} = k_{ri} \quad \text{if} \quad \gamma_{ij}(h_j - h_i) < 0 \tag{3.22}$$

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). *Forsyth and Kropinski* [1997] 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 2-D equation for flow in fractures is:

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

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

The 2-D 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:

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

where volume v_i is defined for the dual continuum nodes, with an expression similar to Equation 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 3-D 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:

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

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

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

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

3.3.5   Tile Drains

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

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

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

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

3.4 Discretized Surface Flow Equation

The discretized 2-D surface flow equation is:

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

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

$$h_{oi} = d_{oi} + z_{oi} \quad (3.43)$$

and:

$$\gamma_{oij} = \int_a \nabla N_i \cdot \mathbf{K_o} \cdot \nabla N_j \ da \quad (3.44)$$

with interpolation functions N defined for the 2-D surface elements.

The 2-D area associated with a given surface node is given by:

$$a_i = \int_a \phi_o N_i \, da \quad (3.45)$$

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

$$(\lambda_o)_{ij+1/2} = k_{roj} \quad \text{if} \ \ \gamma_{oij} \left(h_{oj} - h_{oi}\right) > 0$$

$$(\lambda_o)_{ij+1/2} = k_{roi} \quad \text{if} \ \ \gamma_{oij} \left(h_{oj} - h_{oi}\right) < 0 \quad (3.46)$$

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 2.52 and 2.53, 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 2.52 and 2.53, $[\partial h_o/\partial s]^{-1/2}$, 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 2.52 and 2.53, along with the depth within the first gradient operator of Equation 2.55, 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.

3.4.1 Channels

Using the control volume finite element method, the discretized equation for flow along a channel is:

$$(A_i^{L+1} - A_i^{L}) \frac{l_i}{\Delta t} = \sum_{j \in \eta_{ci}} (\lambda_c)_{(ij+1/2)}^{L+1} \gamma_{cij} (h_{cj}^{L+1} - h_{ci}^{L+1}) - A\Gamma_c^{L+1} l_i + Q'^{L+1}_{ci} \quad (3.47)$$

where η_{ci} is the set of channel nodes connected to channel node i through the 1-D channel elements and where:

$$h_{ci} = \psi_{ci} + z_{ci} \tag{3.48}$$

and:

$$\gamma_{cij} = \int_l K_c \nabla N_i \cdot \nabla N_j \; dl \tag{3.49}$$

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

The length associated with a given channel node is given by:

$$l_i = \int_l N_i \, dl \tag{3.50}$$

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

$$
\begin{aligned}
(\lambda_c)_{(ij+1/2)} = k_{rcj} \quad \text{if} \quad \gamma_{cij}(h_{cj} - h_{ci}) > 0 \\
(\lambda_c)_{(ij+1/2)} = k_{rci} \quad \text{if} \quad \gamma_{cij}(h_{cj} - h_{ci}) < 0
\end{aligned}
\tag{3.51}
$$

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 surface domains, and one-dimensional line elements represent the wells and the tile drains. When discretizing the domain, nodes forming the fracture or surface 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 surface flow nodes, a choice of common or dual nodes is offered, but the surface flow nodes nevertheless have to coincide with porous medium nodes. Additionally, nodes forming the 3-D 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 surface 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:

$$w_f\Gamma_f^{L+1}a_i = \left[(S_{wf})_i^{L+1} - (S_{wf})_i^L\right]\frac{w_f a_i}{\Delta t} - w_f\sum_{j\in\eta_{f_i}}(\lambda_f)_{(ij+1/2)}^{L+1}\gamma_{f_{ij}}(h_{f_j}^{L+1} - h_{f_i}^{L+1}) \quad (3.52)$$

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

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

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

$$w_f\sum_{j\in\eta_{f_i}}(\lambda_f)_{(ij+1/2)}^{L+1}\gamma_{f_{ij}}(h_{f_j}^{L+1} - h_{f_i}^{L+1}) \quad (3.53)$$

In Equation 3.53, 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 3.52 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 when subsurface and surface flow equations are fully coupled. It is the only approach currently available for coupling the porous medium and second subsurface continuum and it is one of the two options available for coupling the porous medium and surface flow.

3.5.1   Dual-continuum Subsurface Coupling

As mentioned previously, the porous medium equation is discretized everywhere for the 3-D simulation grid. On the other hand, the dual continuum need not be present everywhere in the domain. When dual continuum is simulated, the discretized dual continuum Equation 3.28 is applied only to those grid nodes where dual continuum conditions exist. Because the dual node approach is used to link the porous medium and the second continuum, the dual continuum nodes have the same spatial coordinates than the porous medium nodes. However, at a dual node, there are two unknowns to solve for: the total hydraulic head in the porous medium ($\psi + z$) and that in the second continuum ($\psi_d + z_d$). Fluid exchange between the two domains is provided by the exchange term Ω_d.

After the subsurface flow equations have been assembled into an implicit system of matrix equations, the 3-D dual continuum flow equations are added to the matrix along with their porous medium interactions, and the fully-integrated flow system is solved at each time step.
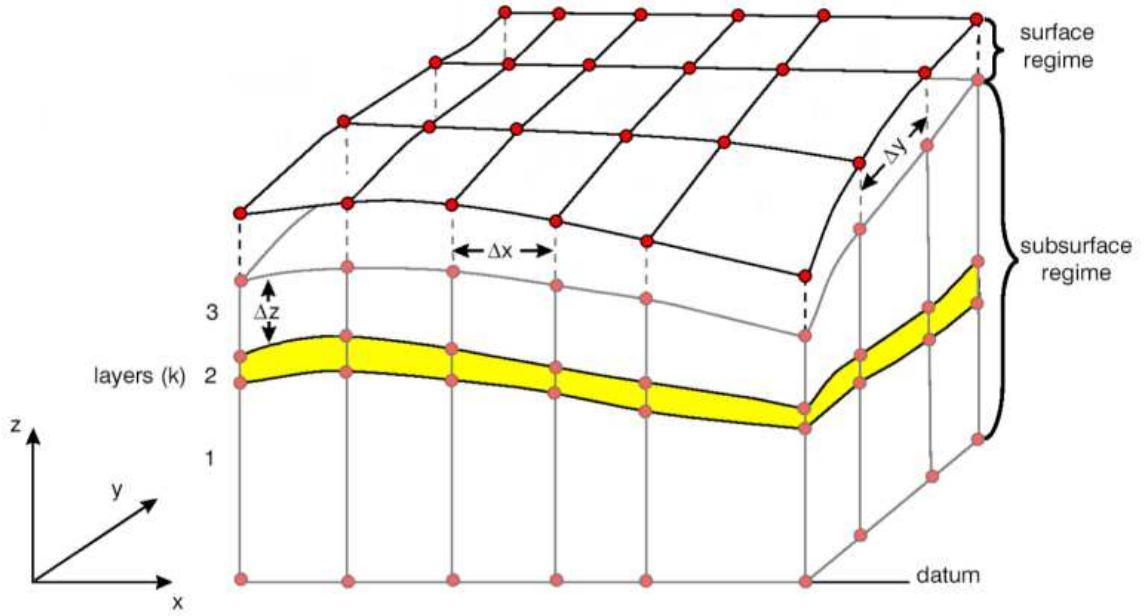
Figure 3.1: Spatial Discretization of the Surface Flow System and its Connection to the Subsurface.

3.5.2 Surface - Subsurface Coupling

The 2-D areal surface flow modules of **HydroGeoSphere** follow the same conventions for spatial and temporal discretizations as those used by the subsurface modules. The surface flow equation is solved on a 2-D finite-element mesh 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. For superposition, the grid generated for the subsurface domain is mirrored areally for the surface flow nodes, with surface flow node elevations corresponding to the top elevation of the topmost active layer of the subsurface grid. Note that surface 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.

Although Figure 3.1 uses 2-D rectangular elements on the surface and 3-D hexahedral elements in the subsurface, **HydroGeoSphere** allows for the use of unstructured grids composed of, for example, 2-D triangles on the surface and 3-D 6 node prismatic or 4-node tetrahedral in elements in the subsurface.

The discretized surface Equation 3.42 is coupled with the 3-D subsurface flow Equation 3.18 via superposition (common node approach) or via leakage through a surficial skin layer (dual node approach). For both approaches, 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 surface 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, **HydroGeoSphere** adds the surface flow equation terms for the 2-D surface mesh to those of the top layer of subsurface nodes in a manner similar to that described in Section 3.5 for the common node approach. In that case, the fluid exchange flux Γ_o, which contains leakance term K_{so} does not need to be explicitly defined.

For subsurface connection via skin leakance, the dual node approach is used and each surface flow node communicates with the first active subsurface flow node directly beneath it to form the subsurface connection. The subsurface flow connection term K_{so} must therefore be added to the left hand side of the respective row for corresponding subsurface flow nodes, in the column that connects them to the surface flow nodes. The interaction flux will also be added to the right hand side of the subsurface flow Equation 3.18. After the subsurface flow equations have been assembled into an implicit system of matrix equations, the 2-D areal surface flow equations are added 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.

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 Equation 3.18 can be manipulated in order to impose prescribed head boundary conditions [*Forsyth*, 1988; *Forsyth and Kropinski*, 1997]. To assign a prescribed pressure head, ψ_b, to a portion of the domain, the source/sink term becomes:

$$Q_i = K_{ij}k_{rw}W_i(\psi_b - \psi_i) \tag{3.54}$$

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:

$$
\begin{aligned}
Q_i = \quad & K^* k_{rw} W_i \left(\psi_{atm} - \psi_i \right) \quad && \psi_i > \psi_{atm} \\
= \quad & 0 \quad && \psi_i < \psi_{atm}
\end{aligned}
\tag{3.55}
$$

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}.

A free drainage boundary is often used for unsaturated flow conditions. It assumes that a unit hydraulic gradient exists along the vertical direction, which results in the following volumetric flow rate out of the domain:

$$
Q_i = K_{zz} k_{rw} A_i
\tag{3.56}
$$

where A_i is the outflow area associated with node i where free drainage occurs.

Drain nodes can be used to simulate fluid flow out of the domain, but without allowing inflow. For a given drain node i, the drain flow rate is given by

$$
\begin{aligned}
Q_i = \quad & C_{DR}(h_i - h_{DR}) \quad && h_i > h_{DR} \\
Q_i = \quad & 0 \quad && h_i \le h_{DR}
\end{aligned}
\tag{3.57}
$$

$$
\tag{3.58}
$$

where C_{DR} is an equivalent conductance $[\text{L}^2 \ \text{T}^{-1}]$ for the drain node and h_{DR} is the drain hydraulic head.

River nodes can be used to simulate fluid flow into or out of the domain. For a given river node i, the flow rate is given by:

$$
Q_i = C_{RIV}(h_i - h_{RIV})
\tag{3.59}
$$

where C_{RIV} is an equivalent conductance $[\text{L}^2 \ \text{T}^{-1}]$ for the river node and h_{RIV} is the river hydraulic head. Assuming the river bed acts as a semipervious layer, the equivalent conductance C_{RIV} is given by:

$$
C_{RIV} = \frac{K_{BED}}{L_{BED}} W_{RCH} L_{RCH}
\tag{3.60}
$$

where K_{BED} is the river bed conductivity $[\text{L} \ \text{T}^{-1}]$, L_{BED} is the river bed thickness $[\text{L}]$, W_{RCH} is the width of the reach $[\text{L}]$ and L_{RCH} is the length of the reach $[\text{L}]$.

3.6.2 Surface Flow

Boundary conditions to the surface 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. Sources/Sinks are applied as net fluxes to the surface 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 surface flow node, subject to similar non-negative depth constraints as discussed for sinks, above.

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 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:

$$Q_o = \frac{1}{n_i} d_o^{5/3} \sqrt{S_o}$$ (3.61)

for the Chezy Equation by:

$$Q_o = C_i d_o^{3/2} \sqrt{S_o}$$ (3.61)

and for the Darcy-Weisbach relation by:

$$Q_o = \sqrt{\frac{8g}{f_i}} d_o^{3/2} \sqrt{S_o}$$ (3.61)

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:

$$Q_o = \sqrt{g d_o^3}$$ (3.62)

3.6.3 Interception and Evapotranspiration

3.6.3.1 Interception

For each time step Δt, the actual interception storage S_{int} is calculated as follows:

$$S_{int}^* = S_{int}^0 + P_p \Delta t - E_p \Delta t$$ (3.63)

with

$$E_{can} = \min(E_p, \frac{S_{int}^0}{\Delta t} + P_p) \tag{3.64}$$

and

$$S_{can} = \min(S_{int}^{Max}, \max[0, S_{int}^*]) \tag{3.65}$$

where S_{int}^0 is the value of S_{int} [L] at the previous time, S_{int}^* is the intermediate value of S_{int} [L], P_p is the precipitation rate [L T^{-1}], E_{can} is the canopy evaporation [L T^{-1}] and E_p is the reference evapotranspiration [L T^{-1}].

E_p may be derived from pan measurements or computed from vegetation and climatic factors (radiation, wind, humidity, and temperature) using the Penman-Monteith equation [*Monteith*, 1981] for vegetated surfaces or a bare-ground evaporation formula [*Sharika et. al.*, 2000] for non-vegetated surfaces, as a function of temperature, wind, and humidity conditions. The reference evapotranspiration is computed efficiently at the start of a simulation, for further use in determining the actual evapotranspiration. The reduced rainfall rate which reaches ground surface after interception is given by:

$$P_p^E = \max[0, P_p - \frac{(S_{int} - S_{int}^0)}{\Delta t} - E_{can}] \tag{3.66}$$

where P_p^E is the effective rainfall rate [L T^{-1}].

Similarly, the potential for evapotranspiration from the soil surface and below is reduced by the canopy evapotranspiration term, which is carried through in the subsequent discussions.

3.6.3.2   Evapotranspiration

Evapotranspiration affects elements and nodes in both surface and subsurface flow domains, and may involve several nodal layers. The rate of transpiration for node i (T_{pi}) can be estimated by substituting the nodal water content θ_i and nodal time-varying root distribution function RDF_i in Equations 2.70 to 2.75.

Equation 2.72 is obtained by expressing Equation 2.74 over the discretized layers of elements containing the roots. As a practical matter, however, the value of RDF at any areal location may be less than one to account for ineffective roots. For each time step, the transpiration (T_p) over the effective root length is calculated as the sum of the transpiration from each of the nodes at depth, as

$$T_p = \sum_{i=1}^{n_R} T_{pi} \tag{3.67}$$

where n_r is the number nodes that lie within the depth interval $0 \le z \le L_r$, at any areal location.

The rate of evaporation for node i (E_{si}) can be estimated by substituting the nodal water content θ_i and nodal evaporation distribution function EDF_i in Equations 2.76 to 2.78.

An appropriate EDF_i for each cell layer may be prescribed as a function of its depth from land surface.

3.7 Elemental Velocities

Elemental velocities or fluid fluxes can be computed after solving for fluid flow. The fluid flux is evaluated by combining its definition:

$$\mathbf{q} = -\mathbf{K} \cdot k_r \nabla(\hat{\psi} + z) \tag{3.68}$$

with the definition of the interpolation function for the unknown. For pressure head, the interpolation function is:

$$\hat{\psi} = \sum_j N_j \psi_j(t)$$

The evaluation of elemental velocities or fluid fluxes is done according to the simple formulas given by Huyakorn et al. [1986] for all types of 1-D, 2-D and 3-D elements available in the model.

3.8 Discretized Solute Transport Equations

3.8.1 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 3-D porous medium, obtained from the application of the control volume finite element method:

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

$$\sum_{j \in \eta_i} \chi_{ij}(C_j^{L+1} - C_i^{L+1}) + (Q_i C_{ups})_i^{L+1} + \left[(R\lambda C_i)_{par} - (\theta_s S_w R \lambda C)_i + \sum \Omega_{\text{ex}}^{L+1}\right] v_i \tag{3.69}$$

where:

$$\begin{aligned} C_{ups} &= & C_i & \text{if} & Q_i < 0 \\ &= & C_{\text{inflow}} & \text{if} & Q_i > 0 \end{aligned} \tag{3.70}$$

and where C_{inflow} is the specified source inflow concentration (recall that Q_i is the source/sink term used to represent boundary conditions). Note that $\mathbf{q}^{L+1}_{(ij+1/2)} = (\lambda)^{L+1}_{(ij+1/2)} \gamma_{ij}(h^{L+1}_j - h^{L+1}_i)$ is the fluid flux at the interface between nodes i and j and is back-calculated from the flow solution.

The term $C_{(ij+1/2)}$ in Equation 3.69 depends on the type of advective weighting used. For central weighting:

$$C_{(ij+1/2)} = \frac{C_i + C_j}{2} \tag{3.71}$$

Upstream weighting gives:

$$
\begin{aligned}
C_{(ij+1/2)} = C_{ups} = C_j \quad &\text{if} \quad \gamma_{ij}\,(h_j - h_i) > 0 \\
C_{(ij+1/2)} = C_{ups} = C_i \quad &\text{if} \quad \gamma_{ij}\,(h_j - h_i) < 0
\end{aligned}
\tag{3.72}
$$

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

$$C_{(ij+1/2)} = C_{ups} + \sigma(r_{ij})\left(\frac{C_{dwn} - C_{ups}}{2}\right) \tag{3.73}$$

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

$$r_{ij} = \left(\frac{C_{ups} - C_{i2ups}}{\|P_{ups} - P_{i2ups}\|}\right)\left(\frac{C_{dwn} - C_{ups}}{\|P_{dwn} - P_{ups}\|}\right)^{-1} \tag{3.74}$$

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 3.73 such that:

$$
\begin{aligned}
\sigma(r) = 0 \qquad &\text{if} \quad r \leq 0 \\
\sigma(r) = (2r)/(1 + r) \quad &\text{if} \quad r > 0
\end{aligned}
\tag{3.75}
$$

We further define:

$$\chi_{ij} = -\int_v \nabla N_i \cdot \theta_s S^{N+1}_w \mathbf{D} \cdot \nabla N_j \, dv \tag{3.76}$$

Other terms in Equation 3.69 are as defined for the discrete flow Equation 3.18.

3.8.2   Discrete Fractures

The discretized 2-D transport equation in fractures is:

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

$$\sum_{j\in\eta_{f_i}} \chi_{fij}(C_{f_j}^{L+1} - C_{f_i}^{L+1}) + \left[(R_f\lambda C_{f_i})_{par} - (S_{wf}R_f\,\lambda\,C_f)_i + \Omega_f^{L+1}\right]a_i \qquad (3.77)$$

where:

$$\chi_{fij} = -\int_a \nabla N_i \cdot S_{wf}^{N+1}\mathbf{D_f} \cdot \nabla N_j\,da \qquad (3.78)$$

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 3.8.1. Other terms in Equation 3.77 are as defined for the discrete fracture flow Equation 3.23.

3.8.3 Double Porosity

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

$$\left[(\theta_{\text{Imm}}C_{\text{Imm}})_i^{L+1} - (\theta_{\text{Imm}}C_{\text{Imm}})_i^L\right]\frac{v_i}{\Delta t} = \Omega_{\text{Imm}}^{L+1}v_i \qquad (3.79)$$

3.8.4 Isotopic Fractionation

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

$$\left[(C_{\text{Imm}})_i^{L+1} - (C_{\text{Imm}})_i^L\right]\frac{v_i}{\Delta t} = \frac{\Omega_{\text{Imm}}^{L+1}}{x_r}v_i \qquad (3.80)$$

3.8.5 Dual Continuum

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

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

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

$$(3.81)$$

All terms in Equation 3.81 are defined in a manner analogous to the discretized porous medium transport equation shown in Section 3.8.1.

3.8.6 Wells

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

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

$$\sum_{j\in\eta_{w_i}} \chi_{wij}(C_{w_j}^{L+1} - C_{w_i}^{L+1}) + (Q_w C_{ups})_i^{L+1} + \pi r_s^2 \left[(\lambda C_{wi})_{par} + \Omega_w^{L+1} \right] l_i \tag{3.82}$$

All terms in Equation 3.82 are defined in a manner analogous to the discretized porous medium transport equation shown in Section 3.8.1.

3.8.7   Tile Drains

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

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

$$\sum_{j\in\eta_{t_i}} \chi_{tij}(C_{t_j}^{L+1} - C_{t_i}^{L+1}) + (Q_t C_{ups})_i^{L+1} + \left[A\,(\lambda C_{ti})_{par} + A\Omega_t^{L+1} \right] l_i \tag{3.83}$$

All terms in Equation 3.83 are defined in a manner analogous to the discretized porous medium transport equation shown in Section 3.8.1.

3.8.8   Surface runoff

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

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

$$+ \sum_{j\in\eta_{o_i}} \chi_{oij}(C_{o_j}^{L+1} - C_{o_i}^{L+1}) + \left[(\phi_o h_o R_o \lambda C_{oi})_{par} - (\phi_o h_o R_o \lambda C_o)_i + \Omega_o^{L+1} \right] a_i \tag{3.84}$$

where:

$$\chi_{oij} = -\int_a \nabla N_i \cdot (\phi_o h_o)^{N+1} D_o \cdot \nabla N_j da \tag{3.85}$$

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 3.8.1.

3.8.9   Channels

One-dimensional transport in a channel is described by the following discretized equation:

$$\left[(C_c)_i^{L+1} - (C_c)_i^L \right] \frac{A l_i}{\Delta t} = \sum_{j\in\eta_{c_i}} (C_c)_{(ij+1/2)}^{L+1} (\lambda_c)_{(ij+1/2)}^{L+1} \gamma_{cij} (h_{c_j}^{L+1} - h_{c_i}^{L+1}) +$$

$$\sum_{j\in\eta_{c_i}} \chi_{cij}(C_{c_j}^{L+1} - C_{c_i}^{L+1}) + (Q_c C_{ups})_i^{L+1} + \left[A\,(\lambda C_{ci})_{par} + A\Omega_c^{L+1} \right] l_i \tag{3.86}$$

All terms in Equation 3.86 are defined in a manner analogous to the discretized porous medium transport equation shown in Section 3.8.1.

3.8.10 Thermal Energy

The numerical implementation of variably-saturated subsurface thermal energy transport is similar to that given by Graf (2005). The discrete form of equation 1 is given by:

$$\sum_{J=1}^{N_n} T_J^{L+1} \left\{ \sum_e \int_{V^e} \left(\frac{\rho_b c_b}{\Delta t} w_I w_J + (q_i \rho_w c_w) w_I \frac{\partial w_J}{\partial x_i} + (k_b + \rho_b c_b D_{ij}) \frac{\partial w_I}{\partial x_i} \frac{\partial w_J}{\partial x_j} \right) dV^e \right\}$$

$$= \sum_{J=1}^{N_n} T_J^{L} \left\{ \sum_e \int_{V^e} \frac{\rho_b c_b}{\Delta t} w_I w_J dV^e \right\} + \sum_{J=1}^{N_n} \left\{ \sum_e \oint_{\Gamma_e} \left((k_b + \rho_b c_b D_{ij}) w_I \frac{\partial T}{\partial n} \right) d\Gamma_e \right.$$

$$\left. + \sum_e \int_{V^e} (\pm Q_T) w_I dV^e + \sum_e \int_{\Gamma_b} \Omega_o w_I d\Gamma_b \right\} \quad i, j = 1, 2, 3 \qquad (3.87)$$

where N_n represents the total number of subsurface nodes, L represents the time level at which a solution is known and $L+1$ represents the time level for which a solution is desired, $\sum_e$ represents the sum over all porous media elements connected to node J, V_e represents the elemental volume, w_I and w_J represent the linear basis functions following the Galerkin approach, Γ_e represents the elemental boundary with n as the unit vector normal to the boundary, and Γ_b represents the elemental boundary between the surface and subsurface domains.

The numerical implementation of the surface thermal energy transport equation is similar to that for the subsurface; however, the surface equation is depth-averaged and is thus evaluated over the elemental area (A_e), not volume. In addition, the surface thermal energy transport equation includes E_{atm} representing the atmospheric inputs to the surface thermal energy system. The discrete equation for the surface regime is given by:

$$\sum_{J=1}^{N_{on}} T_{oJ}^{L+1} \left\{ \sum_{e^o} \int_{A^e} \left(\frac{\rho_w c_w d_o}{\Delta t} w_I w_J + (q_{oi} \rho_w c_w) w_I \frac{\partial w_J}{\partial x_i} + (k_w + \rho_w c_w D_{oij}) d_o \frac{\partial w_I}{\partial x_i} \frac{\partial w_J}{\partial x_j} \right) dA^e \right\}$$

$$= \sum_{J=1}^{N_{on}} T_{oJ}^{L} \left\{ \sum_{e^o} \int_{A^e} \left(\frac{\rho_w c_w d_o}{\Delta t} w_I w_J \right) dA^e \right\} + \sum_{J=1}^{N_{on}} \left\{ \sum_{e^o} \oint_{\Gamma_e} \left((k_w + \rho_w c_w D_{oij}) d_o w_I \frac{\partial T}{\partial n} \right) d\Gamma_e \right.$$

$$\left. + \sum_{e^o} \int_{A^e} (E_{atm} \pm Q_T) w_I dA^e + \sum_{e^o} \int_{\Gamma_b} \Omega_o d_o w_I d\Gamma_b \right\} \quad i, j = 1, 2 \qquad (3.88)$$

where N_{on} represents the number of overland flow nodes, and e^o represents overland elements.

3.9   Travel Time Probability

Since the time-Dirac delta function $\delta(t)$ cannot properly be handled in the time domain, the PDF's are evaluated by solving initial value problems (cf Eqs. (2.134) and (2.135)). The following summarizes the main operations implemented in **FRAC3DVS**:

1. The age/life expectancy/travel time are taken as specific non-reactive species.

2. The input mass is controlled by nodal influence volumes (times mobile water content), according to Eq. (2.134b) or (2.135b). The injection point $\boldsymbol{x}_i$ corresponds to any point in the system for a travel time evaluation from $\boldsymbol{x}_i$ to $\boldsymbol{x}$, to any point on the recharge area Γ_- for the age problem, or to any point on the discharge area Γ_+ for the life expectancy problem.

3. The travel time probability can be calculated:

   - For each element: $\mathbf{q}(\boldsymbol{x}, t)$ and $g(\boldsymbol{x}, t)$ are evaluated at the element centroid, using element shape functions. The control plane section area $A(\boldsymbol{x})$ is evaluated by intersecting the plane normal to velocity with the corresponding element (see Fig. 3.2).

   - At each node: The **S**uper-**C**onvergent **P**atch **R**ecovery method (SCPR, see [*Zienkiewicz*, 1992, *Wiberg*, 1997]) is used to evaluate the nodal velocity (see Fig. 3.3). The fundamental basis of the SCPR method is that there are specific locations in an element where the derivatives are more accurate for a given polynomial degree. These particular locations are referred to as element Super-Convergent points. The SCPR method can briefly be summarized by the following:
     - For a given node, a patch is made of neighboring elements (1D, 2D, 3D);
     - Velocity is evaluated at some sample points within the patch. These sample points correspond to a set of Gauss points;
     - A least-square fit is performed through these samples, and the local problem is solved with a Singular Value Decomposition Factorization solution method.

4. Travel time statistics are post-processed for each element (or node).

3.10   Solute 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.
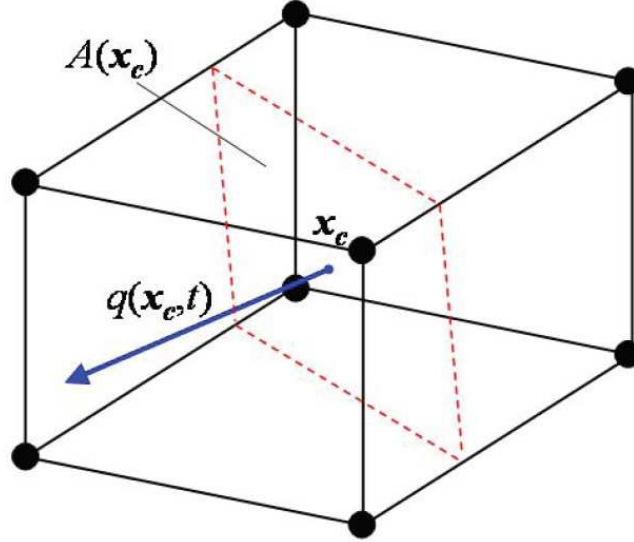
Figure 3.2: Procedure for the evaluation of the travel time PDF at the element centroid, for the case of a brick element.
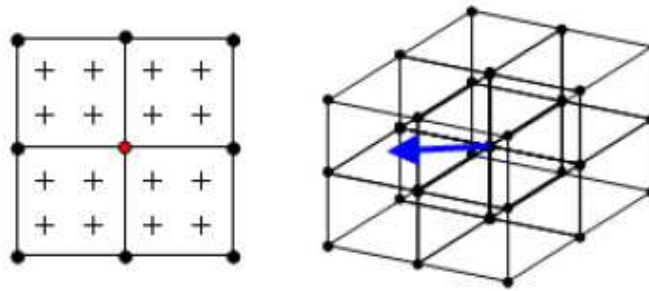
Figure 3.3: Schematic illustration of SCPR method. Left: Four quadrangles defining a patch of elements used to evaluate the velocity at a node (red circle), with indicated sample points (+); Right: Example of a patch made of 8 brick elements.

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:

$$\Omega_d^{L+1} v_i = \left[-(u_m C)_i^{L+1} - (u_d C_d)_i^{L+1} \right] v_i \tag{3.89}$$

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

$$\Omega_o^{L+1} = C_{ups}^{L+1} \Gamma_o \tag{3.90}$$

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

3.11   Solute Transport Boundary Conditions

3.11.1   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.

Some input fluxes and boundary conditions for thermal energy transport were previously incorporated into **HydroGeoSphere** by Graf (2005), such as the prescribed temperature boundaries. Other thermal energy inputs were developed during this research; specifically the temperature flux input and atmospheric inputs.

The temperature input flux is incorporated into **HydroGeoSphere** to account for the thermal energy associated with incoming fluid flux. The temperature input flux can be used in both the surface and subsurface domains. The general equation is given as:

$$Q_{heat_i} = Q(\rho_{in} c_{in} T_{in} - \rho_i c_i T_i) \tag{3.91}$$

where Q_{heat_i} is the temperature flux into node i, Q is the volumetric flux of the carrier fluid, ρ is the density, c is the heat capacity, T is the temperature, in denotes the input term, and i denotes a nodal term. The density and heat capacity terms vary depending on the material associated with the node or input flux (bulk or water). A temperature flux associated with surface water entering the surface domain (for example, a stream inlet) would use ρ and c terms for water for both the input condition (incoming surface water), and the surface node receiving the incoming fluid. However, the injection of heated water into the subsurface would use ρ and c terms for water for the injected fluid, but would use bulk values for the subsurface node.

The atmospheric temperature inputs implemented into **HydroGeoSphere** include prescribed values (i.e. incoming shortwave radiation, air temperature, cloud cover, etc.) or

values determined by a sinusoidal input function. The addition of atmospheric inputs which can vary sinusoidally with time allows for the representation of diurnal fluxes for air temperature, shortwave radiation and wind speed.

3.12 Numerical Techniques

3.12.1 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:

$$Ax = b \tag{3.92}$$

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.

3.12.2   Newton-Raphson Method

The Newton-Raphson technique is used to linearize the non-linear equations arising from discretization of variably-saturated subsurface or surface 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:

$$f_i^r = \left\{ [\theta_s S_w]_i^{L+1} - [\theta_s S_w]_i^L \right\} \frac{v_i}{\Delta t} - \sum_{j \in \eta_i} (\lambda)_{(ij+1/2)}^{L+1} \gamma_{ij} (h_j^{L+1} - h_i^{L+1}) - Q_i^{L+1} \qquad (3.93)$$

where r represents the iteration level. Application of the Newton method to Equation 3.93 produces the following matrix equation:

$$F_{ij}^r \Delta \psi_j^{r+1} = -f_i^r \qquad (3.94)$$

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

In Equation 3.94, the Jacobian matrix, F_{ij}^r, is defined as:

$$F_{ij}^r = \frac{\partial f_i^r}{\partial \psi_j^r} \qquad (3.95)$$

and vector f_i^r represents 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; *Forsyth and Kropinski*, 1997]. Term by term evaluation of the Jacobian can be represented by [*Forsyth and Kropinski*, 1997]:

$$\frac{\partial f_i^r}{\partial \psi_i^r} \approx \frac{f_i^r(\psi_i^r + \epsilon) - f_i^r(\psi_i^r)}{\epsilon} \qquad (3.96)$$

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

Obviously, from Equation 3.96, 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 3.93 also makes it intuitively easy to evaluate the Jacobian numerically. This is because the fluid flow terms appearing in the summation in Equation 3.93 depend only on nodes i and j. *Forsyth and Simpson* [1991] and *Forsyth and Kropinski* [1997] 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 Equations 3.93 and 3.95 as:

$$\frac{\partial f_i^r}{\partial \psi_i^r} = \frac{\partial}{\partial \psi_i^r} \left\{ [\theta_s S_w]_i^r - [\theta_s S_w]_i^L \right\} \frac{v_i}{\Delta t} - \sum_{J \in \eta_i} \frac{\partial}{\partial \psi_i^r} (\lambda)_{(ij+1/2)}^r \gamma_{ij} (h_j^r - h_i^r) - \frac{\partial Q_i^r}{\partial \psi_i^r} \quad (3.97)$$

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

$$F_{ji} = \frac{\partial f_j^r}{\partial \psi_i^r} \quad (3.98)$$

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

$$\frac{\partial f_j^r}{\partial \psi_i^r} = -\frac{\partial}{\partial \psi_i^r} (\lambda)_{(ji+1/2)}^r \gamma_{ji} (h_i^r - h_j^r) \quad (3.99)$$

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:

$$\lambda_{ij} = \lambda_{ji}$$

$$\gamma_{ij} = \gamma_{ji} \quad (3.100)$$

Using Equation 3.100, 3.99 becomes:

$$\frac{\partial f_j^r}{\partial \psi_i^r} = \frac{\partial}{\partial \psi_i^r} (\lambda)_{(ij+1/2)}^r \gamma_{ij} (h_j^r - h_i^r) \quad (3.101)$$

The right hand side of Equation 3.101 is also found in the summation appearing in Equation 3.97. The expression for the diagonal term (Equation 3.97) can therefore be expressed as:

$$\frac{\partial f_i^r}{\partial \psi_i^r} = \frac{\partial}{\partial \psi_i^r} \left\{ [\theta_s S_w]_i^r - [\theta_s S_w]_i^L \right\} \frac{v_i}{\Delta t} - \sum_{j \in \eta_i} \frac{\partial f_j^r}{\partial \psi_i^r} - \frac{\partial Q_i^r}{\partial \psi_i^r} \quad (3.102)$$

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 *Forsyth and Kropinski* [1997] show that for n unknowns and with the summation in Equation 3.93 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.

3.12.3   Primary Variable Substitution

Forsyth et. al. [1995] 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:

IF

$\qquad\qquad S_i > tol_f$ Use ψ_i as primary variable at node i

ELSE IF

$\qquad\qquad S_i < tol_b$ Use S_i as primary variable at node i (3.103)

ELSE

$\qquad\qquad$ Do not change primary variable at node i

ENDIF

with the requirements that:

$$tol_f < 1 \tag{3.104}$$

and:

$$tol_f \neq tol_b \tag{3.105}$$

3.12.4   Time Stepping

A variable time-stepping procedure similar to the one outlined by *Forsyth and Sammon* [1986] and *Forsyth and Kropinski* [1997] has been incorporated in the solution procedure. The time step is defined according to the rate of change of the solution unknown. For flow, the rate of change of hydraulic head, pressure head or saturation can be chosen. For transport, the rate of change of concentration or temperature is used for variable time-stepping.

Let's assume that X represent one of the unknowns mentioned above. After obtaining the solution at time level L, the next time-step is selected according to:

$$\Delta t^{L+1} = \frac{X_{max}}{\max \mid X_i^{L+1} - X_i^L \mid} \Delta t^L \tag{3.106}$$

where X_{max} is a desired maximum change in the unknown during a single time-step, defined by the user.

This implementation of variable time-stepping allows the use of increasingly larger time steps if the dependent variable does not experience drastic changes. 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]).

3.12.5 Mass Balance

Mass balance calculation are done by **HydroGeoSphere** after solving for fluid flow and solute transport. The mass balance calculation gives information to the user on fluid flow and solute exchange through external boundaries. It also provides information on the accuracy of the numerical solution since the discretized flow and transport equations (for example, Equations 3.18 and 3.69 for the porous medium) are in a mass conservative form and are actually expressions for mass balance for the volume associated to a node in the grid. These equations are assembled into a global matrix whose solution is obtained by using iterative methods. Therefore, if the iterative process for solving the matrix converges within a prescribed tolerance, mass balance for each equation should be ensured to the level of precision of the convergence tolerance.

Mass balance for fluid flow and solute transport is performed by first computing the total mass change in the domain for a given time step (for steady-state simulations, there is no global mass change in the domain). The mass entering or leaving the domain through the boundaries or through internal sources and sinks is then computed and is compared to the total mass change, providing the mass balance check for the simulation. The procedure used to compute the mass entering or leaving the domain is similar to that described in *Huyakorn and Pinder* [1983] and involves reassembling the discretized equation for the first-type nodes.

3.12.6 Solution Procedures

The solution methodology for 2-D areal surface flow is embedded into the time looping of the solution methodology for the subsurface calculations of **HydroGeoSphere** (i.e., at each iteration, assembly of the matrix of flow equations for the subsurface is followed by assembly of its surface 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

In order to verify the numerical techniques and demonstrate applicability of **HydroGeo-Sphere** 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 or experimental applications.

4.1 Subsurface Flow

4.1.1 Level 1: Drawdown in a Theis Aquifer

In order to verify the accuracy of **HydroGeoSphere** 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

Table 4.1: Parameter Values for Simulation of Theis Problem.

Parameter	Value	Unit
Pumping rate	4.0×10^{-3}	m s^{-1}
Hydraulic conductivity	0.0023	m s^{-1}
Aquifer thickness	1.0	m
Aquifer storativity	7.5×10^{-4}	
Radial distance to observation point	55	m
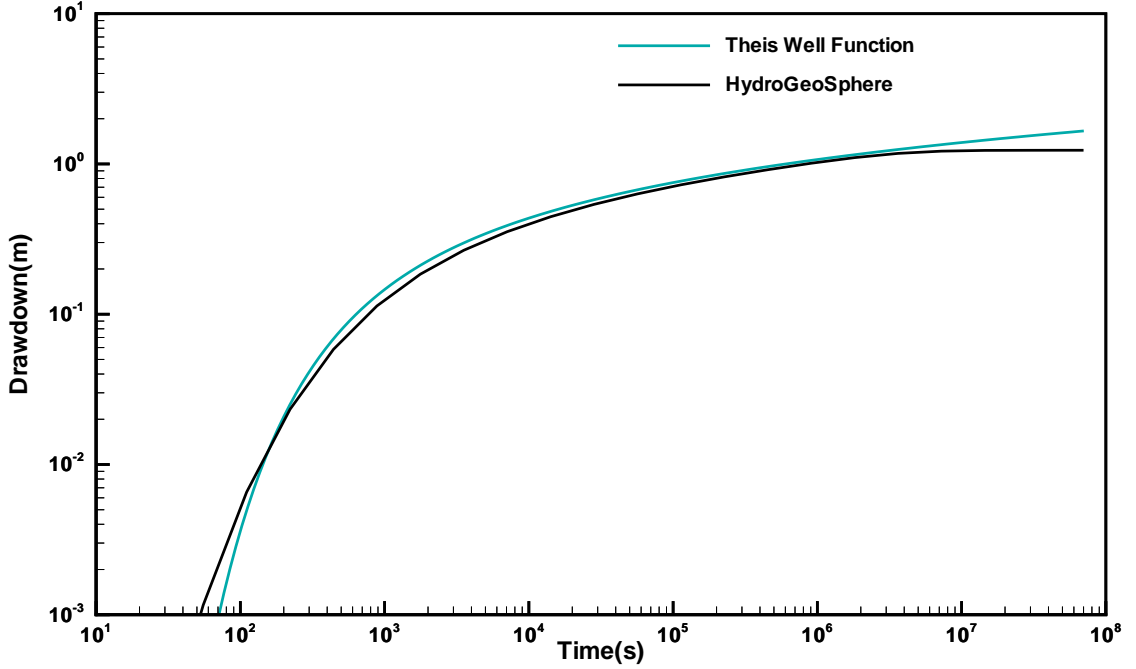
Figure 4.1: Results for Pumping in a Theis Aquifer.

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 Figure 4.1.

The results from both the full 3-D 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   Level 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. The column has dimensions of 50 cm in each of the horizontal (x and y) directions. Initially,

Table 4.2: Water Saturation Versus Pressure Head Relationship for the Unsaturated Column Example.

ψ(cm)	S_w
-0.01	.333
0.0	1.0

Table 4.3: Relative Permeability Versus Water Saturation Relationship for the Unsaturated Column Example.

S_w	K_{rw}
.333	0.0
1.0	1.0

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 d^{-1} is then applied for a period of 10 days. The saturated hydraulic conductivity of the soil is 10 cm d^{-1} 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 **HydroGeoSphere** are also presented. It can be seen that the results are almost identical.

4.1.3 Level 2: 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. The computational domain is shown in Figure 4.3. All boundaries are no flow except for
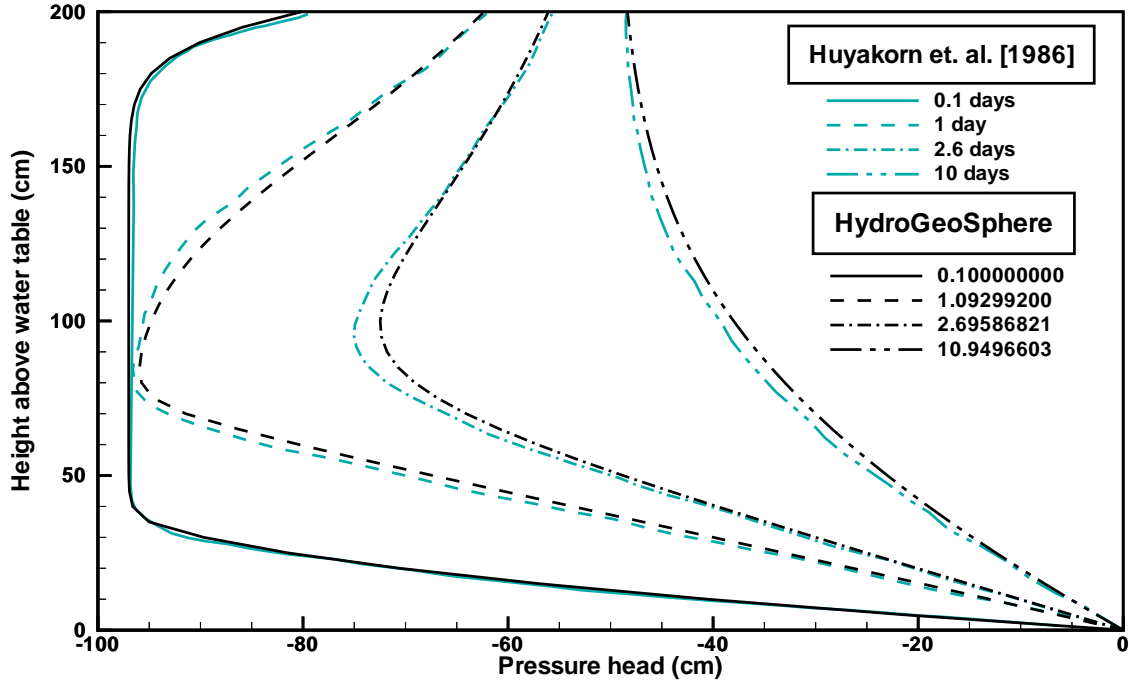
Figure 4.2: Pressure Head Profiles for the Unsaturated Flow Verification Example.

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 **HydroGeoSphere** using upstream weighting and Forsyth's one phase, central weighting case. The saturation front produced by **HydroGeoSphere** is considerably sharper than that shown by Forsyth.

4.1.4   Level 2: 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:

$$k_r(\psi) = \frac{1}{6(4 + \beta b_c)}\{[24 - \exp(-\beta b_s)(24 + 24\beta b_s + 12\beta^2 b_s^2 + 4\beta^3 b_s^3 + \beta^4 b_s^4)]$$
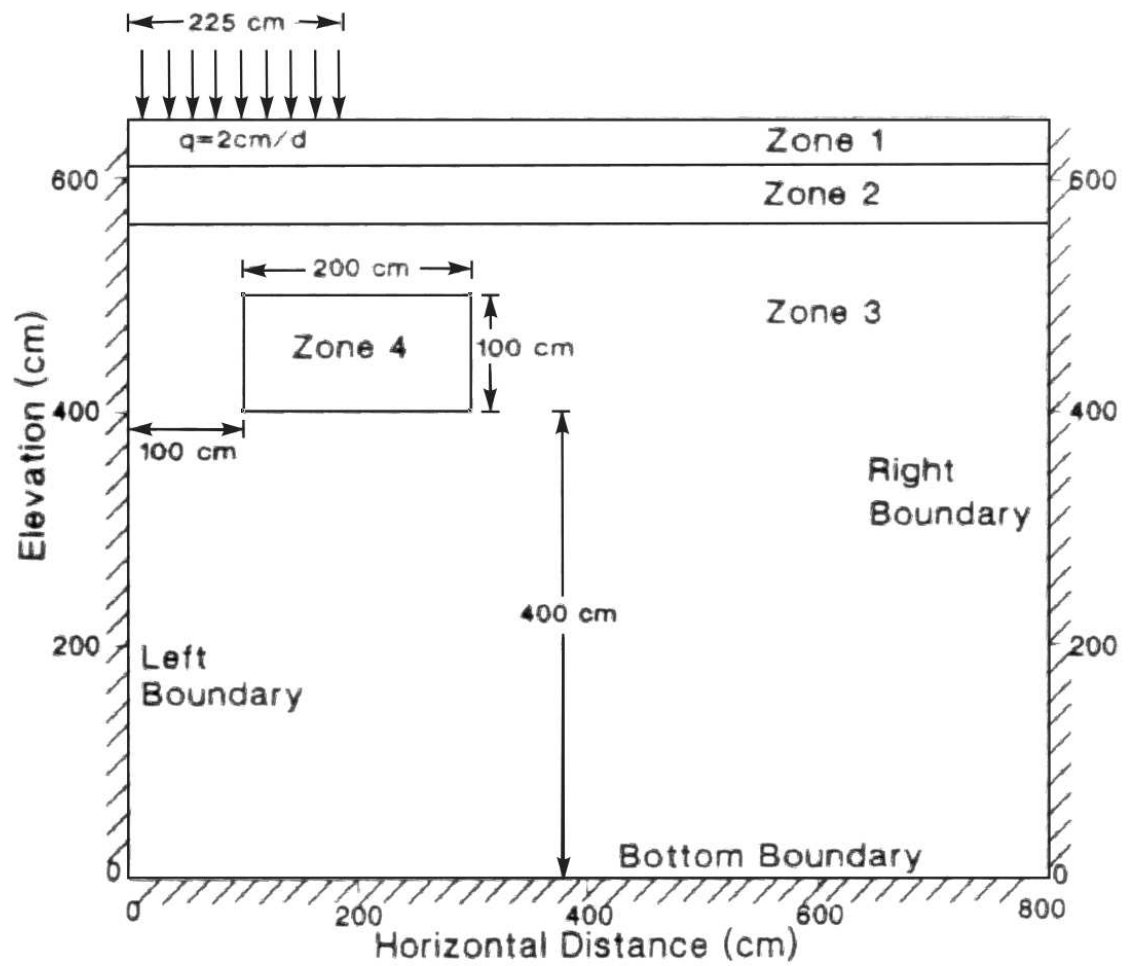
Figure 4.3: Schematic for Very Dry Initial Conditions Problem.

Table 4.4: Material Properties for the Simulation of *Forsyth et al.* [1995], Example 2.

Parameter	Value	Unit
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	
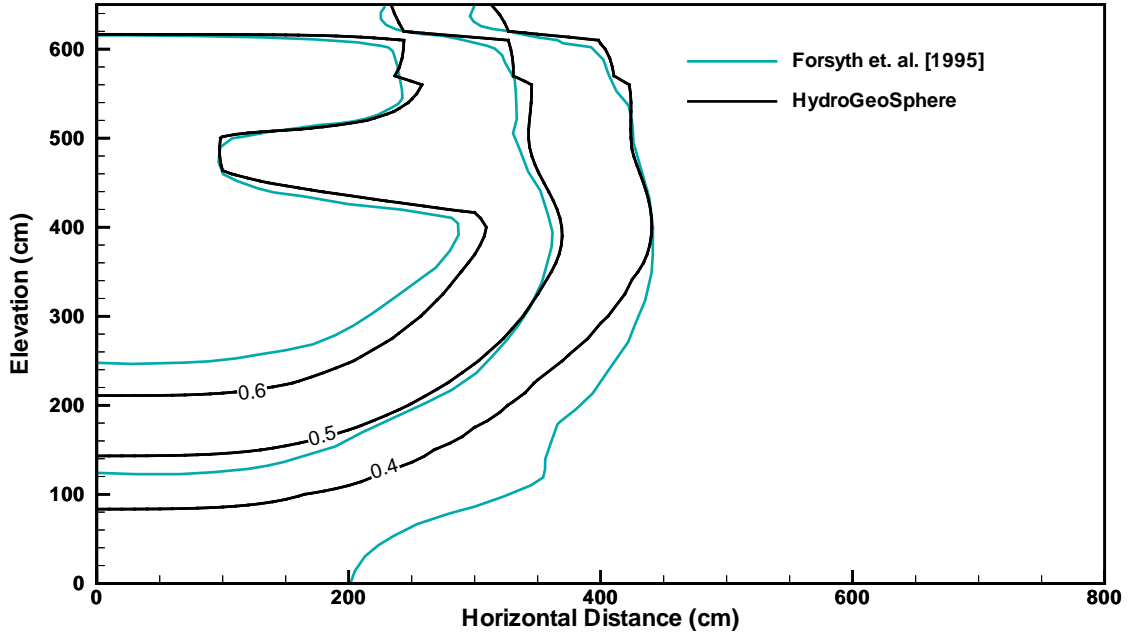
Figure 4.4: Results For Very Dry Initial Conditions Problem

$$+ \beta b_c [6 - \exp(-\beta b_s)(6 + 6\beta b_s + 3\beta^2 b_s^2 + \beta^3 b_s^3)]\} \tag{4.1}$$

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

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

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:

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

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

Table 4.5: Parameter Values Used for *Wang and Narasimhan* [1985] Relationships.

Parameter	Value	Unit
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	m^{-1}
β_V	1.04×10^4	m^{-1}

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 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^{-1}, 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 (Equations 2.7 and 2.8), 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
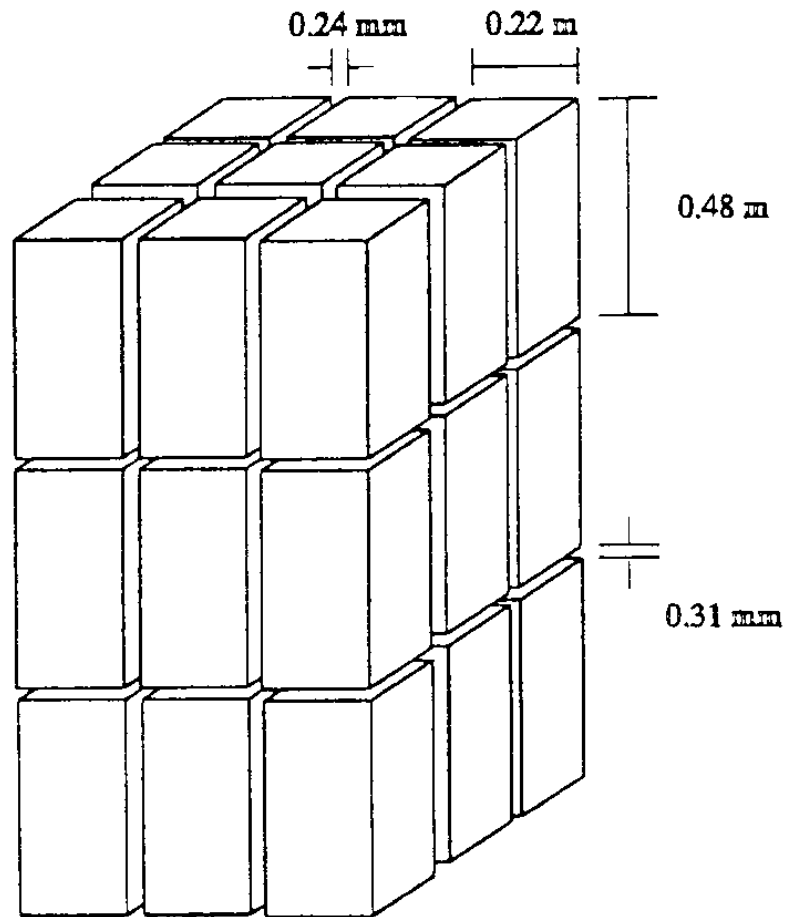
Figure 4.5: Verification Example Involving Fractured Porous Tuff, from *Wang and Narasimhan*, [1985].
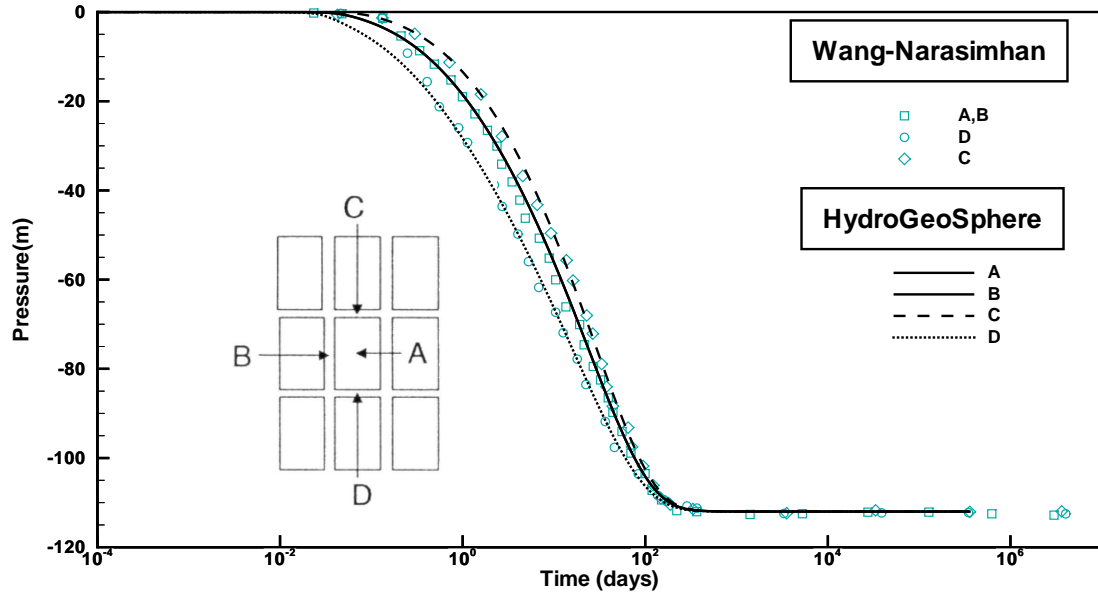
Figure 4.6: Pressure Drop at Selected Points During the Drainage of a Fractured Porous Tuff.

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.

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 Equation 3.106. 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

Table 4.6: Parameter Values for Simulation of 1-D Hydromechanical Coupling Problem.

Parameter	Value	Unit
Hydraulic conductivity	1×10^{-3}	m yr^{-1}
Specific storage	1×10^{-6}	m^{-1}
Loading efficiency	1.0	

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.

4.1.5   Level 1: 1-D Hydromechanical Coupling

This verification example consists of the one-dimensional transient head response to uniform loading on the top of a saturated vertical column. The specifications of the problem are taken from *Lemieux* [2006], who developed an exact analytical solution for a column of semi-infinite length, which is subjected to loading by a mass M, added at constant intervals such that dM/dt is constant and where the top of the column is drained and the base is a no-flow boundary condition.

HydroGeoSphere was used to model this case and was compared to the analytical solution. A domain of 10,000 m length was used. A load of 0.3 m/yr was applied for a period of 10,000 years. Drainage at the top of the column is achieved by specifying a head of 0.0 m at the top of the column. The initial head along the length of the column was set to 0 m. The properties of the rock mass are described in Table 4.6.

Figure 4.7 shows the hydraulic head versus time at different depths in the column from *Lemieux* [2006]. Also shown is the head response from **HydroGeoSphere** at a depth of 500 m below the top of the column. it can be seen that the numerical solution precisely corresponds to the analytical solution.

4.1.6   Level 1: 1-D Hydromechanical Coupling with Externally Computed Stresses

This example is used to demonstrate that **HydroGeoSphere** could be used in conjunction with other hydromechanical models.

In general, because **HydroGeoSphere** cannot generate equilibrated hydromechanical total stresses for use in Equation 2.21, they must be generated by other models and used as input to **HydroGeoSphere**. Theoretically, time-dependent, average, element-specific total stresses are required.

As shown in Figure 4.8, a column of soil of height 1,000 metres supporting a load Φ_{zz} is
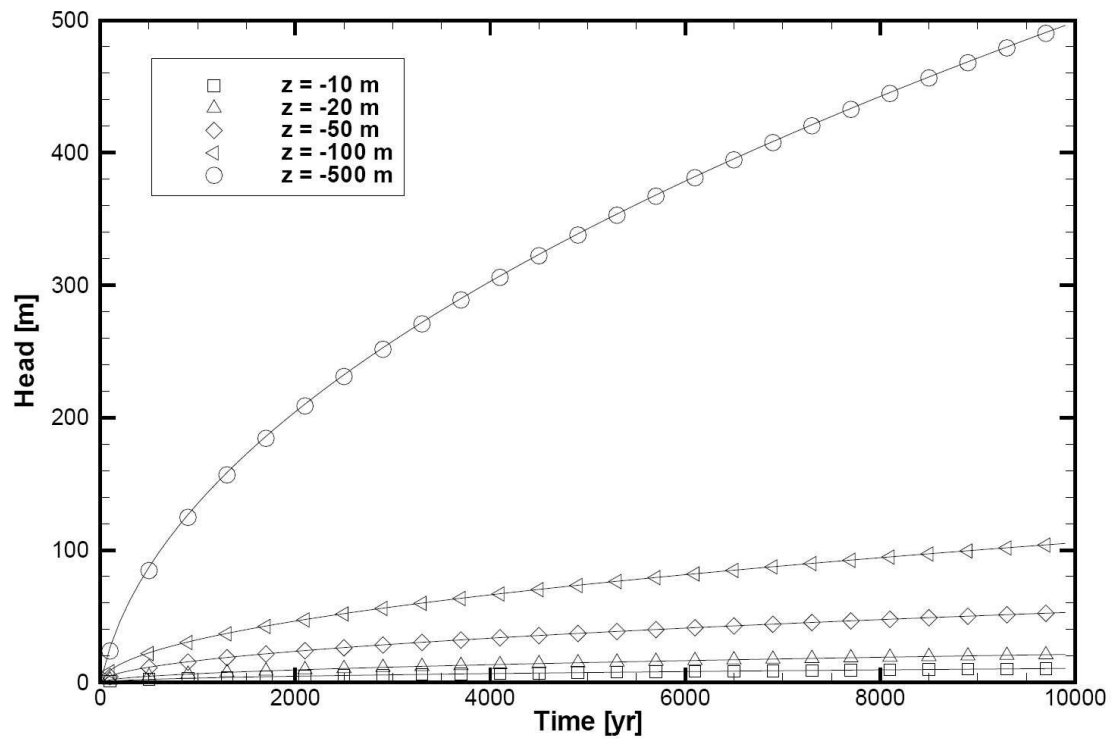
Figure 4.7: Results for 1D Hydromechanical Coupling Example.

Table 4.7: Parameter Values for Simulation of 1-D Hydromechanical Coupling Problem.

Parameter	Value	Unit
Hydraulic conductivity	0.1	m d^{-1}
Specific storage	0.0018	m^{-1}
Loading efficiency	1.0	

confined laterally in a rigid sheath so that no lateral expansion can occur. It is assumed that no water can escape laterally or through the bottom while it is free to escape at the upper surface. The height of 1000 m is assumed to reflect the magnitude of depth that **HydroGeoSphere** is likely to be applied. Material properties for the soil are given in Table 4.7.

A vertical stress of 0.5 MPa was applied to the surface, which causes the water pressure head to rise by 50.9 metres at the onset.

For cases of purely vertical strain such as we have here, *Guvanesan* [2007] presents a method for computing the average hydromechanical stresses at any time from water pressures obtained from an analytical solution developed by *Biot* [1941]. Details of the mathematical theory and procedures are not given here, but instead the reader is referred to *Guvanesan* [2007].

The analytical solution of *Biot* [1941] is based on the assumption that the vertical load is instantaneously applied and remains constant thereafter. In applying **HydroGeoSphere**, it was assumed that the load was increased in a linear fashion from 0 to 50.9 m in 0.1 days.

Average hydromechanical stresses versus time (expressed as water pressure in m) computed by *Guvanesan* [2007] at various elevations are plotted in Figure 4.9. These stresses were used as input to **HydroGeoSphere**

Figure 4.10 shows the hydraulic head versus time at different elevations in the column from *Biot* [1941] and **HydroGeoSphere**. The agreement between **HydroGeoSphere** and the analytical solution is an indication that **HydroGeoSphere** can be interfaced with other hydromechanical models.

4.2   Surface Flow

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

The surface water flow capabilities of **HydroGeoSphere** are verified against analytical and numerical solutions of diffusive wave and dynamic wave equations, presented by *Govindaraju et al.* [1988a and 1988b]. The problem considered involves one-dimensional surface flow along a plane of one unit width (Figure 4.11). 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
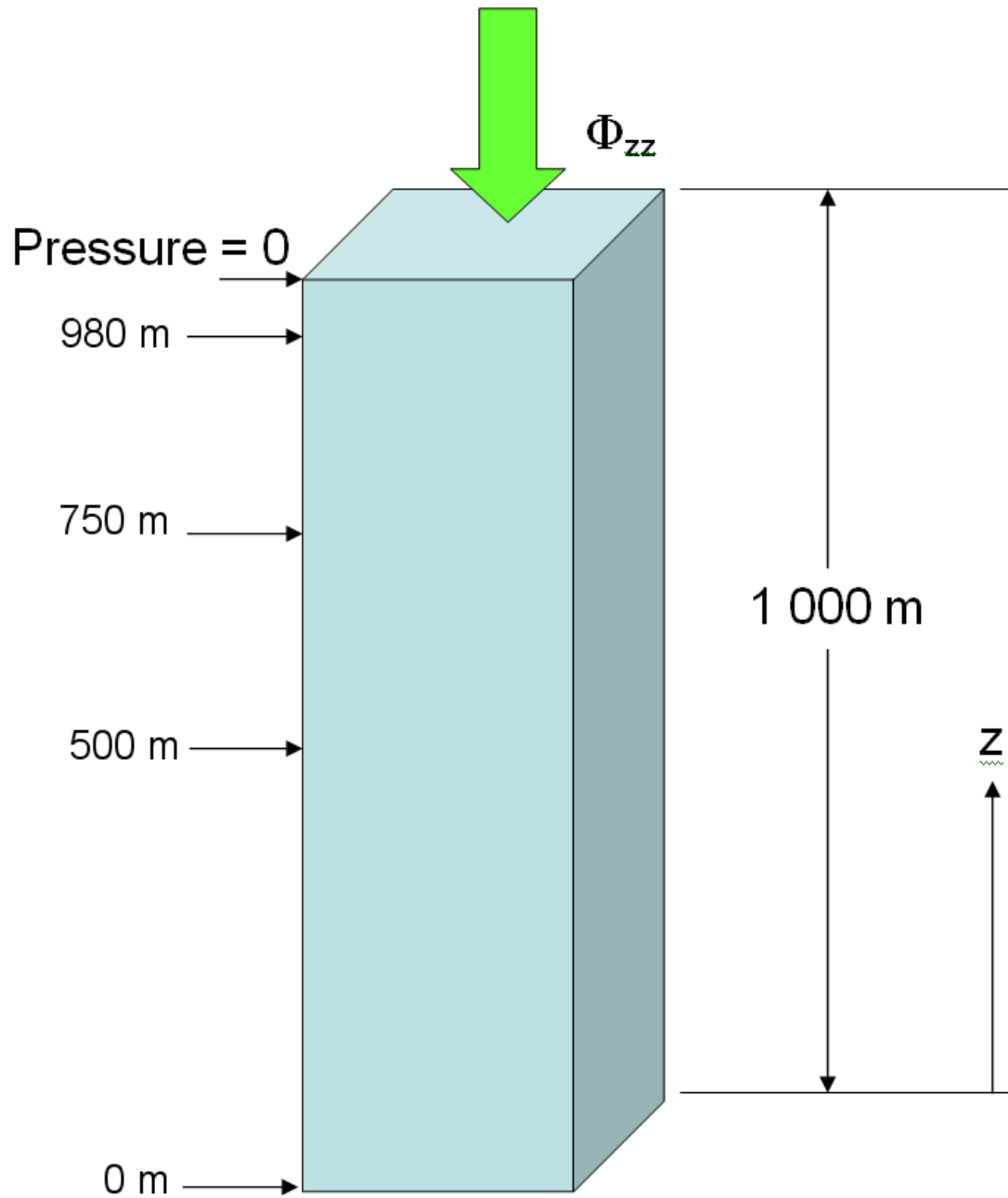
Figure 4.8: Schematic for 1D Hydromechanical Coupling with External Stresses Example.
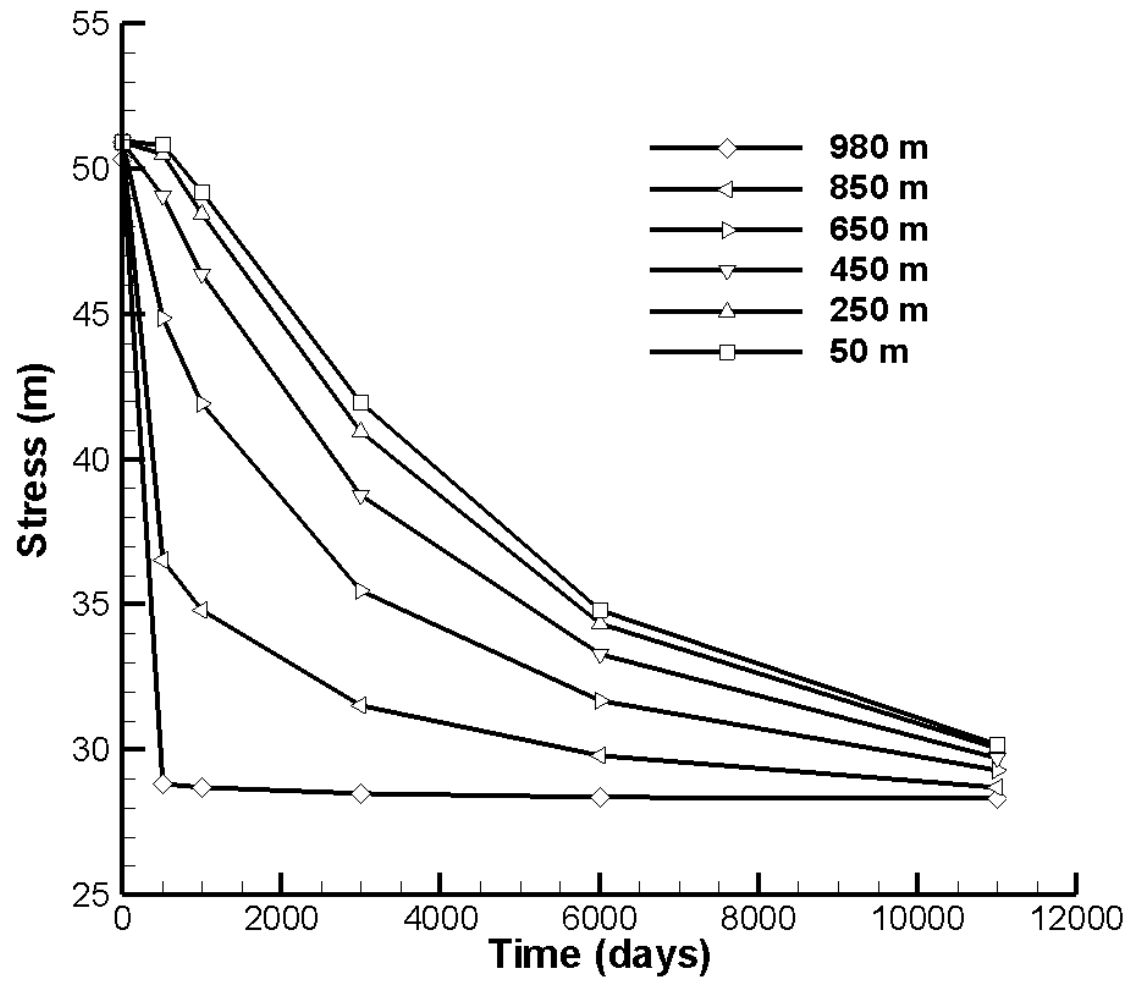
Figure 4.9: Mean Normal Stress Versus Time at Various Elevations.
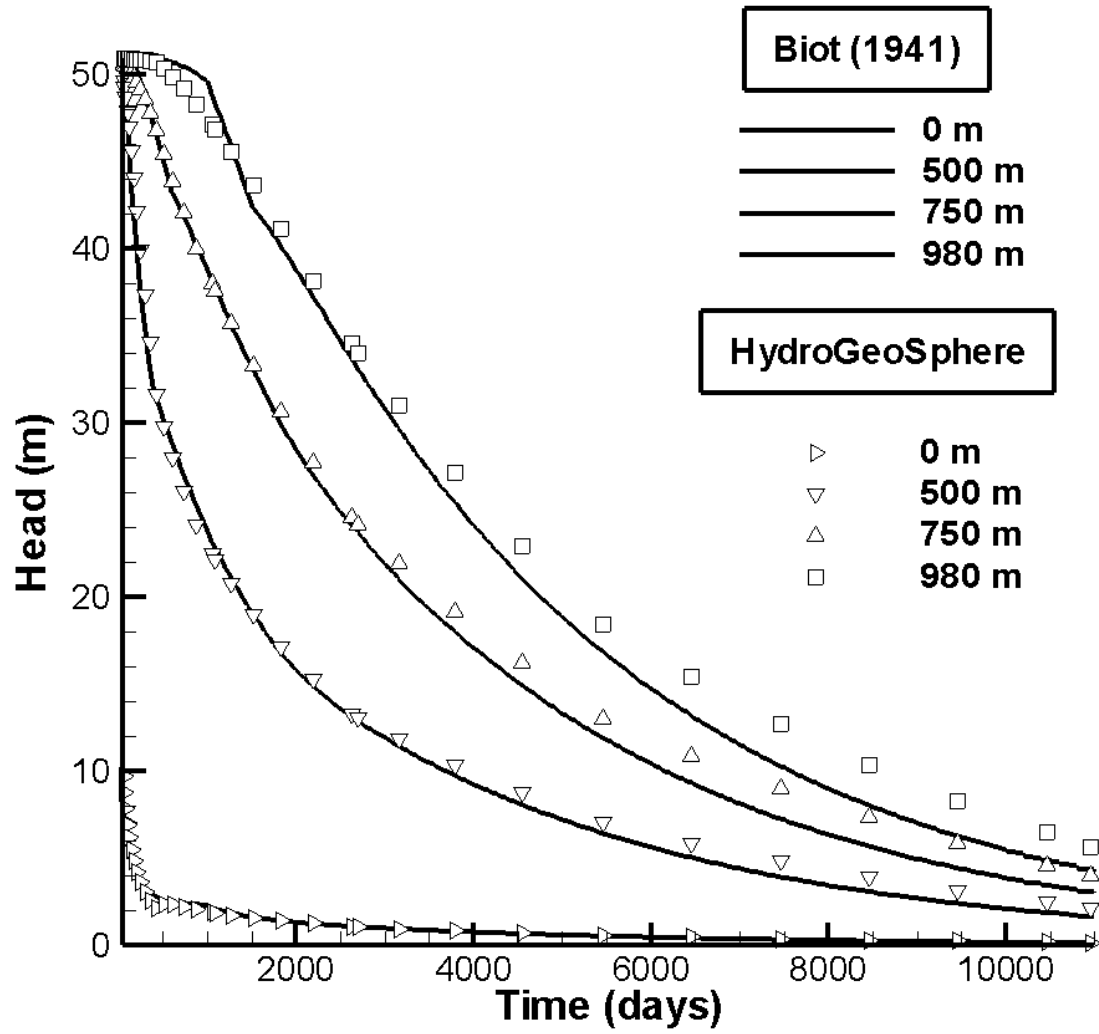
Figure 4.10: Results for 1D Hydromechanical Coupling with External Stresses Example.
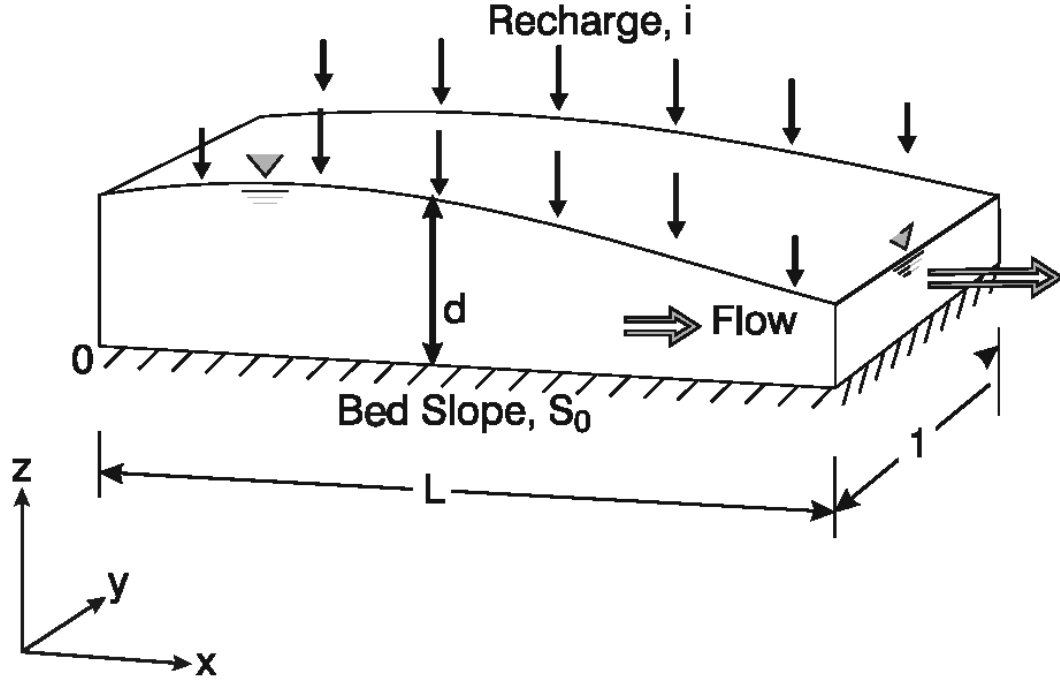
Figure 4.11: Schematic Description of the *Govindaraju et al.* [1988a and 1988b] Problem.

($F_o = 1.5$) and at different kinematic wave numbers $K (= S_o L / d_o F_o^2)$ ranging from 3 to 50 ($K > 20$ indicates a kinematic wave), 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.

We chose a single case for comparison, using the parameters shown in Table 4.8.

A zero-depth gradient boundary condition (see Equation 3.61) was applied at the downstream end of the system.

Modeling Approach and Results

The slope was discretized into 100 columns and 1 row of elements with dimensions 1m $\times$ 1m each. Only surface 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 and a maximum time-step size of 2.5s. Newton iteration considerations include a maximum of 20 iterations an absolute convergence tolerance of 10^{-4}. Figure 4.12 shows the discharge hydrograph at the downstream end of the slope. The hydrograph was normalized by dividing the discharge by the normal discharge Q_o, in this case 0.3962 m^3 s^{-1}, and time by $t_o (t_o = L / u_o)$.

The results obtained using **HydroGeoSphere** are in good agreement with both the diffusion wave solution of *Govindaraju et al.* [1988a and 1988b], where Picard and Newton iterative approaches provide almost identical solutions, and the Modflow-Surface solution. However, due to the limitations of the diffusive wave approximation, the solution deviates

Table 4.8: Parameter Values for Simulation of a 1-D Flow Example from *Govindaraju et al.* [1988a and 1988b].

Parameter	Value	Unit
Froude number F_o	0.5	
Kinematic wave number K	10	m s^{-1}
Slope S_o	0.01	
Slope length L	100	m
Uniform recharge i_o	0.0040	m s^{-1}
Uniform velocity u_o	0.9905	m s^{-1}
Uniform depth d_o	0.4000	m
Manning's roughness coefficient n	0.0548	$\text{s m}^{-1/3}$
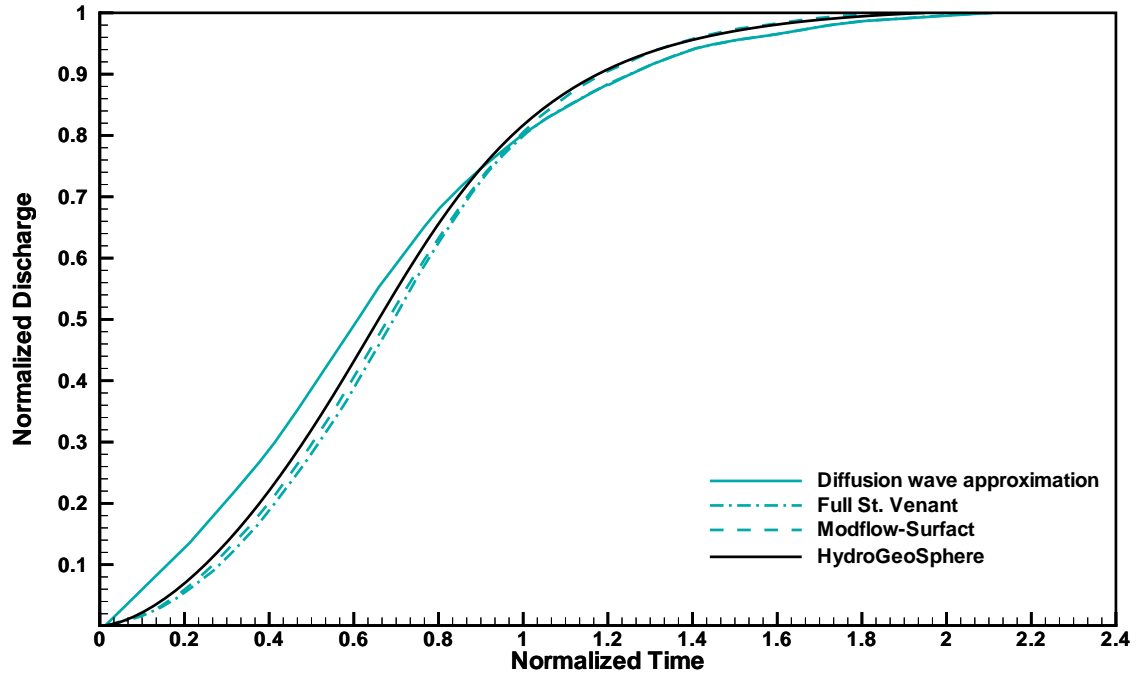
Figure 4.12: Comparison of Normalized Rising Hydrographs for Saint Venant Equations, the Diffusion Wave Approximation [*Govindaraju et al.*, 1988a] and MSVMS for $F_o = 0.5$ and $K = 10$.

from the dynamic wave (i.e., the full Saint Venant solution) for the small value of the kinematic wave number K used in this example.

4.2.2 Level 2: Conjunctive Surface-Subsurface Flow Study of *Smith and Woolhiser,* [1971]

Smith and Woolhiser [1971] present an experimental study of combined surface and subsurface flow which may be used to validate the surface and subsurface flow routines, and their interactions, in **HydroGeoSphere**. The experiments consisted of providing rainfall of 25 cm hour^{-1} 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 4.13. Several conjunctive models of surface and subsurface flow have selected this experiment as a test case. *Smith and Woolhiser* [1971] present a model using the kinematic wave approximation for surface flow. *Akan and Yen* [1981] and *Singh and Bhallamudi* [1998] solve the full dynamic wave equation for surface flow. *Govindaraju and Kavvas* [1991] solve the diffuse wave equation for surface flow for this case. All above models solve the Richards equation for flow in the subsurface, and use flux coupling between the surface 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 that adjusted the saturated moisture content and saturated hydraulic conductivity values. Figure 4.14 shows the observed moisture retention curve for soil layer 1, as provided by *Smith and Woolhiser* [1971], and the fitted Van Genuchten functions, and indicates that the Van Genuchten parameters used for the simulation provide a good fit to the experimental data in the range of suction valid for this study. Figure 4.15 shows the observed and fitted relative hydraulic conductivity vs. moisture content curves for soil layer 1. The fitted Van Genuchten functions for the other two soil layers, not shown here, also had very good fits. *Singh and Bhallamudi* [1998] provide details of the Brooks-Corey relations fitted to the data of *Smith and Woolhiser* [1971]. Table 4.9 provides a summary of the fitted Van Genuchten parameter values of the simulation for all 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^{-1}. The average discharge for the experiments was approximately 1×10^{-5} m^3 s^{-1} for a flume width of 0.051 m, giving $q = 10^{-5}/0.051 = 1.96 \times 10^{-4}$ m^2 s^{-1} 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$^{1/2}$ minute^{-1} and the Manning constant as $n = d^{1/6}/C = 0.000180d^{1/6}$ minutes cm$^{-1/3}$.
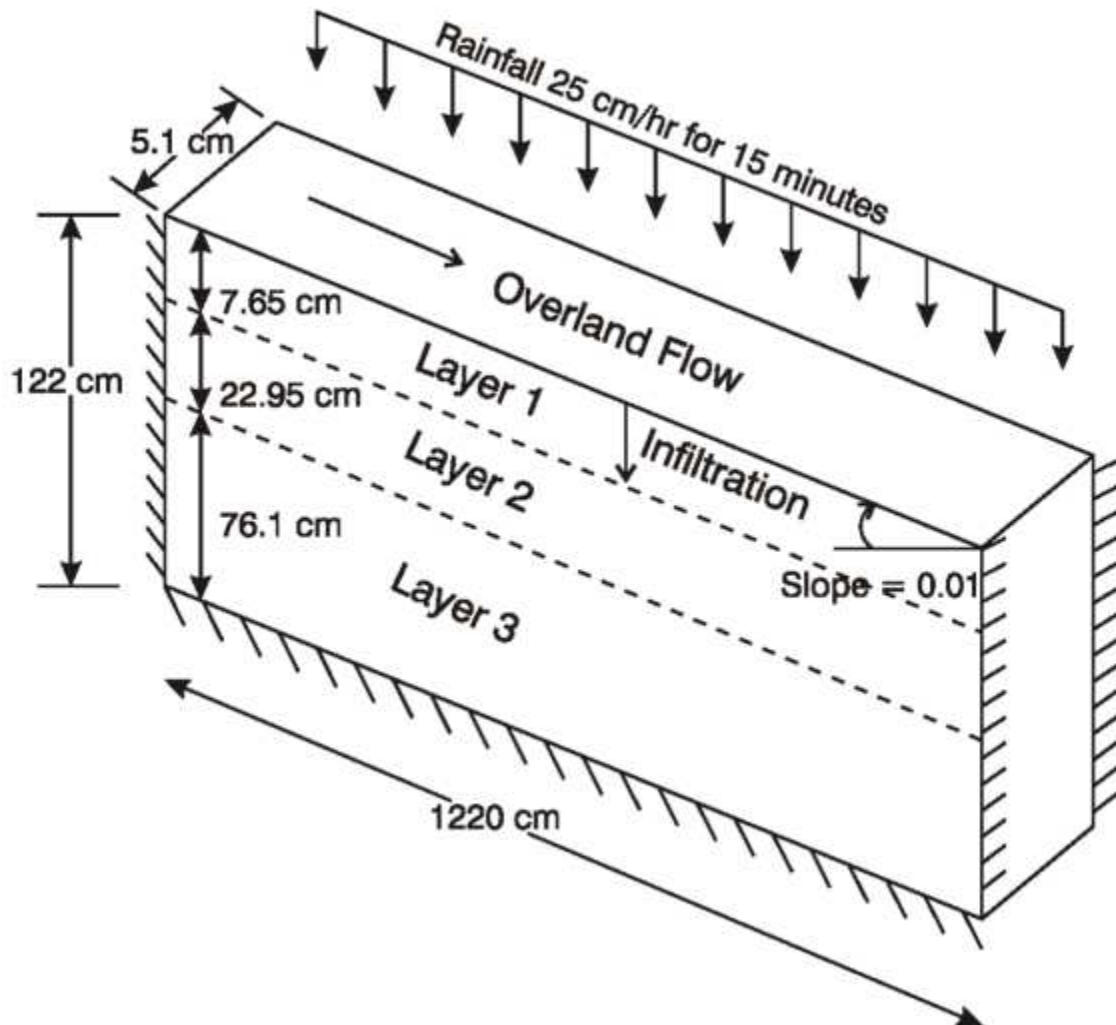
Figure 4.13: Experimental Setup of the *Smith and Woolhiser* [1971] Study.
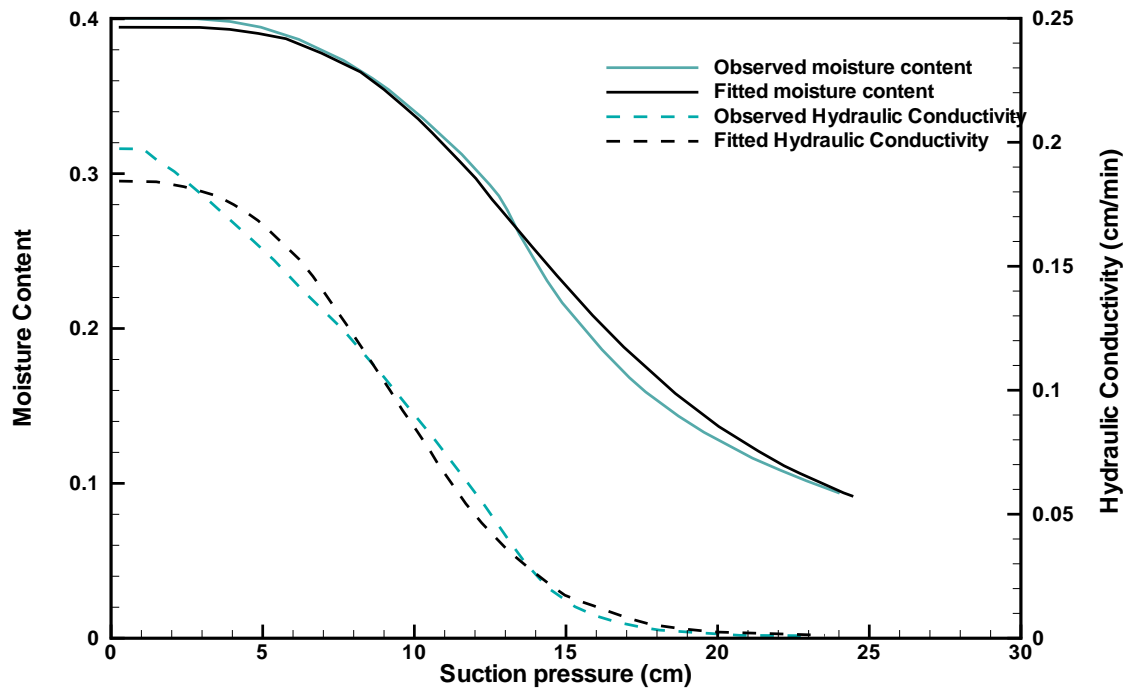
Figure 4.14: Moisture Retention Curve for Soil Layer 1.
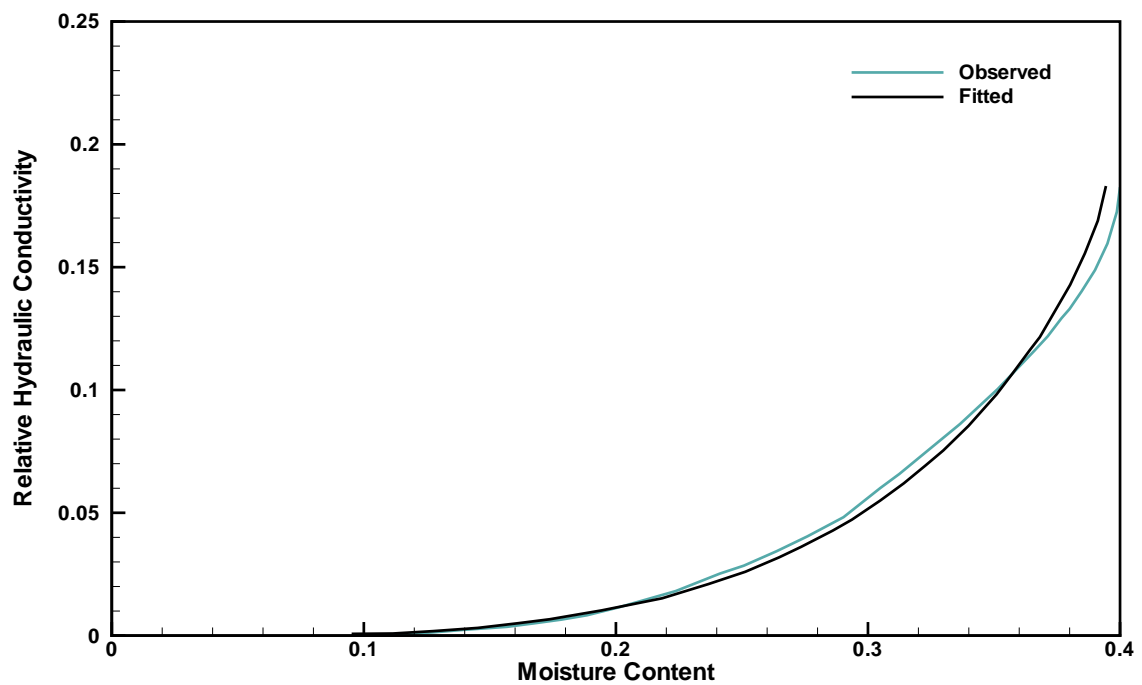
Figure 4.15: Hydraulic Conductivity versus Soil Moisture for Soil Layer 1.

Table 4.9: Parameter Values for Simulation of the *Smith and Woolhiser* [1971] Experiment.

Parameter	Value	Unit
Soil layer 1 [†]		
porosity, n	0.3946	
saturated conductivity, K_s	0.184	cm minute^{-1}
Van Genuchten parameter, α	0.07	cm^{-1}
Van Genuchten parameter, β	3.4265	
residual saturation, S_r	0.05068	
Soil layer 2		
porosity, n	0.4387	
saturated conductivity, K_s	0.1452	cm minute^{-1}
Van Genuchten parameter, α	0.056	cm^{-1}
Van Genuchten parameter, β	4.1371	
residual saturation, S_r	0.05699	
Soil layer 3		
porosity, n	0.4764	
saturated conductivity, K_s	0.1296	cm minute^{-1}
Van Genuchten parameter, α	0.0443	cm^{-1}
Van Genuchten parameter, β	4.3565	
residual saturation, S_r	0.05248	
Darcy-Weisbach friction, f_r	0.91	
Equivalent Chezy coefficient, C_c	5569.1	cm$^{1/2}$ minute^{-1}
Equivalent Manning coefficient, n	0.000122	minute cm$^{-1/3}$
Rainfall intensity, i	0.417	cm minute^{-1}
Channel slope	0.01	
Channel length	1220	cm
Channel width	5.1	cm
gravitational acceleration	3.532×10^6	cm minute^2

[†] The soil parameters provide the fit to experimental data of *Smith and Woolhiser* [1971], as shown in Figures 4.14 and 4.15 for soil layer 1.

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 minutes cm$^{-1/3}$.

Modeling Approach and Results

The two-dimensional domain was discretized into 100 columns, 1 row and 40 layers of elements. 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 length of each element was 12.2 cm with a width of 5.1 cm and a thickness of 1.53 cm in soil layers 1 and 2, and a thickness of 3.805 cm in soil layer 3. The entire rectangular domain is tilted along its length to provide a slope of 0.01 to the model domain. The dual node approach, with a coupling length of 1.35 m and a rill storage height of 0.01 m was used to couple the surface and subsurface flow systems. Rainfall at a rate of 0.416667 cm minute^{-1} is supplied for a duration of 15 minutes followed by 5 minutes with zero recharge. Adaptive time-stepping uses an initial time-step size of 0.01 minutes and a maximum time-step size of 1 minute. Newton iteration considerations include a maximum of 18 iterations an absolute convergence tolerance of 10^{-3} cm. A critical-depth gradient boundary condition is applied at the downstream end of the surface 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 initial water depth for the surface flow nodes was set to 1×10^{-4} m to represent dry starting conditions at the surface. Simulations were performed using the Newton Raphson scheme, as well as using Manning's equations to represent surface flow conditions.

Figure 4.16 shows the saturation profiles within the soil at a distance of 550 cm from the upstream end at different times, and show that infiltration causes the saturation front to advance within the soil. The infiltration front predicted by **HydroGeoSphere** compares fairly well with other simulation attempts [*Singh and Bhallamudi* 1998; *Smith and Woolhiser*, 1971] and with available experimental data.

Figure 4.17 shows the outflow hydrograph at the downstream end of the surface 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].

Figure 4.18 shows the surface water depth profiles at 8.3, 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.

Figure 4.19 shows key components of the fluid balance for the simulation. The data are expressed as flow rates in centimetres cubed per minute, so, for example, the rainfall rate shows up as a uniform value of 2592 cm^3 minute^{-1} until a time of 15 minutes, when it drops to zero. Initially, most of the rainfall is taken up by the subsurface, and accumulation in the surface flow domain is negligible before a time of about 3 or 4 minutes. Accumulation in the
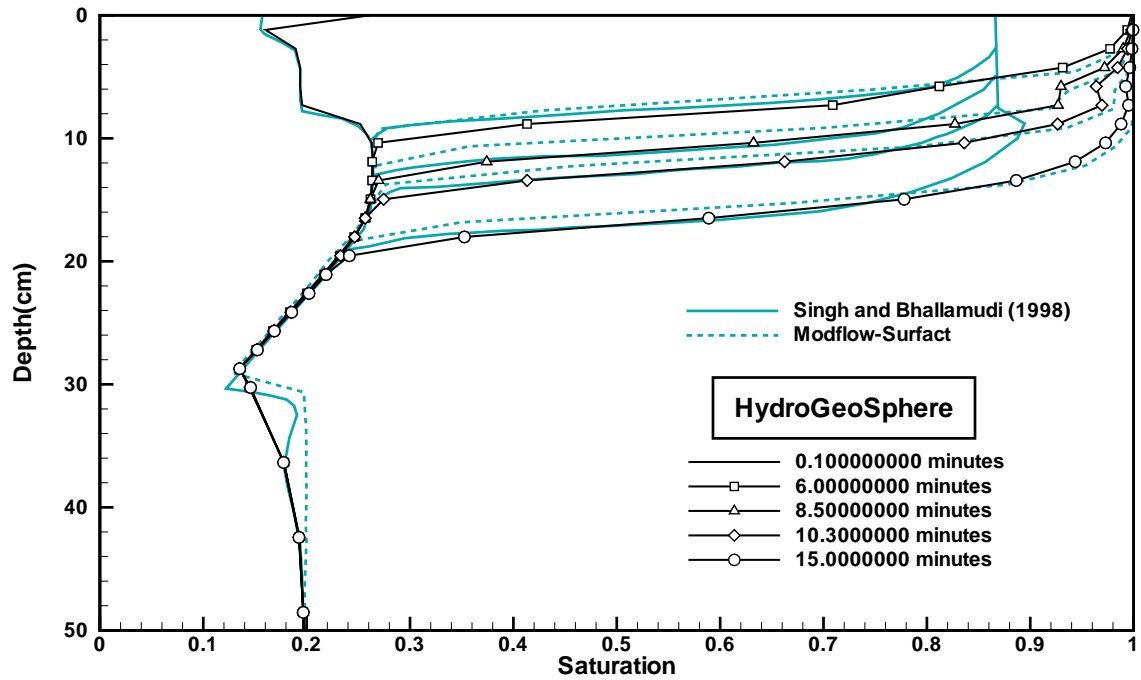
Figure 4.16: Soil Saturation Profile at 550 cm from Upstream End for Simulation of the *Smith and Woolhiser* [1971] Study.
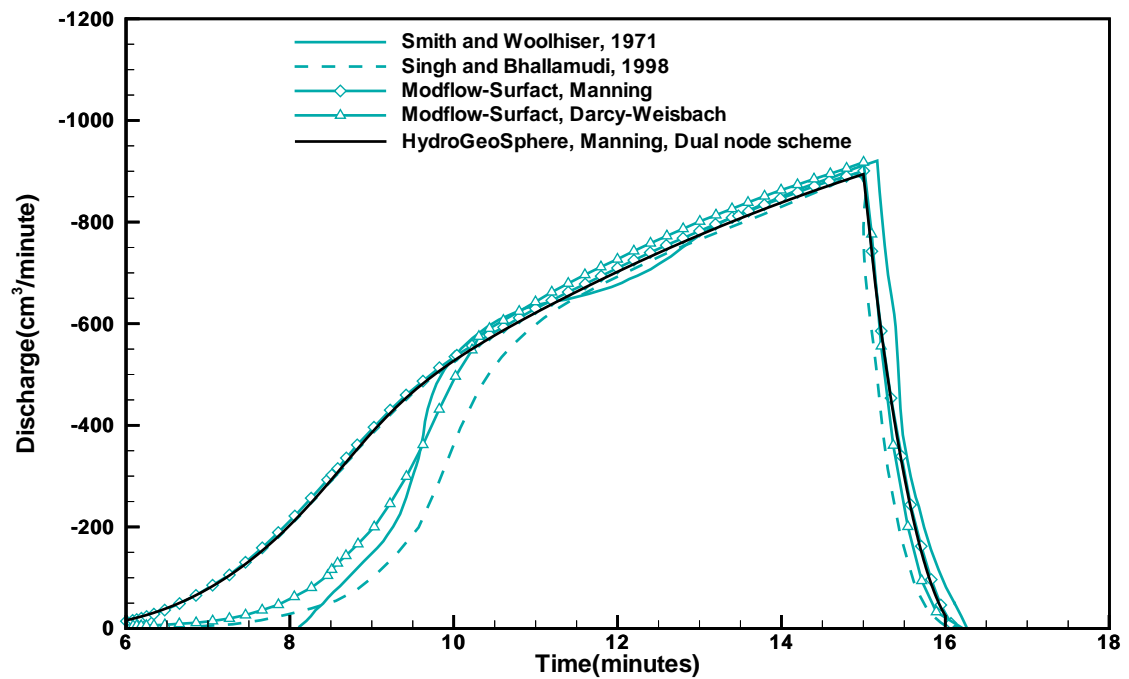
Figure 4.17: Outflow Hydrograph for Simulation of the *Smith and Woolhiser* [1971] Study.
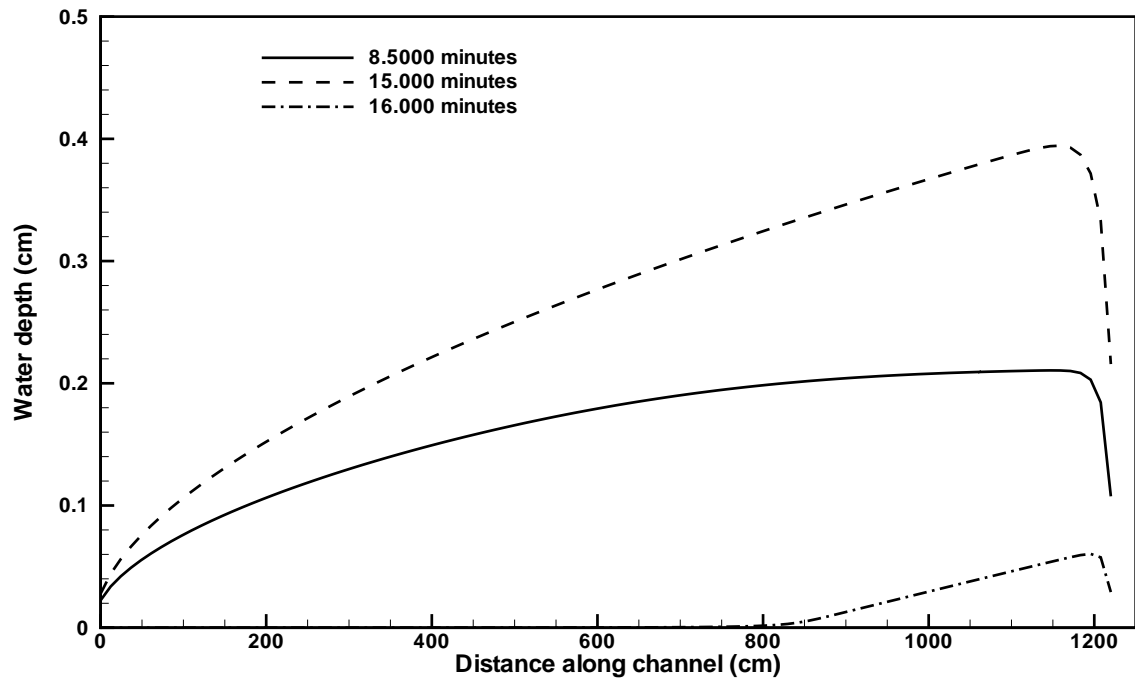
Figure 4.18: Surface Water Depth Profiles at Different Times for the Simulation of the *Smith and Woolhiser* [1971] Study.
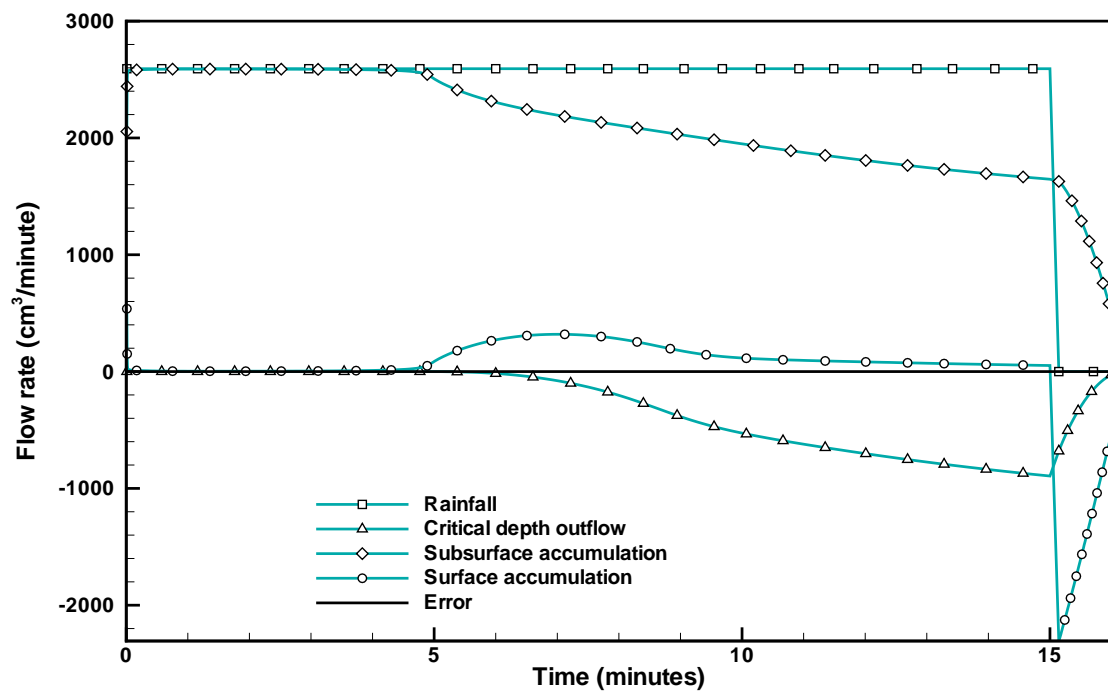
Figure 4.19: Fluid Balance Results for the Simulation of the *Smith and Woolhiser* [1971] Study.

surface flow domain peaks at about 7 minutes and then declines as water begins discharging at the critical depth boundary. Fluid balance errors, which in this case are the difference between the rainfall plus the critical depth accumulation rates and the subsurface plus the surface domain accumulation rates, were less than a fraction of a percent throughout the simulation.

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

Two-dimensional areal surface flow is verified 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 surface flow from a tilted V-catchment (Figure 4.20) generated by a 90 minute duration, 3×10^{-6} m s^{-1} intensity rainfall event. Only one half of the domain need be simulated due to symmetry, with published outflow discharge halved, 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 0.15 and 0.015 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 10 rows and 9 columns, with the last column representing the 10m wide channel. The remaining grid blocks are 100m $\times$ 100m. Only surface flow was simulated, with rainfall applied at the rate of 3×10^{-6} m s^{-1} for 90 minutes, followed by no rainfall 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. Newton iteration considerations include a maximum of 20 iterations an absolute convergence tolerance of 10^{-4}.

Figure 4.21 compares predicted discharge from **HydroGeoSphere** with predictions from several other codes. Excellent agreement is noted between all results. Figure 4.22 shows the channel stage at two points at the outlet: one near the centre of the channel and one near the edge of the channel. The channel stage from Modflow-Surfact, a cell-centred finite difference model, is seen to fall within the edge and centre channel stages of **HydroGeoSphere**. 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 iterations, with negligible mass balance errors throughout the simulation.
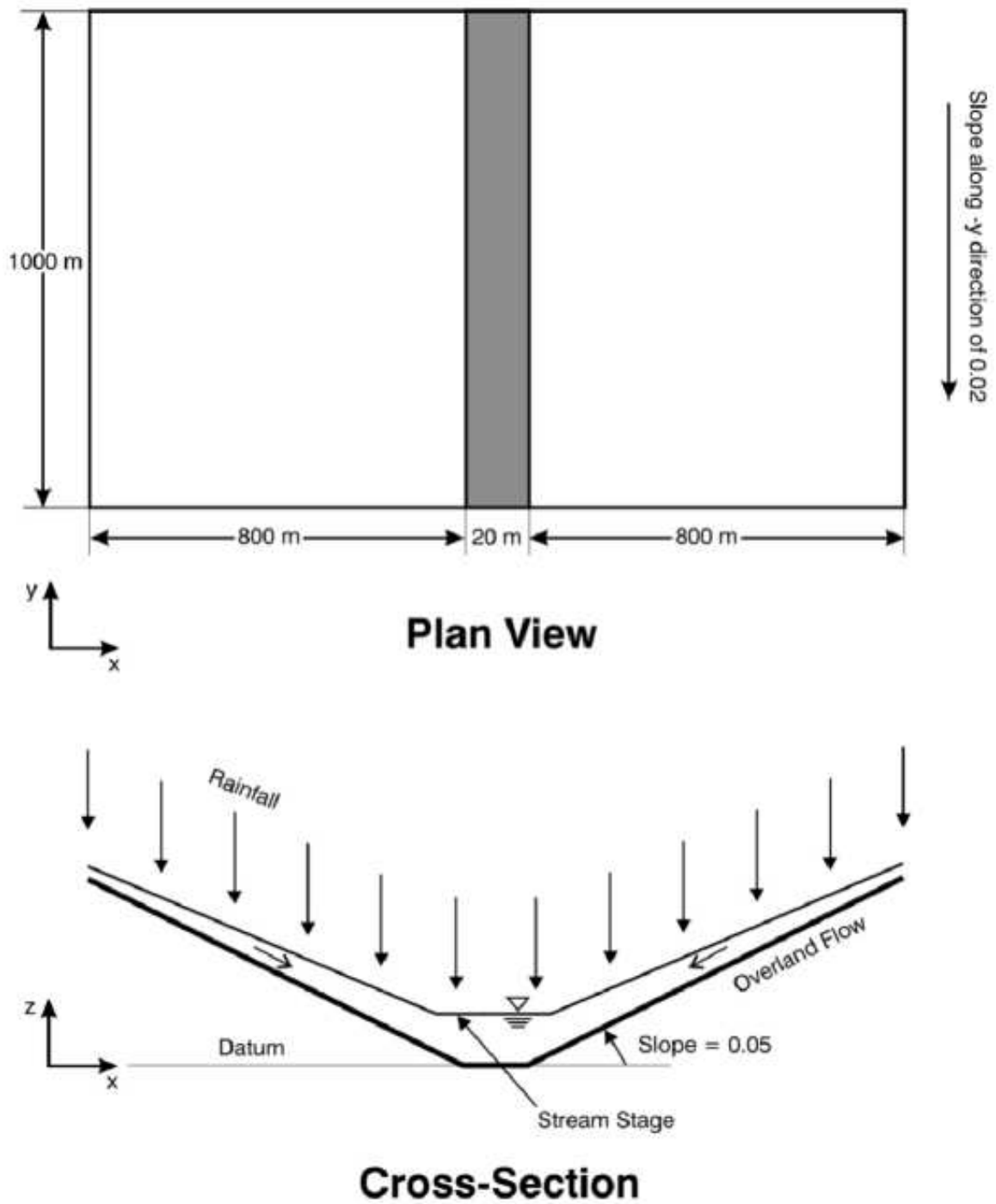
Figure 4.20: Schematic Description of 2-D Surface Water Flow Study of *diGiammarco et al.* [1996].
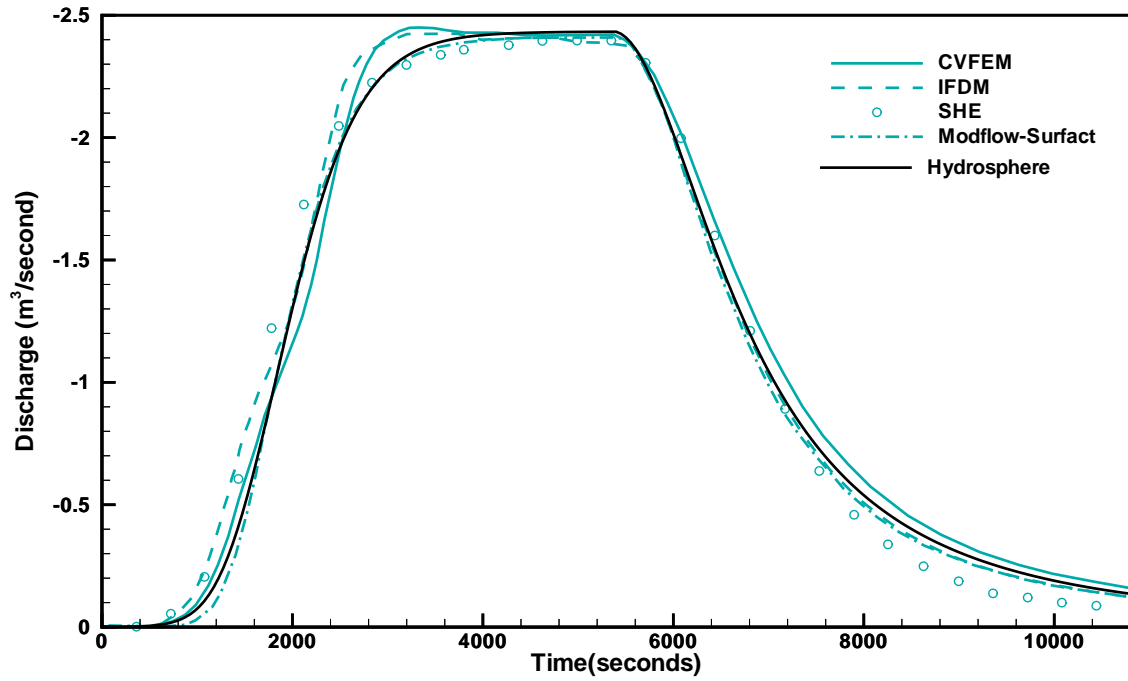
Figure 4.21: Outflow Hydrograph for Simulation of 2-D Surface Water Flow Study of *di-Giammarco et al.* [1996].
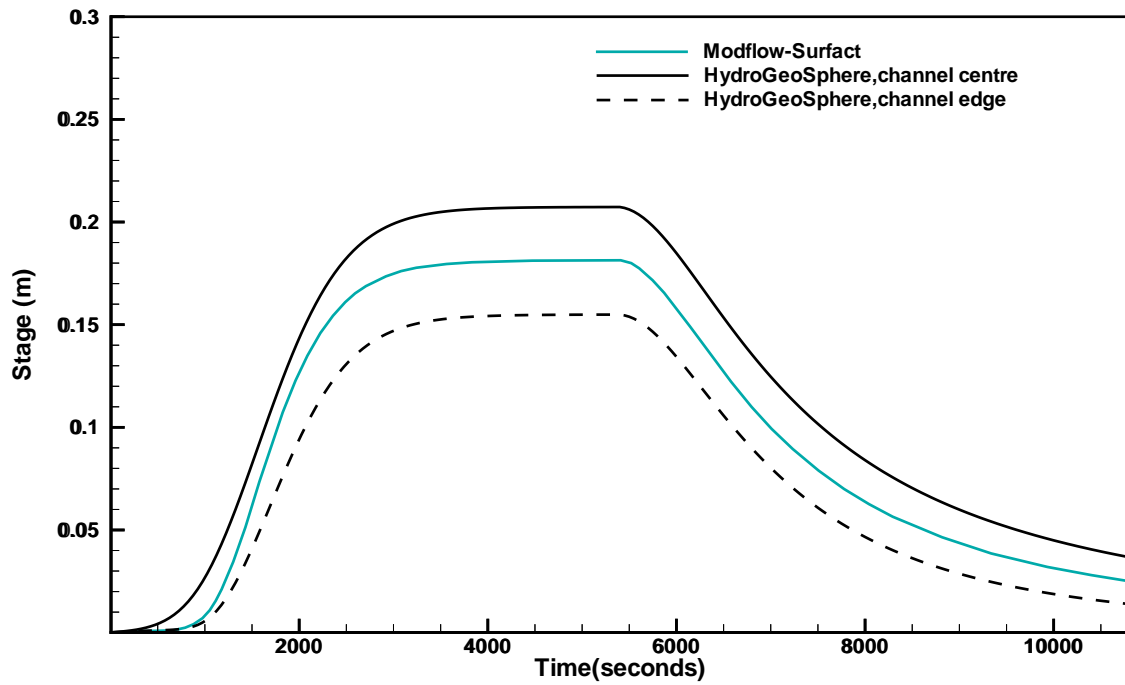
Figure 4.22: Channel Stage at Outlet for Simulation of 2-D Surface Water Flow Study of *diGiammarco et al.* [1996].
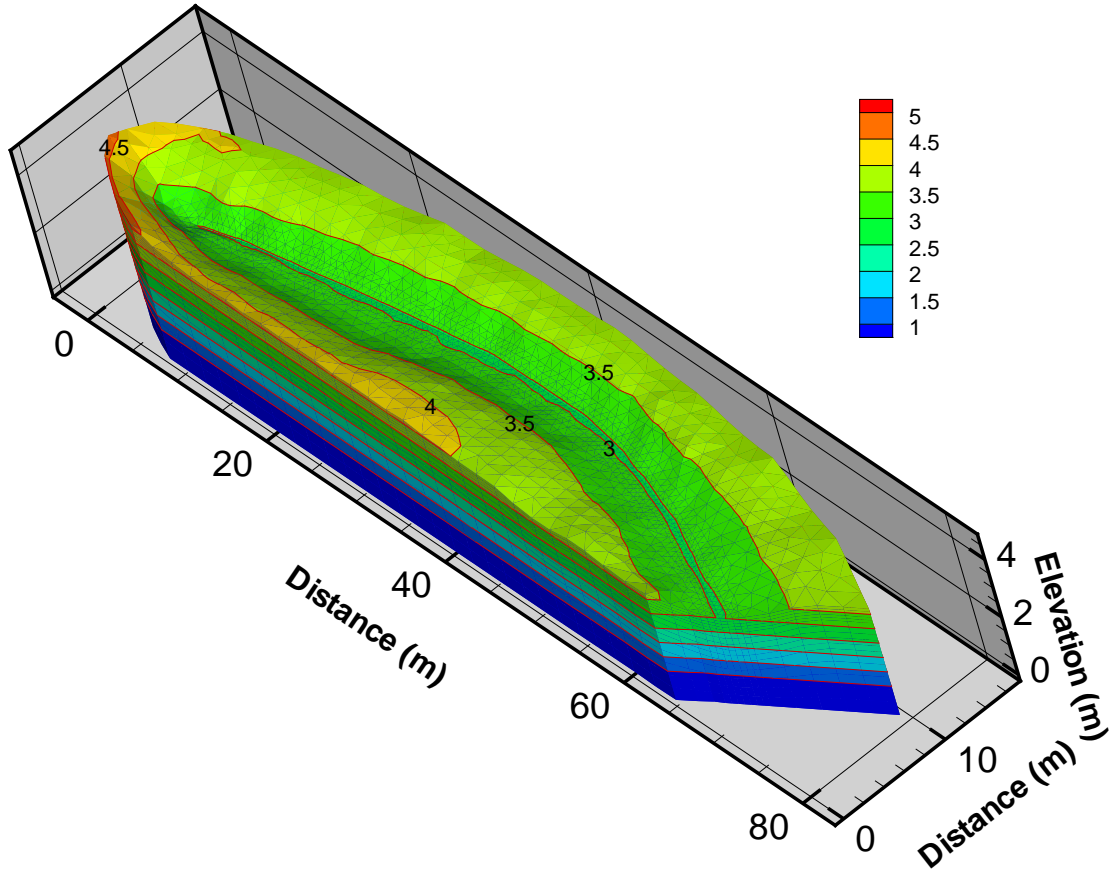
Figure 4.23: Site Description for Rainfall-Runoff Field Experiment of *Abdul* [1985] [from *VanderKwaak*, 1999].

4.3 Coupled Surface/Subsurface Flow

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

A field scale simulation is performed with **HydroGeoSphere** to verify its capabilities for fully three-dimensional surface/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 4.23 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 hour^{-1} 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 4.10.

Table 4.10: Parameter Values for Simulation of the 3-D Field Scale Study of *Abdul* [1985].

Parameter	Value	Unit
porosity, Θ	0.37	
hydraulic conductivity, K	1×10^{-5}	m s^{-1}
storage coefficient, S_s,	1.2×10^{-7}	m^{-1}
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	
Manning coefficient for plot	0.3	s m$^{-1/3}$
Manning coefficient for channel	0.03	s m$^{-1/3}$
Initial water table elevation	2.78	m

Modeling Approach and Results

The domain was discretized areally into 1372 nodes and 2651 triangular elements. Vertically, the grid is distorted to conform with the topographic elevation as shown in Figure 4.24. Fifteen layers of elements were used with a fine discretization of 0.1 m near the surface which was enlarged to 1 m near the bottom. The dual node approach was used to simulate surface flow. The resulting 3-D finite element mesh is identical to the one used by *VanderKwaak* [1999]. Recharge is provided at the rate of 5.56×10^{-6} m s^{-1} for a first stress period of 50 minutes, with zero recharge for the second 50 minute stress period. A critical depth boundary condition is applied all around the upper surface of the domain, and the simulation is performed for 100 minutes, with no recharge occurring for the second 50 minute stress period. Adaptive time-stepping is used with an initial time-step size of 5s, a maximum time-step size of 100s, and time-step incrementing and decrementing factor limits of 2.0 and 0.5, respectively. Iteration parameters include 100 and 15 inner and outer iterations respectively, and a convergence tolerance of 1×10^{-4} m. Figure 4.25 shows the outflow hydrograph to be close to the experimental values, and to the simulation of *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.

The spatial distribution of water depth within the domain at t=50 mins is shown in Figure 4.26 for both *VanderKwaak* [1999] and **HydroGeoSphere**. These compare well for regions of maximum water depth around the channel but differ elsewhere.

These small differences between the outflow hydrograph and surface water depth between the two simulations may be due to the different approaches used to define the flow coupling term between the surface and subsurface meshes in the dual node approach. *VanderKwaak* [1999] used a flow coupling length (in this case 1×10^{-4} m), height of microtopography (1×10^{-2} m) and mobile water depth (1×10^{-4} m) to define the coupling term while **Hydro-GeoSphere** uses a coupling length of 1×10^{-1} m and a rill storage height of 2×10^{-3} m. Also, *VanderKwaak* [1999] assigned critical depth boundary conditions to a few nodes at the ouflow end of the channel, while in **HydroGeoSphere**, the critical depth boundary was assigned all around the outside of the domain, which would account for the depressions
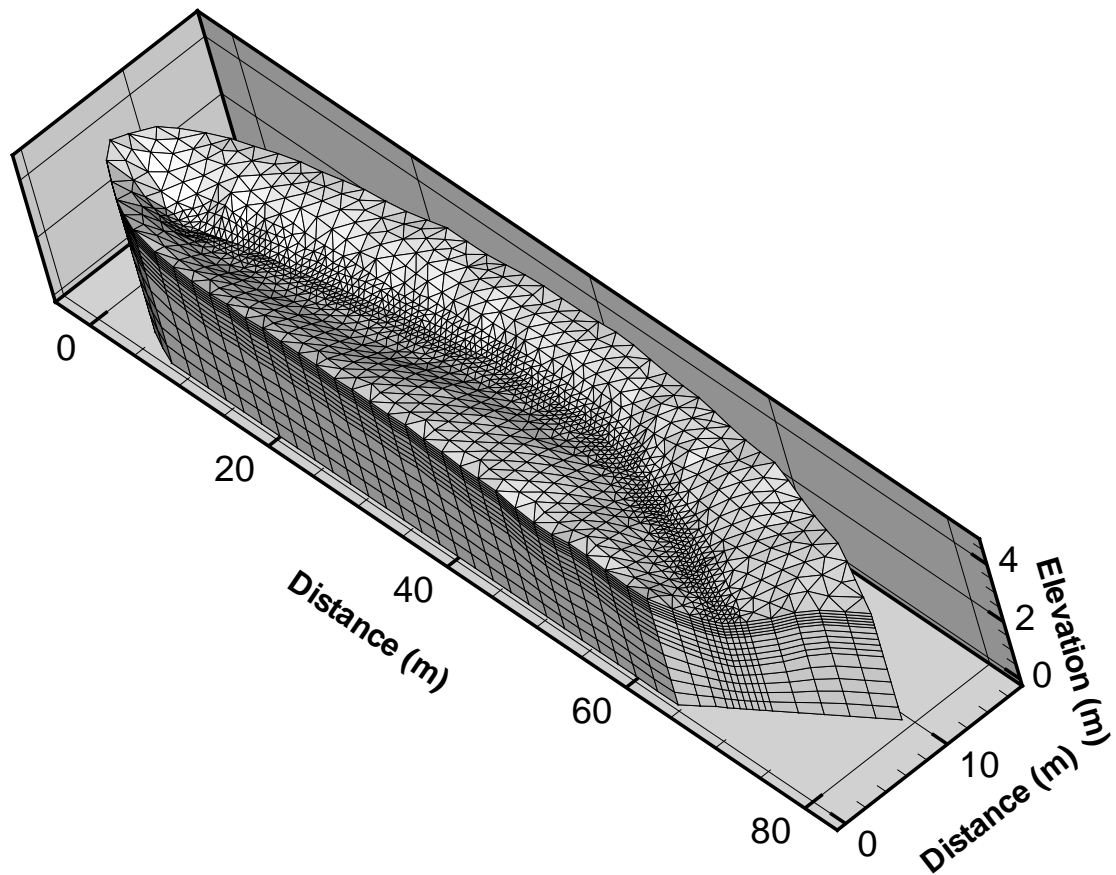
Figure 4.24: Three-dimensional View of Topography and Finite element Grid, for Simulation of the *Abdul* [1985] Rainfall-Runoff Field Experiment.
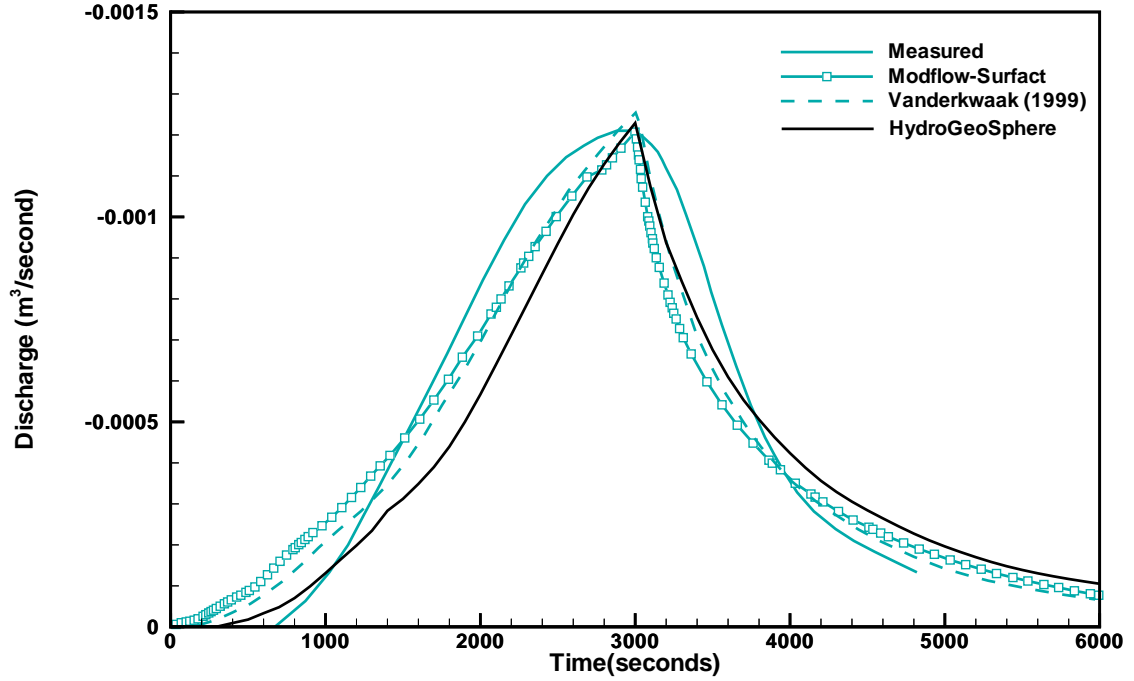
Figure 4.25: Outflow Hydrograph for Simulation of the *Abdul* [1985] study.

in the water depth surface observed for **HydroGeoSphere** in Figure 4.26.

4.3.2   Level 2: 3-D Surface/Subsurface Flow and Evapotranspiration

Evapotranspiration is verified using the example of *Panday and Huyakorn*[2004], who extended the 2-D V-catchment surface flow problem described in Section 4.2.3 to include the subsurface domain, which extends to a depth of 20 m below the channel outlet. The domain is discretized into 11 subsurface layers with areal gridding mirroring the grid of the overland flow domain. The top ten subsurface layers each have a thickness of 1 m and the bottom of the last layer is at z=-20 m. Again, only one half of the domain need be simulated due to symmetry, with published outflow discharge halved, to produce equivalent results. The subsurface domain has horizontal and vertical hydraulic conductivity values of 5×10^{-5} and 5×10^{-6} m s^{-1} respectively, a porosity value of 0.1, and Van Genuchten parameters α, β and S_{wr} equal to 2.25 m^{-1}, 1.89 and 0.16 respectively.

The same rainfall intensity of 3×10^{-6} m s^{-1} as for the previous case is applied for a period of 35 days, followed by no rainfall for the second stress period of 15 days.

In order to gauge the performance of the evapotranspiration formulation, evapotranspiration losses, as parameterized in Table 4.11 were applied. Because Modflow-Surfact uses a block-centred approach while **HydroGeoSphere** uses a node-centred approach, the evaporation and root zone distribution functions are similar, but not identical.
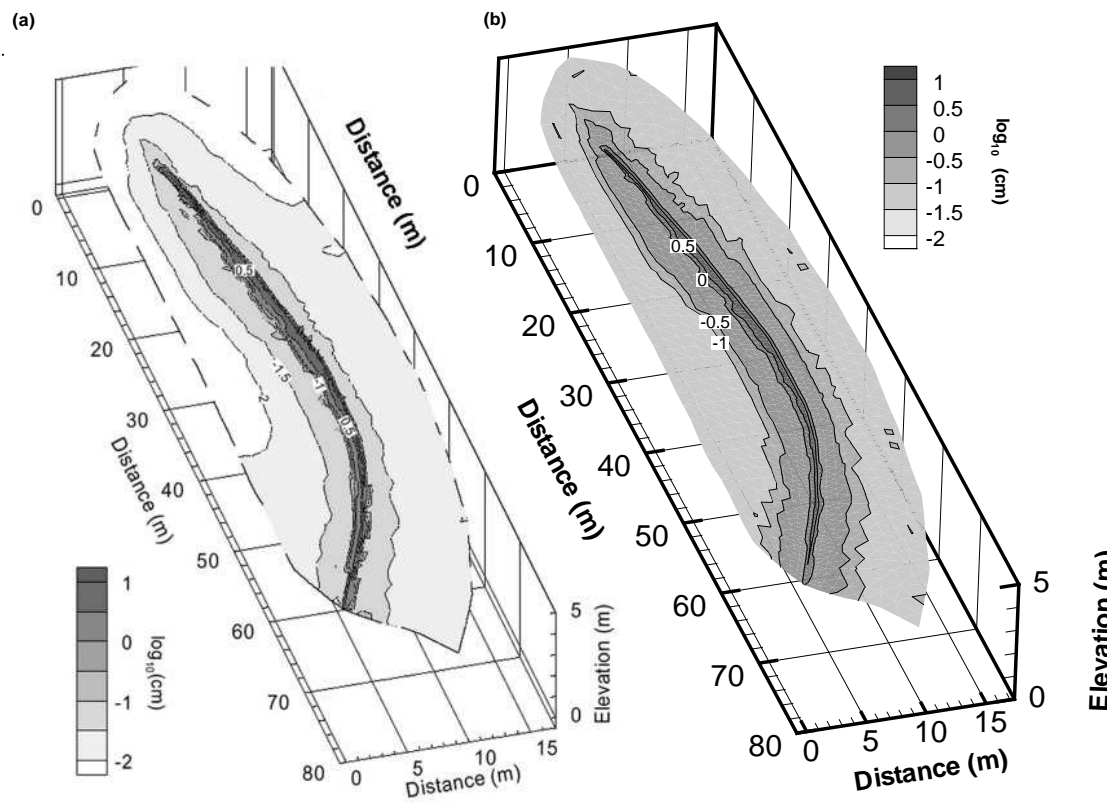
Figure 4.26: Spatial Distribution of Water Depth after 50 Minutes of Field Experiment for (a) *VanderKwaak* [1999] and (b) **HydroGeoSphere**.

Table 4.11: Parameter Values for the Simulation of the *Panday and Huyakorn*[2004] Study.

Parameter	Value	Unit
Reference evapotranspiration	3×10^{-7}	m
Leaf area index	2.08	
Saturation at field capacity	0.32	
Saturation at wilting point	0.2	
Saturation at oxic limit	0.76	
Saturation at anoxic limit	0.9	
Energy limiting stage saturation	0.32	
Evaporation limiting stage saturation	0.2	
Transpiration constants		
C_1	0.3	
C_2	0.2	
C_3	3×10^{-6}	m s^{-1}
Soil evaporation distribution function		
Depth = 0.0	0.703	
Depth = 1.0	0.259	
Depth = 2.0	0.037	
Depth = 3.0	0.0	
Root zone distribution function		
Depth = 0.0	0.703	
Depth = 1.0	0.259	
Depth = 2.0	0.037	
Depth = 3.0	0.0	
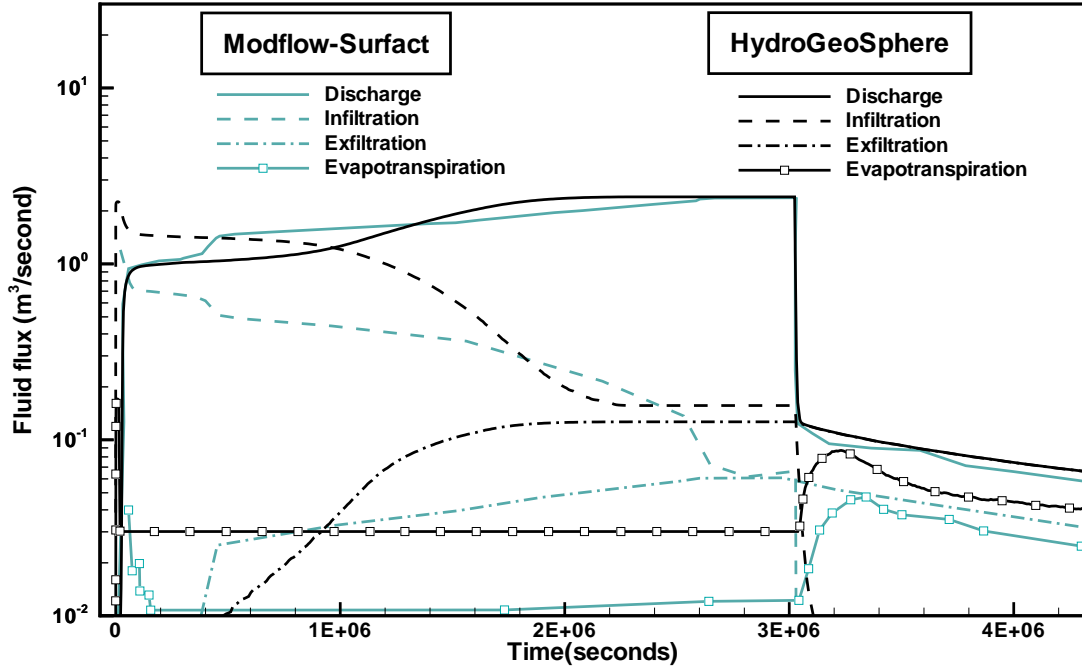
Figure 4.27: Water Budget Components for the Simulation of the *Panday and Huyakorn.*[2004] Study.

Adaptive time stepping with an initial time-step size of 5s, a maximum time-step size of 10000s and a time-step incrementing factor of 2.0 is used for the simulation. Newton iteration considerations include a maximum of 15 iterations and an absolute convergence tolerance of 10^{-3}.

Figure 4.27 compares predicted fluid fluxes from **HydroGeoSphere** with predictions from Modflow-Surfact. The responses of the two models are somewhat similar. The disparity is attributed to differences in the formulation of the surface/subsurface flux coupling term between the block-centred and node-centred approaches.

4.4 Subsurface Transport

4.4.1 Level 1: Chain Decay Transport in a Porous Medium

In order to verify the accuracy of **HydroGeoSphere** 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:

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

Table 4.12: Parameter Values for Chain-decay Transport in a Porous Medium.

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

The input parameters and values for the analytical solution are shown in Table 4.12:

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^{-1} parallel to the x-axis was produced. A constant concentration of 1.0 was specified for Uranium^{234} at the upstream x-face.

Figure 4.28 shows concentration profiles for the three members of the decay chain at a time of 10000 years for both CMM and **HydroGeoSphere**. It can be seen that the results are almost identical.

4.4.2   Level 1: Chain Decay Transport in a Single Fracture

In order to verify the accuracy of **HydroGeoSphere** 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 4.13.

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

Table 4.13: Parameter Values for Chain-decay Transport in a Fracture.

Parameter	Value	Unit
Velocity in fracture	100.0	m yr^{-1}
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^{-1}
Inlet dispersion	0.0	m yr^{-1}
Diffusion coefficient [‡]		
Uranium^{234}	3.1536×10^{-2}	m^2 yr^{-1}
Thorium^{230}	3.1536×10^{-2}	m^2 yr^{-1}
Radium^{226}	3.1536×10^{-2}	m^2 yr^{-1}
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
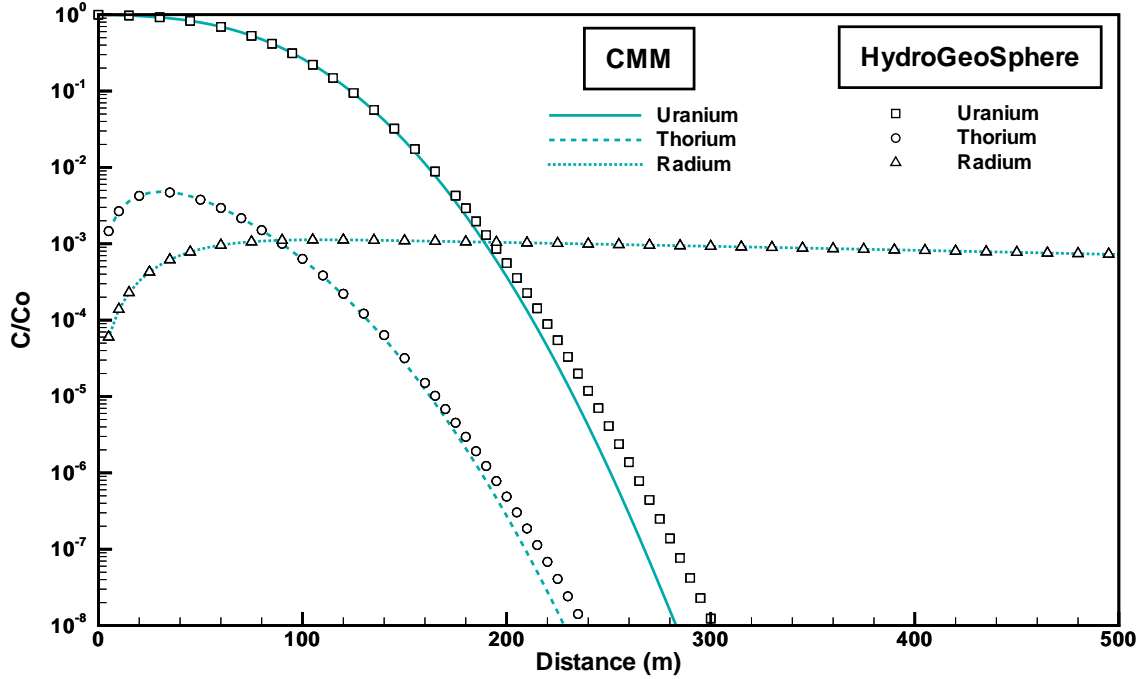
Figure 4.28: Results for a 3-member Decay Chain in a Porous Medium at 10000 Years.

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^{-1} 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 4.29 shows the concentration profiles of Uranium, Thorium and Radium at 10000 years for both DKCRACK and **HydroGeoSphere** simualtions. The results are nearly identical.

4.4.3   Level 1: Time-variable Source Condition

In order to verify the accuracy of **HydroGeoSphere** 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 4.30. Radioactive decay is neglected in the simulation. Other input parameters for the analytical solution are shown in Table 4.14.

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.
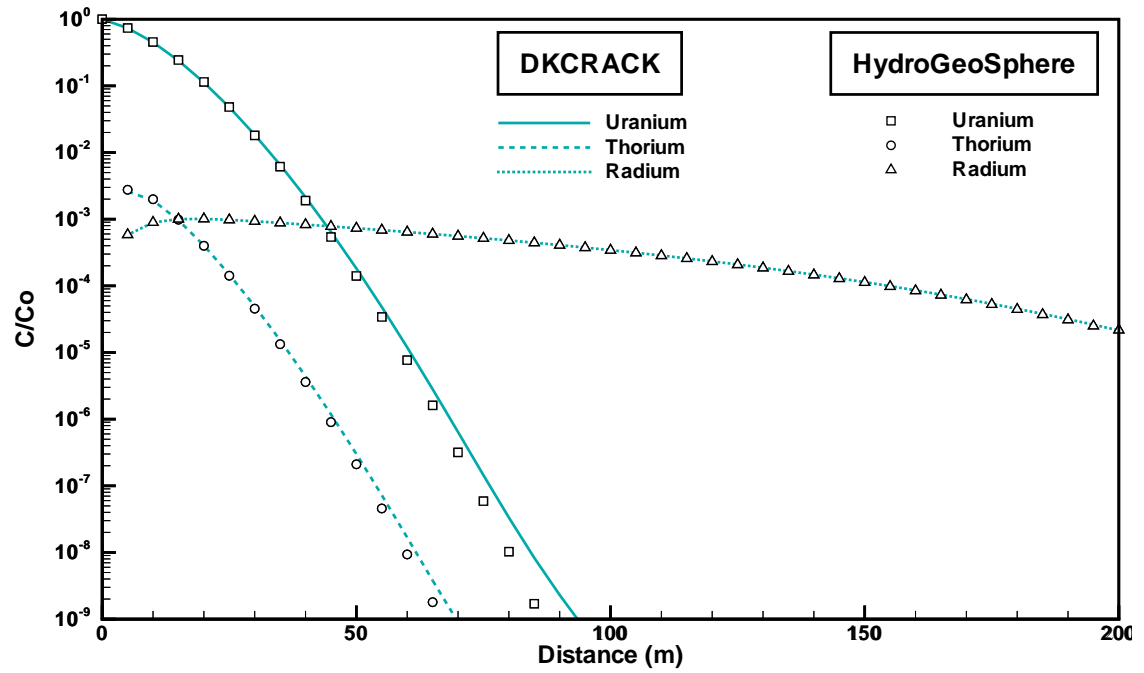
Figure 4.29: Results for a 3-member Decay Chain in a Fractured Medium at 10000 Years.
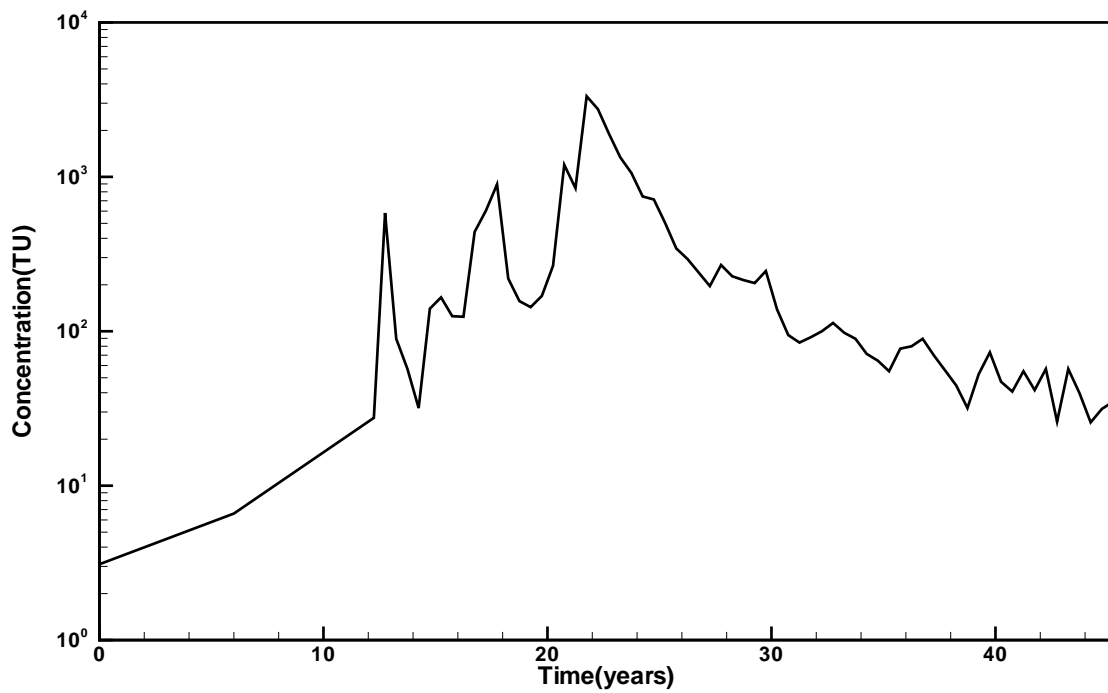
Figure 4.30: Input Function for Time-variable Source Transport.

Table 4.14: Parameter Values for Time-varying Source Transport Simulation.

Parameter	Value	Unit
Velocity	1.0	m yr^{-1}
Dispersivity	1.0	m
Diffusion coefficient	0.0	m^2 yr^{-1}
Retardation factor	1.0	
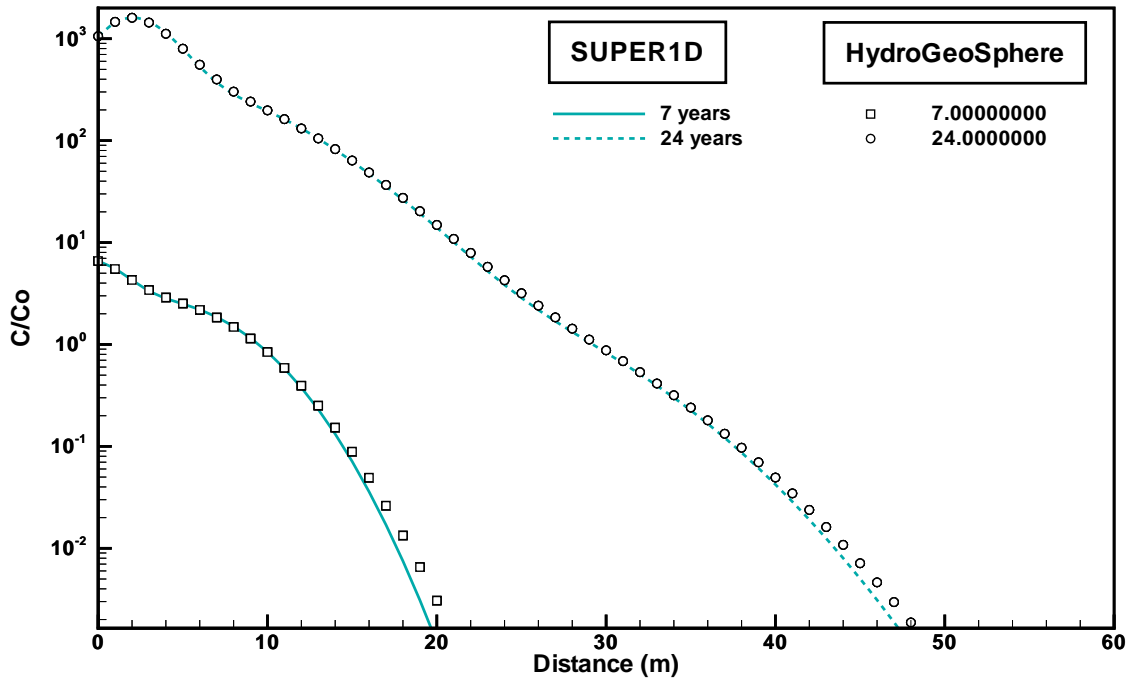
Figure 4.31: Results for a Time-variable Source Function.

Medium properties and flow boundary conditions were set up such that a uniform average linear subsurface water flow velocity of 1.0 m yr^{-1} 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 4.31. Results from **HydroGeoSphere** 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.

Table 4.15: Parameter Values for Dual-porosity Transport Simulation.

Parameter	Value	Unit
Darcy velocity	2.5	cm day^{-1}
Total porosity	0.5	
Mobile fraction	0.5	
Dispersion coefficient	10.0	cm^2 day^{-1}
Retardation factor	1.0	
Mass transfer coefficient	0.1	day^{-1}

4.4.4 Level 1: Transport in a Dual-Porosity Medium

In order to verify the accuracy of **HydroGeoSphere** 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 4.15.

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 day^{-1} 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 day^{-1} when multiplied by the average linear subsurface water velocity.

Figure 4.32 shows a concentration profile at a time of 2.5 days for both MPNE and **HydroGeoSphere**. It can be seen that the results are almost identical.

4.4.5 Level 2: Coupled Flow and Transport in a Dual-Permeability Medium

The capability of **HydroGeoSphere** to simulate transient flow and solute transport in a coupled porous medium-dual continuum system was tested by comparing it to results obtained by *Gerke and Van Genuchten* [1993]. The problem consists of a one-dimensional column 40 cm in length, in which the macropores are assumed to be planes with a spacing of 2 cm that subdivide the porous medium into uniform blocks. Similar to *Gerke and Van Genuchten* [1993], the domain was finely discretized in the vertical direction, using elements which were 0.1 m in length. The initial pressure head in the domain was set to -1000 cm. A uniform rainfall of 50 cm day^{-1} was applied to the dual continuum, and water was allowed to drain freely from the base of the system in both the porous medium and dual-continuum. The initial solute concentration in the domain was set to 1.0. The hydraulic and transport parameters of the system are presented in Table 4.16. The fluid exchange term α_{wd} is 0.6 cm^{-1} day^{-1}, which is half that used by *Gerke and Van Genuchten* [1993]. This is a way of reconciling the results of the two codes, as *Gerke and Van Genuchten* [1993] use
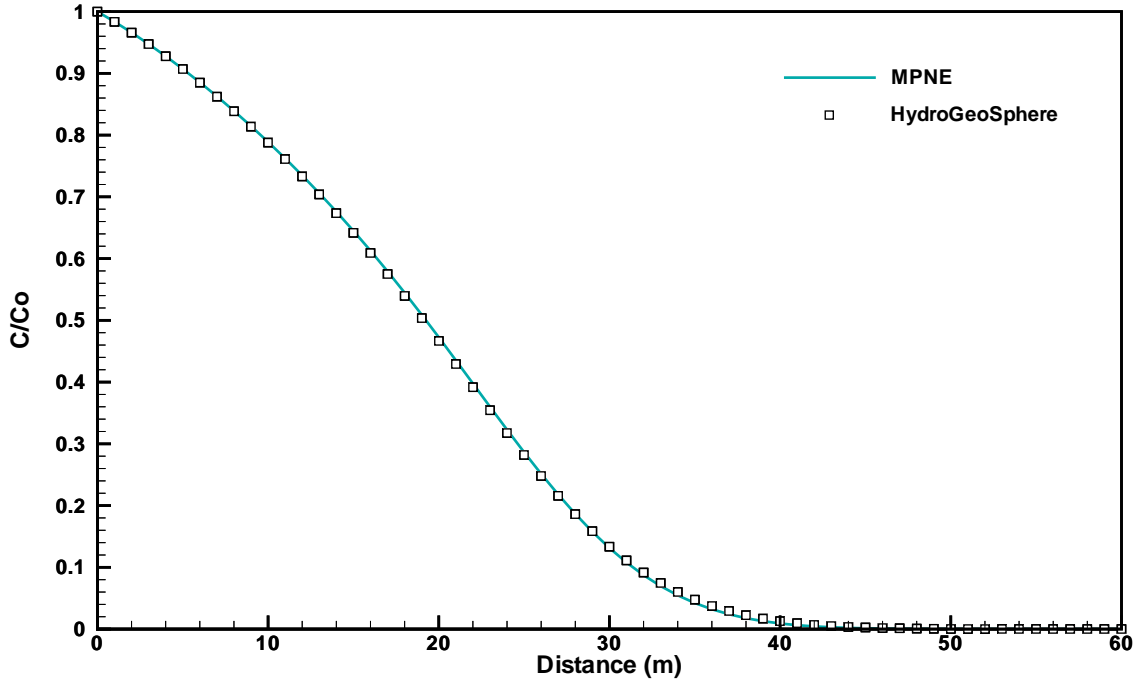
Figure 4.32: Results for Transport in a Dual-porosity Medium.

central weighting of relative permeability for computing the exchange fluxes between the two continuua, while **HydroGeoSphere** uses upstream weighting.

Figure 4.33 shows pressure head profiles versus depth at 0.01, 0.04 and 0.08 days for both the porous medium and dual continuum.

Figure 4.34 shows concentration profiles versus depth at 0.01, 0.04 and 0.08 days for both the porous medium and dual continuum.

4.4.6   Level 2: 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 4.35). 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 4.17. 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,

Table 4.16: Parameters for the *Gerke and Van Genuchten* [1993] study.

Parameter	Value	Unit
Porous medium		
Saturated hydraulic conductivity, $\mathbf{K}$	1.0526	cm day^{-1}
Porosity, θ_s	0.5	
Specific storage, S_s	1×10^{-7}	cm^{-1}
Volume fraction, w_m	0.95	
Van Genuchten α	0.005	cm^{-1}
Van Genuchten β	1.5	
Residual water saturation S_{wr}	0.10526	
Dispersivity α_l, α_t	2.0	cm
Dual continuum		
Saturated hydraulic conductivity, $\mathbf{K_d}$	2000	cm day^{-1}
Porosity, θ_{sd}	0.5	
Specific storage, S_{sd}	1×10^{-7}	cm^{-1}
Volume fraction, w_d	0.05	
Van Genuchten α	0.01	cm^{-1}
Van Genuchten β	2.0	
Residual water saturation S_{wrd}	0.0	
Dispersivity α_{ld}, α_{td}	2.0	cm
Interface		
Saturated hydraulic conductivity, K_a	0.01	cm day^{-1}
Geometrical shape factor β_d	3	
Van Genuchten α	0.005	cm^{-1}
Van Genuchten β	1.5	
Residual water saturation S_{wr}	0.10526	
Fluid exchange term α_{wd}	0.6	cm^{-1} day^{-1}
Mass exchange term α_s	0.15	day^{-1}
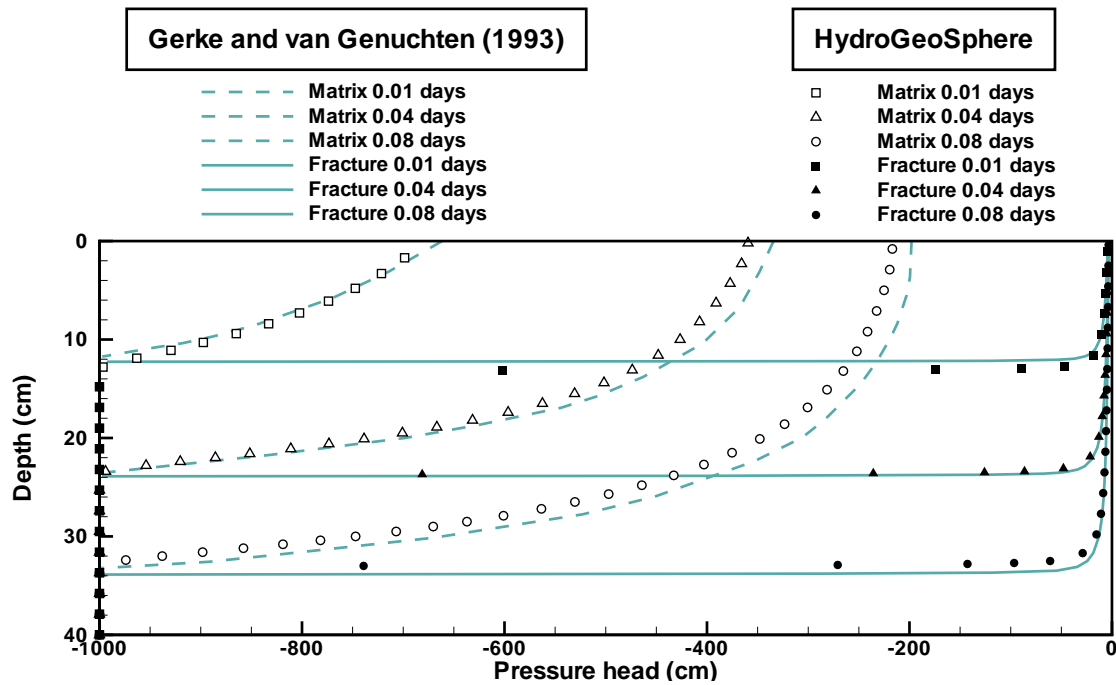
Figure 4.33: Pressure head profiles of the *Gerke and Van Genuchten* [1993] study.
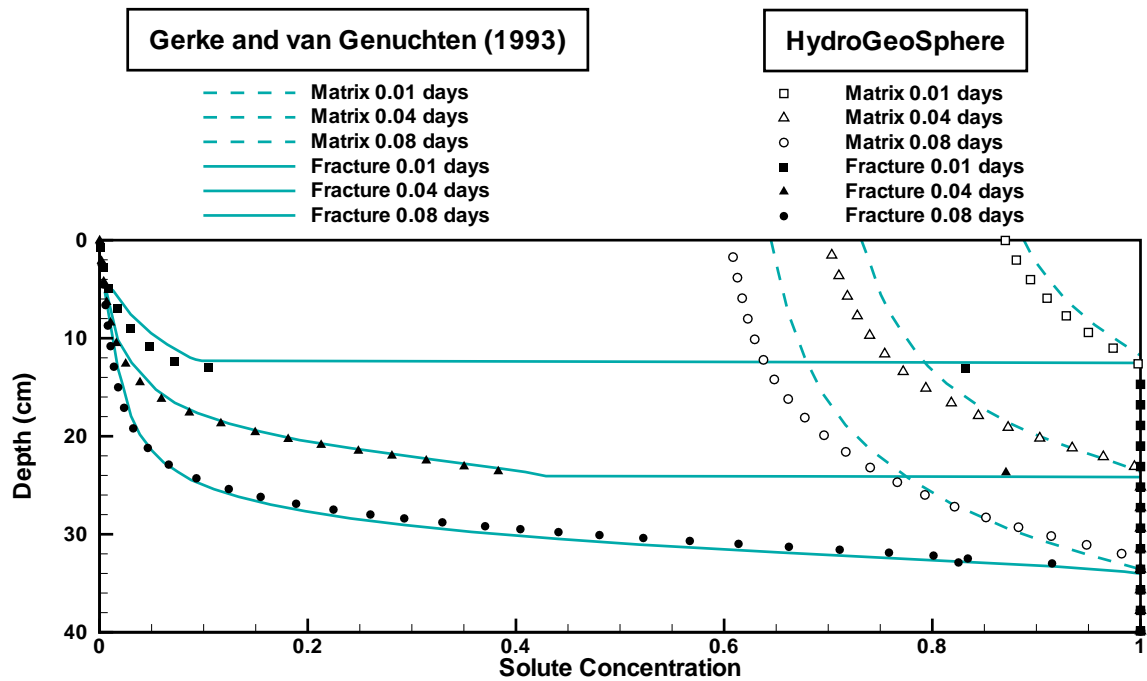
Figure 4.34: Concentration profiles of the *Gerke and Van Genuchten* [1993] study.
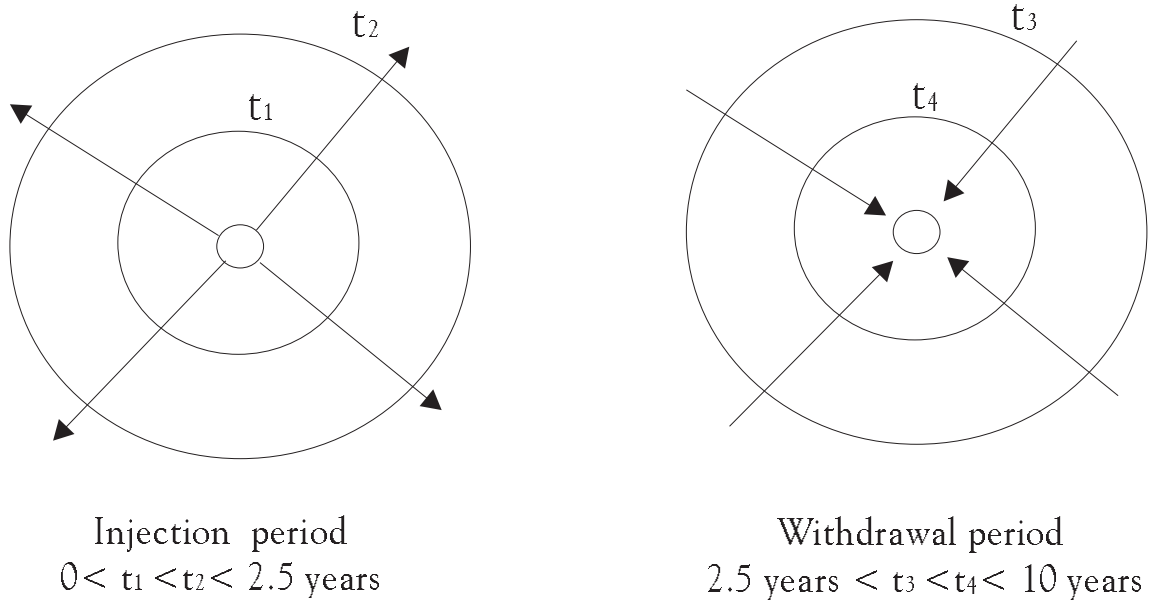
Figure 4.35: Injection/withdrawal Well System.

into 28 time steps. The simulation with **HydroGeoSphere** is compared with the analytical solution in Figure 4.36. The breakthrough curve obtained from **HydroGeoSphere** is comparable to the analytical solution.

4.4.7 Level 1: 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 4.37. 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 **HydroGeoSphere**. The model parameters, shown in Table 4.18, 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 **HydroGeoSphere** for a non-conservative species.

Both simulations were performed using a rectangular domain containing 741 30 m × 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 4.38. Numerical results from **HydroGeoSphere** 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 **HydroGeoSphere** produces solutions which are of comparable accuracy with

Table 4.17: Model Parameters for the Simulation of Transport From an Injection/extraction Well.

Parameter	Value	Unit
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^{-1}
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^{-1}
Concentration during the first stress period	1	
Length of the injection period	2.5	yr
Length of the extraction period	7.5	yr
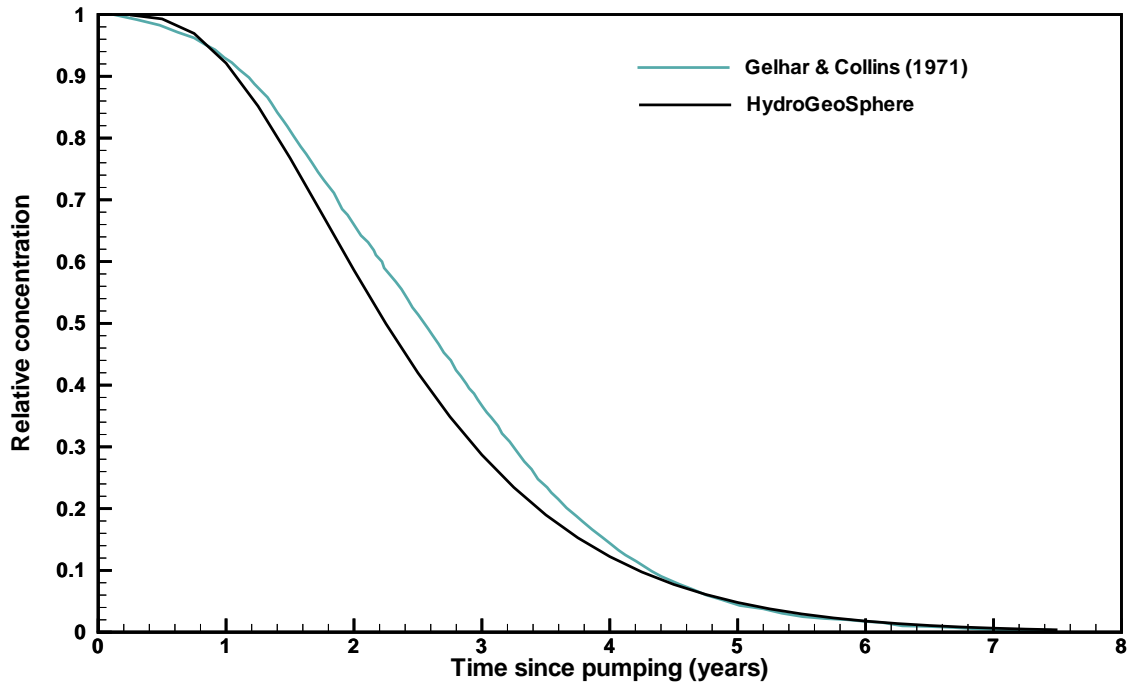
Figure 4.36:   Breakthrough Curve from Simulation of Transport Due to an Injection/withdrawal Well.
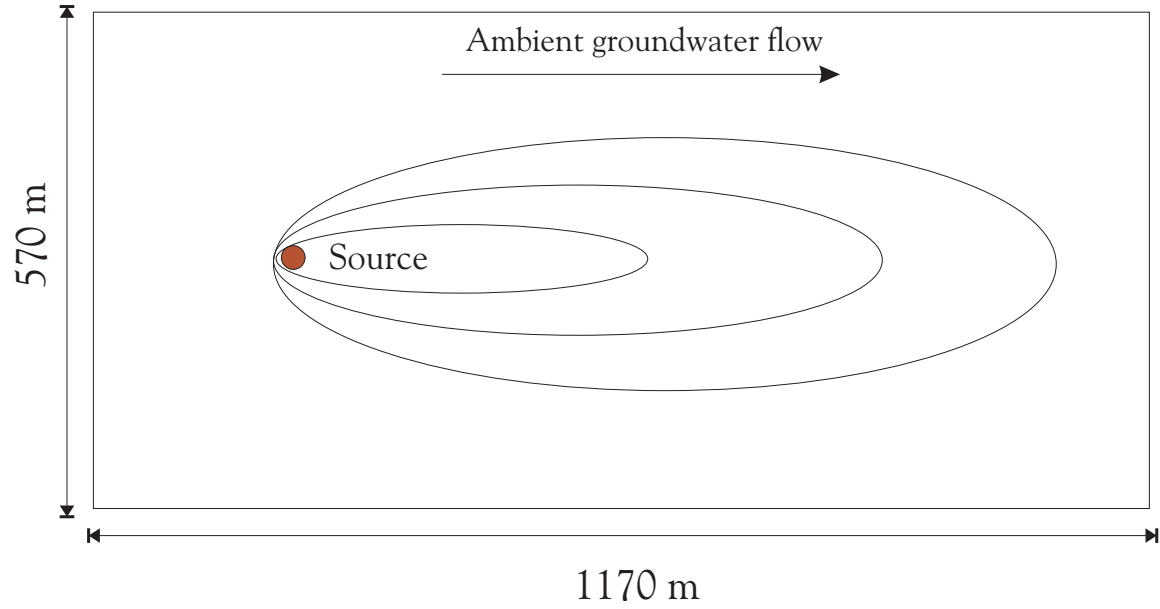
Figure 4.37: Two-dimensional Transport from a Point Source.

Table 4.18: Parameters for Simulation of 2-D Transport from a Point Source

Parameter	Value	Unit
Darcy velocity q	0.161	m day^{-1}
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^{-1} day^{-1}
CASE 1:		
Linear adsorption coefficient, K_d	0	m^3 kg^{-1}
Retardation coefficient, $R = \rho_b K_d / \varphi$	1	
Decay constant, λ	0	day^{-1}
CASE 2:		
Linear adsorption coefficient, K_d	0.14	m^3 kg^{-1}
Retardation coefficient $R = \rho_b K_d / \varphi$	2	
Decay constant λ	0.00019	day^{-1}
Bulk density ρ_b	2.5	kg m^{-3}
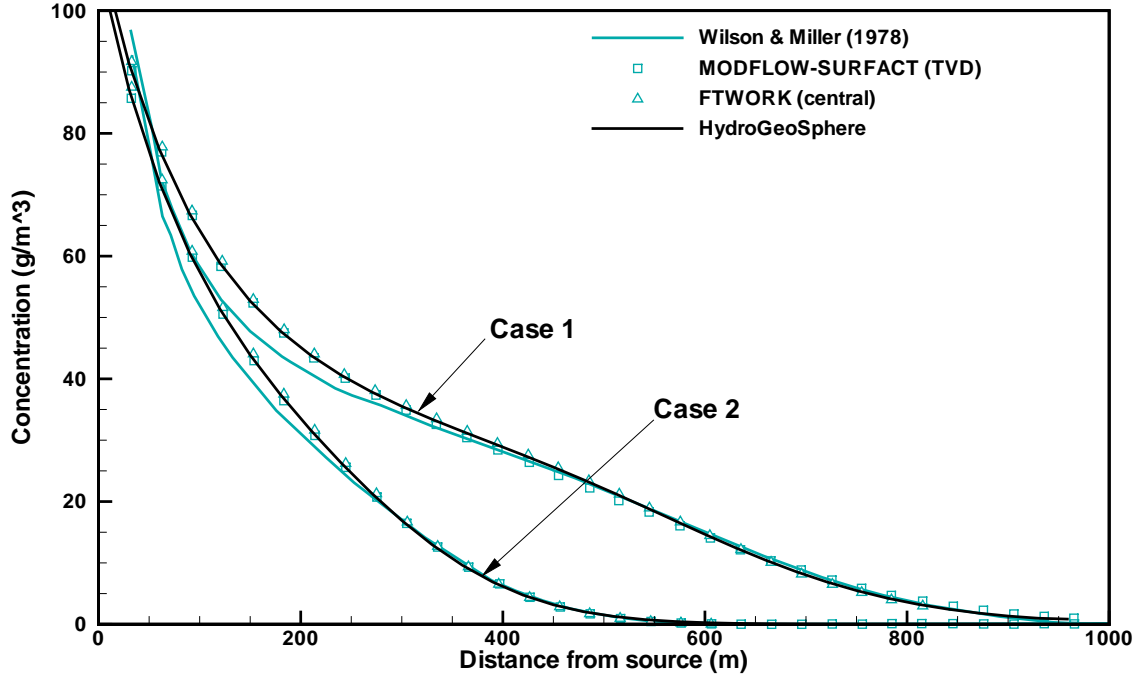
Figure 4.38: Concentration Profiles Along the Center Line for t =1400 Days.

respect to the other models.

4.4.8   Level 1: Transport Due to an Injection-Withdrawal Well Pair

This problem concerns solute transport between a pair of discharging and recharging wells operating at a constant flow rate. Both wells fully penetrate a constant-thickness confined aquifer that is assumed to be homogeneous, isotropic and of infinite areal extent. The flow field is assumed to be in steady-state with typical flow lines and solute fronts depicted in Figure 4.39.

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 4.19. The hydraulic conductivity used is 0.005 cm s^{-1}.

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 4.40 presents the results obtained from **HydroGeoSphere** 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.
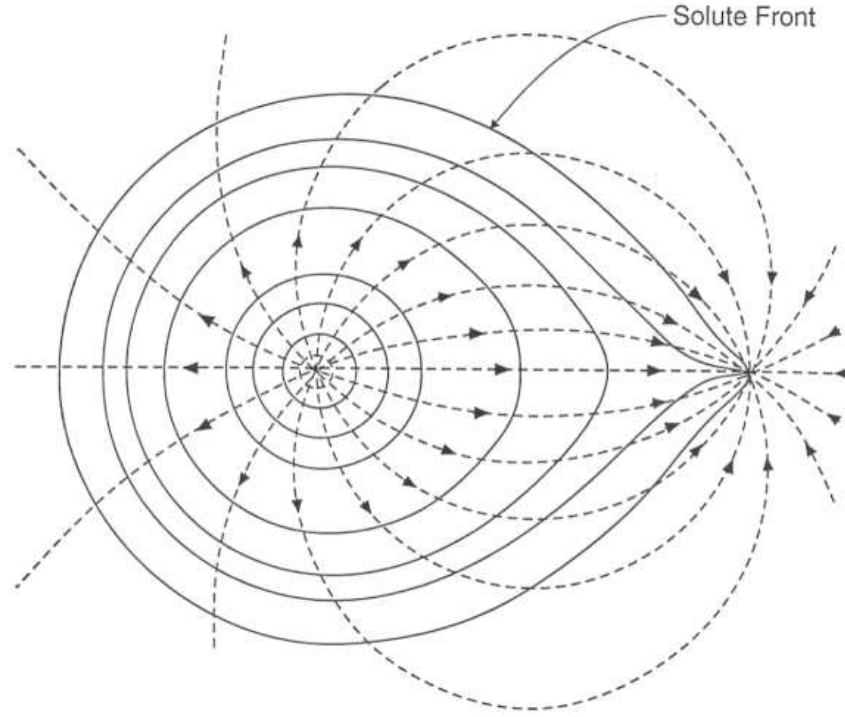
Figure 4.39: Injection-withdrawal Well Pair.

Table 4.19: Parameters for Simulations of Transport Due to an Injection-withdrawal Pair.

Parameter	Value	Unit
Well flow rate, $Q_{inj}=Q_{pump}$	2.339	$cm^3\ s^{-1}$
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	
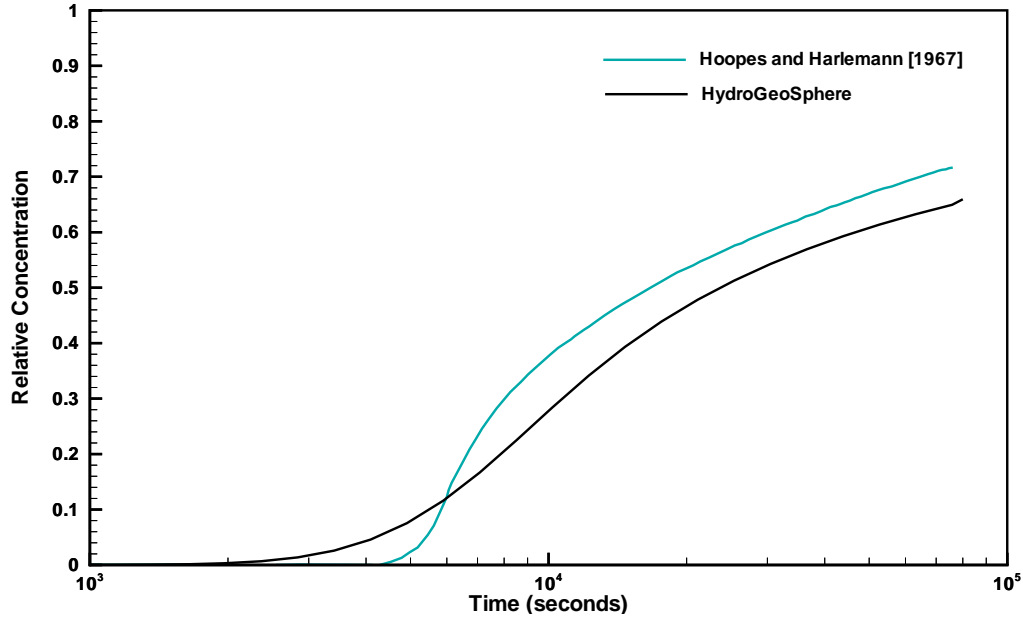
Figure 4.40: Breakthrough Curve of Concentration Solute at the Pumping Well.

4.4.9   Level 2:   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 4.41. 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 × 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 **HydroGeoSphere** simulations contains 18 columns, 14 rows and one layer of cells of uniform dimensions of 100 m × 100 m. The simulation was done with 100 equal time steps for a total time of 6 years. Results show a good correlation between the **HydroGeoSphere** simulation and MT3D results (see Figure 4.42) with negligible mass balance error as shown in Figure 4.43.

No mass flux

H=53.75 m

Constant Head: h=250 m

$Q_i = 0.001$ m^3/s
$C_i = 57.87$ ppm
$t_i = 1$ year

$Q_p = 0.0189$ m^3/s
$t_p = 6$ years

No mass flux

Injection well

Pumping well

Low conductivity zone
K= 1.474 E-7 m/s

600 m

Gradient 2.5/100

1400 m

Confined aquifer: thickness =10 m
K= 1.474 E-4 m/s

N

200 m

h=18.75 m

No mass flux

1800 m
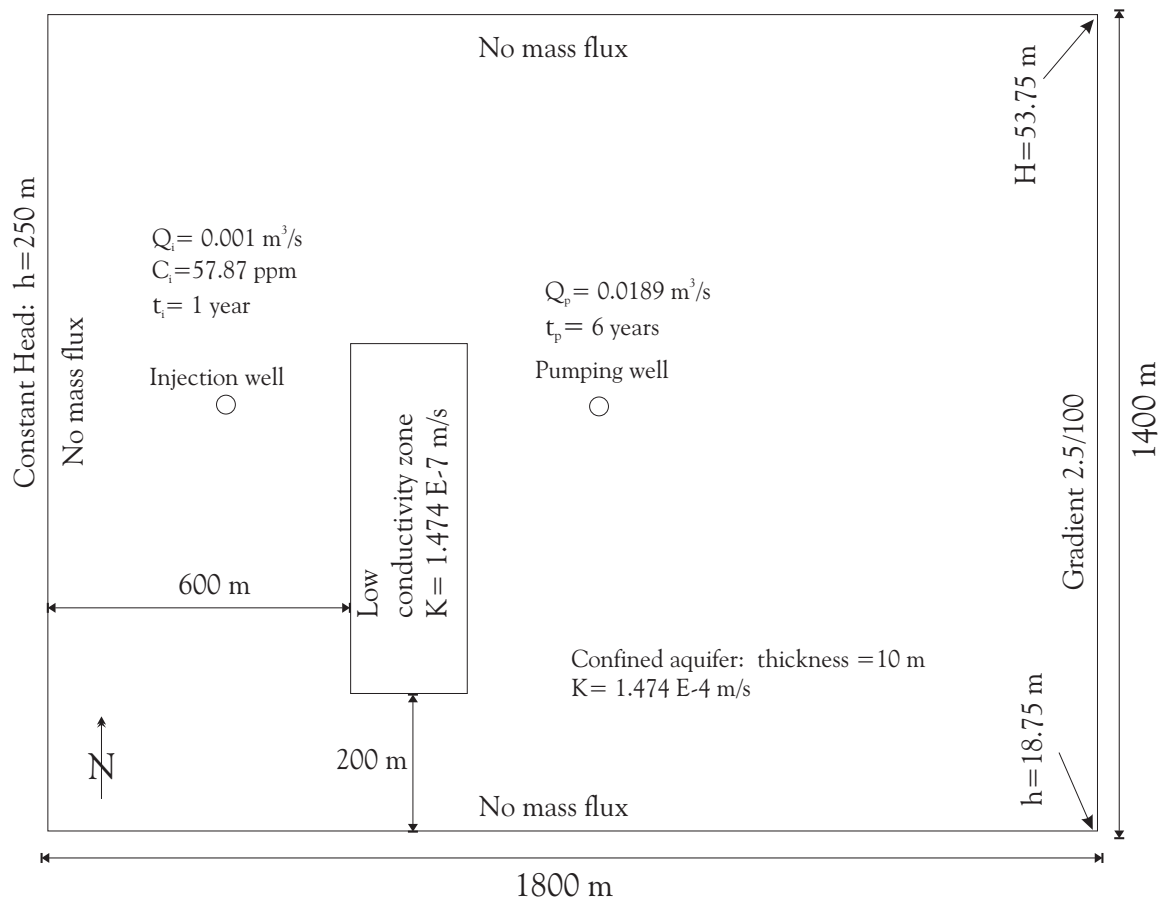
Figure 4.41: Problem Description for 2-D Transport in a Heterogeneous Confined Aquifer.
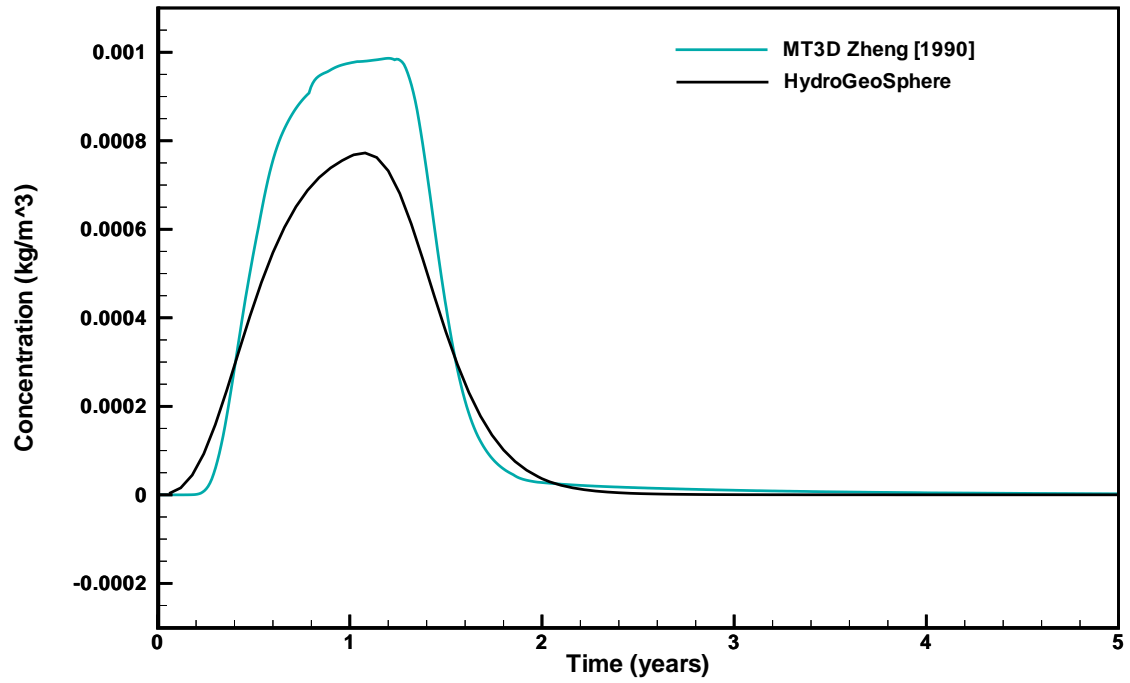
Figure 4.42: Concentrations Observed at the Pumping Well.
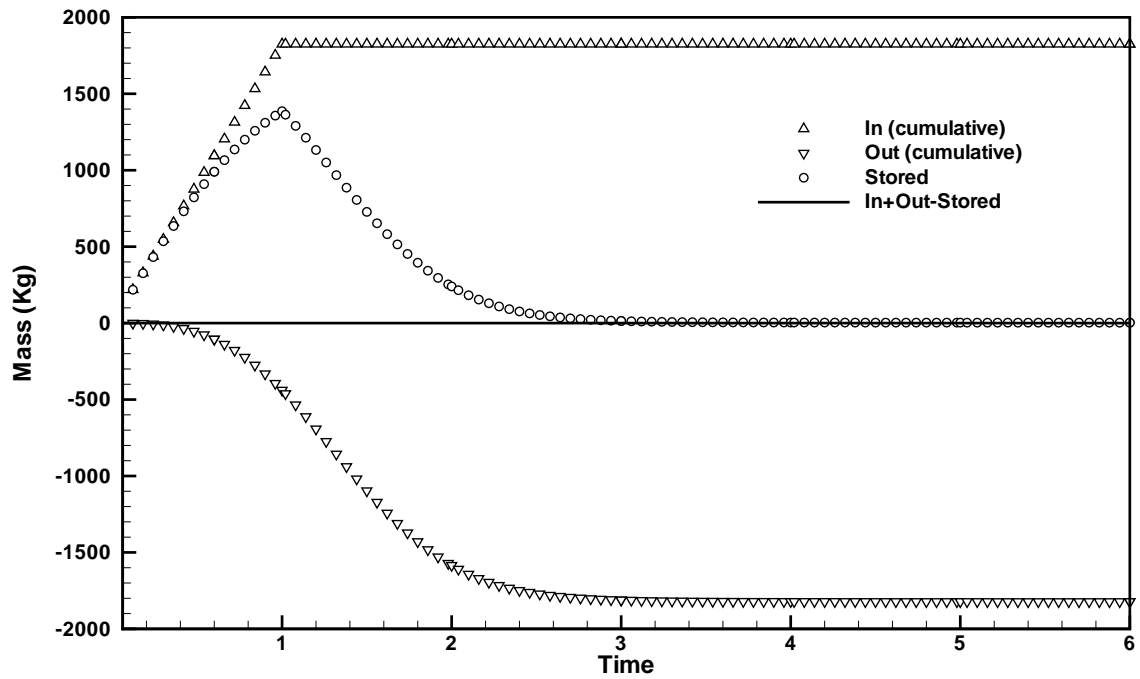
Figure 4.43: Mass Budget for the 2-D Transport Simulation.

No mass flux

Constant head: h=20 m

No mass flux

Q_i= 0.15 m³/s
C_i=100 ppb
t_i= 1 year

Injection well
○

Q_p= Transient

Pumping well
○

Low
conductivity zone
K= 1.474 E-6 m/s

Confined aquifer: Total thickness = 25 m
K= 1.474 E-3m/s
S=0.01

Constant head: h=16 m

1400 m

No mass flux

1800 m
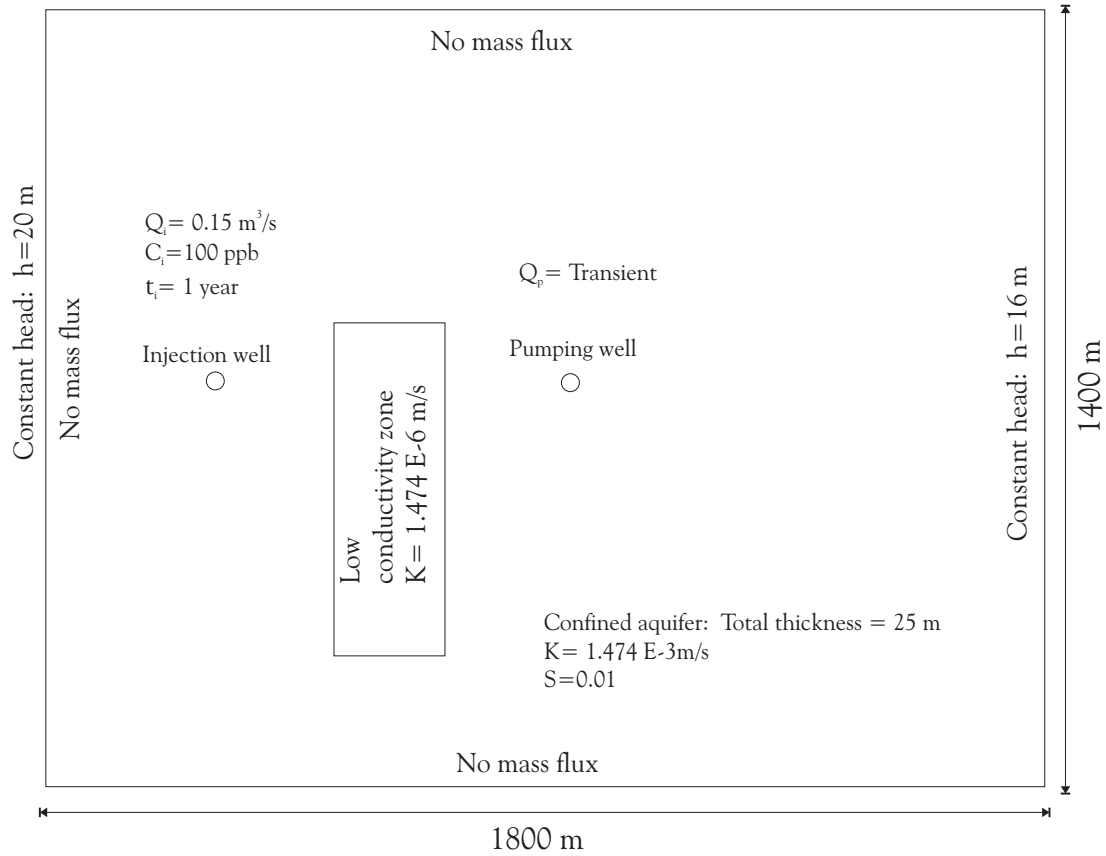
Figure 4.44: Problem Description for 2-D Transport.

4.4.10   Level 2: Two-Dimensional Transport of a Contaminant Plume in a Heterogeneous Confined Aquifer

This problem has the same setting as the one presented in Section 4.4.9 except that the simulation is done within a transient flow field. The total thickness of the aquifer is 25 m. The set-up is shown in Figure 4.44. 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 4.4.9. The pumping rate is 0.24 m³ s^{-1} 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 4.45. The comparison is made with MODFLOW.

Solute concentrations at the pumping well were evaluated during the simulation and are presented in Figure 4.46. **HydroGeoSphere** results compare fairly well with the break-
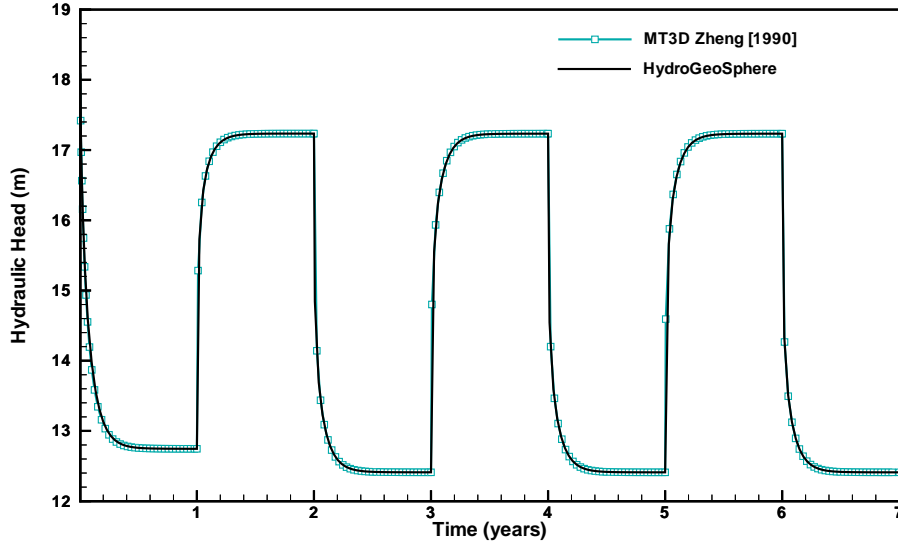
Figure 4.45: Hydraulic Head Distribution at the Pumping Well During the Simulation.

through obtained from MT3D. The solute mass balance evaluated in Figure 4.47 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.

4.4.11   Level 2: Two-Dimensional Transport of Contaminant in the Water Phase of an Unsaturated Rectangular Soil Slab

This problem concerns transport of a non-conservative solute in a transient 2-D unsaturated flow field. The 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 4.48. 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 with constant cell spacing of 1cm. The soil is assumed to be homogeneous and isotropic. Table 4.20 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 4.21. 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 days and are presented in Figure 4.49.
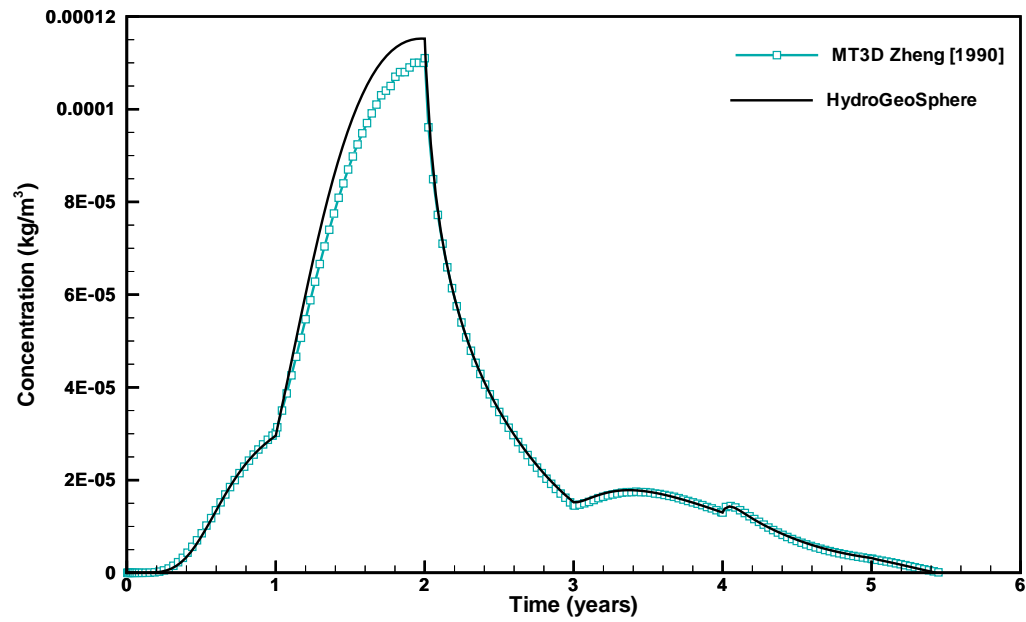
Figure 4.46: Breakthrough Curve Observed at the Pumping Well for 2-D Transport Simulation.
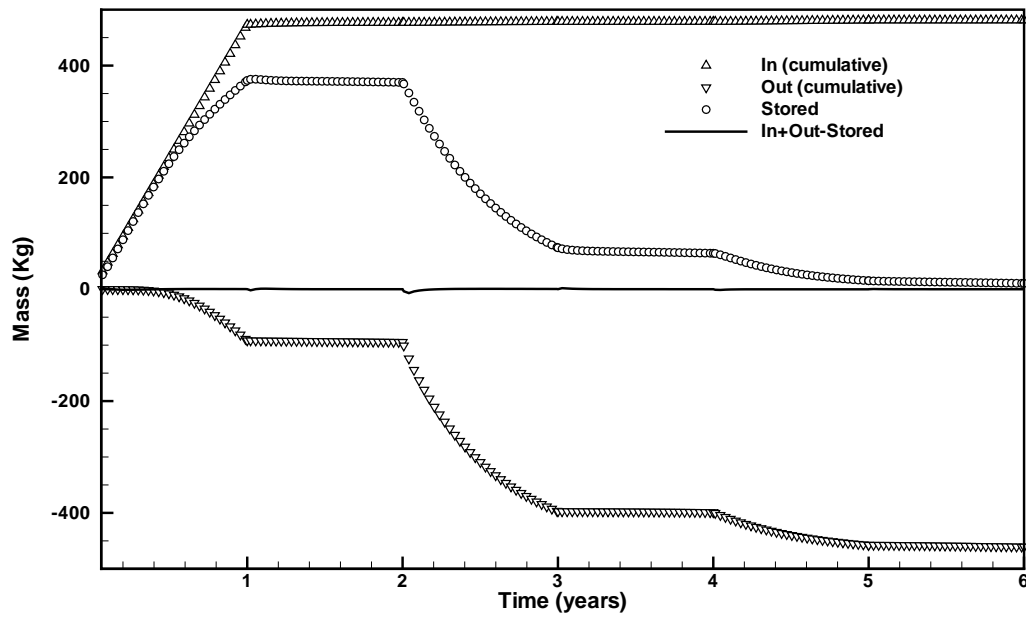
Figure 4.47: Solute Mass Balance for 2-D Transport Simulation in Transient Flow Field.
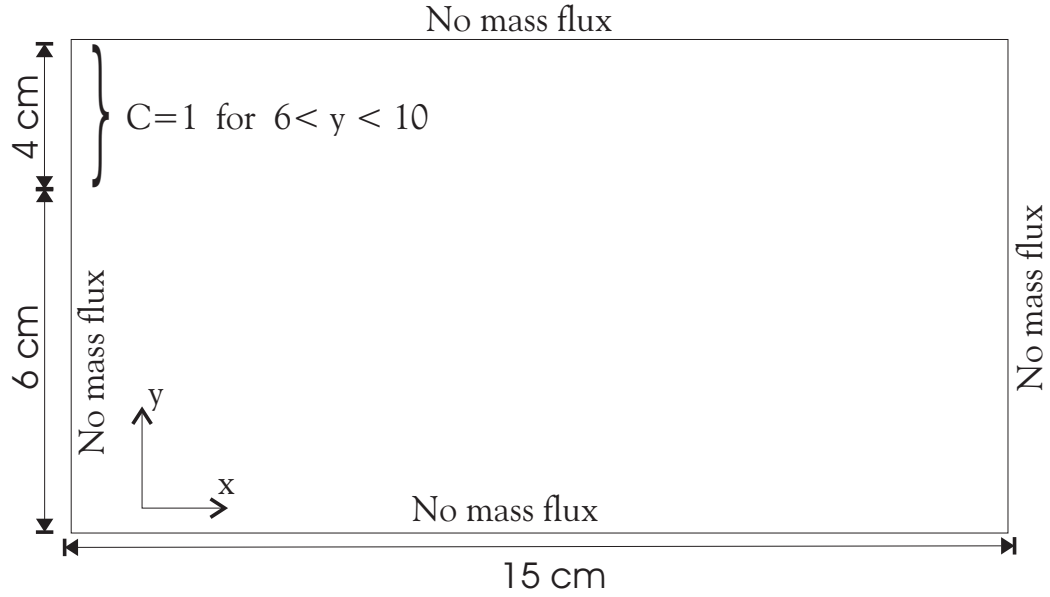
Figure 4.48: Problem Description for 2-D Transport in an Unsaturated Rectangular Soil Slab.

Table 4.20: Hydraulic Properties of the Rectangular Soil Slab.

Parameter	Value	Unit
Saturated hydraulic conductivity, K	1	cm day^{-1}
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 4.21: Physical Parameters Values for Simulation of Transport in an Unsaturated Rectangular Soil Slab.

Parameter	Value	Unit
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 day^{-1}
Decay constant	0.001	day^{-1}
Bulk density	1.46	g cm^{-3}
Distribution coefficient K_d	0.308	cm^3 g^{-1}
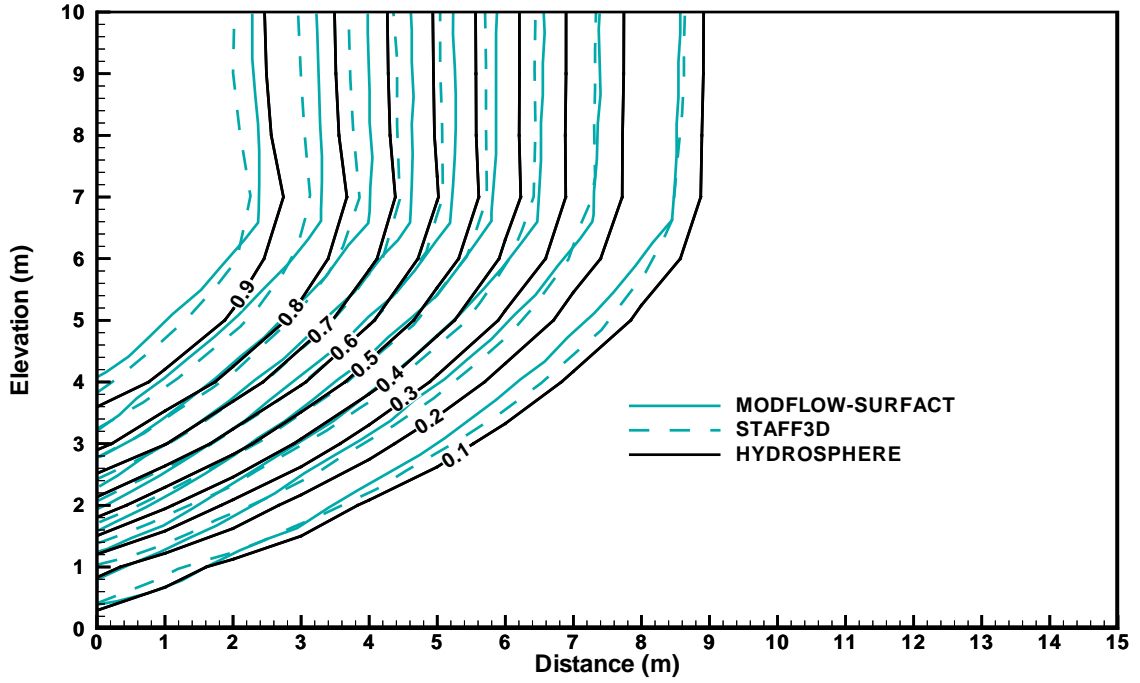
Figure 4.49: Simulated Contaminant Concentrations in an Unsaturated Rectangular Soil Slab at 0.508d.

4.5 Variable-Density Flow

4.5.1 Level 2: Variable-Density Flow in Porous Media, Elder's Problem

Variable-density flow and transport in porous media was verified in two dimensions using the *Elder* [1967] salt convection problem. The results presented by *Frolkovič and De Schepper* [2000] were chosen to be compared to those of **HydroGeoSphere**, and are considered more trustworthy since both numerical models use the control volume finite element method as well as the same flow variable (fluid pressure, P).

Frolkovič and De Schepper [2000] carried out their numerical simulations in the half domain of the symmetric Elder problem, using an extremely fine grid consisting of 32,768 nodes. All simulations used implicit transport time weighting, as is common in other variable-density simulations, and full upstream weighting as proposed by *Frolkovič and De Schepper* [2000].

The physical parameters given by *Oswald and Kinzelbach* [2004] were used and are shown in Table 4.22.

Their results are in very good visual agreement with those from the **HydroGeoSphere** model (Figure 4.50).
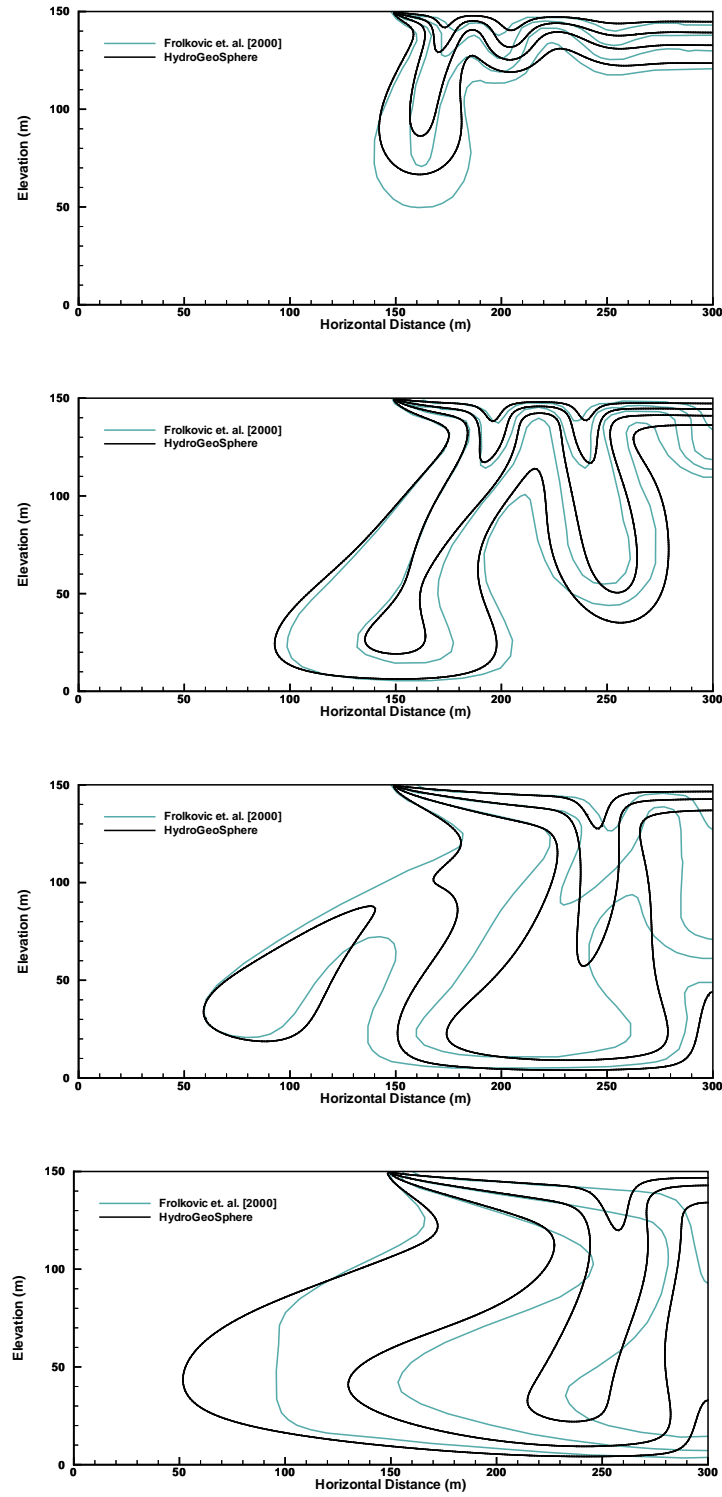
Figure 4.50: Results of the Elder problem for an extremely fine grid (256×128 elements in the half domain) at 2.5, 5, 10 and 20 years simulation time. Shown are the 20%, 40%, 60% and 80% contours.

Table 4.22: Parameters used for the Saltpool_1 Simulation

Parameter	Value	Unit
Free-solution diffusion coefficient (D_d)	3.56×10^{-6}	$m^2\ s^{-1}$
Brine density (ρ_{max})	1200	$kg\ m^{-3}$
Reference density (ρ_0)	1000	$kg\ m^{-3}$
Fluid dynamic viscosity (μ)	1.124×10^{-3}	$kg\ m^{-1}\ s^{-1}$
Hydraulic conductivity (K)	150	$m\ yr^{-}1$
Porosity (n)	0.1	
Longitudinal dispersivity (α_L)	0.0	m
Transverse dispersivity (α_T)	0.0	m

Table 4.23: Parameters used for the Saltpool_1 Simulation

Parameter	Value	Unit
Free-solution diffusion coefficient (D_d)	8.7×10^{-10}	$m^2\ s^{-1}$
Brine density (ρ_{max})	1005.9	$kg\ m^{-3}$
Reference density (ρ_0)	998.23	$kg\ m^{-3}$
Fluid dynamic viscosity (μ)	1.002×10^{-3}	$kg\ m^{-1}\ s^{-1}$
Hydraulic conductivity (K)	9.325×10^{-3}	$m\ s^{-}1$
Porosity (n)	0.372	
Longitudinal dispersivity (α_L)	0.0012	m
Transverse dispersivity (α_T)	0.00012	m

4.5.2 Level 3: Variable-Density Flow in Porous Media, Saltpool Experiment

A new benchmark problem for variable-density transport in 3D has been presented by *Oswald and Kinzelbach* [2004]. This problem is based on the three-dimensional variable-density flow and solute transport experiments in porous media conducted by *Oswald* [1999]. In these experiments, a 0.2 m × 0.2 m × 0.2 m closed box initially contained saltwater from the bottom up to 6 cm, with the rest of the box filled with freshwater. A constant freshwater recharge through one upper corner of the box disturbed this stable layering of two miscible fluids. The concentration of the mixed fluid versus time was measured at the discharging open hole on the opposite side of the input location. *Oswald* [1999] used two different initial concentrations $c_{01} = 0.01$ (case 1) and $c_{02} = 0.1$ (case 2). The experimental results were numerically reproduced by *Johannsen et al.* [2002], who also present tabular data of the measured concentrations versus time.

The **HydroGeoSphere** model output was compared in three dimensions with experimental results of *Oswald* [1999], given in *Johannsen et al.* [2002]. The physical parameters given by *Oswald and Kinzelbach* [2004] were used and are shown in Table 4.23.

The first problem of the lower initial concentration 0.01 (case 1) was used because *Johannsen et al.* [2002] showed that, in this case, grid convergence is achieved with a relatively coarse grid, whereas for case 2, the solution converged only for a very fine grid, consisting of at least
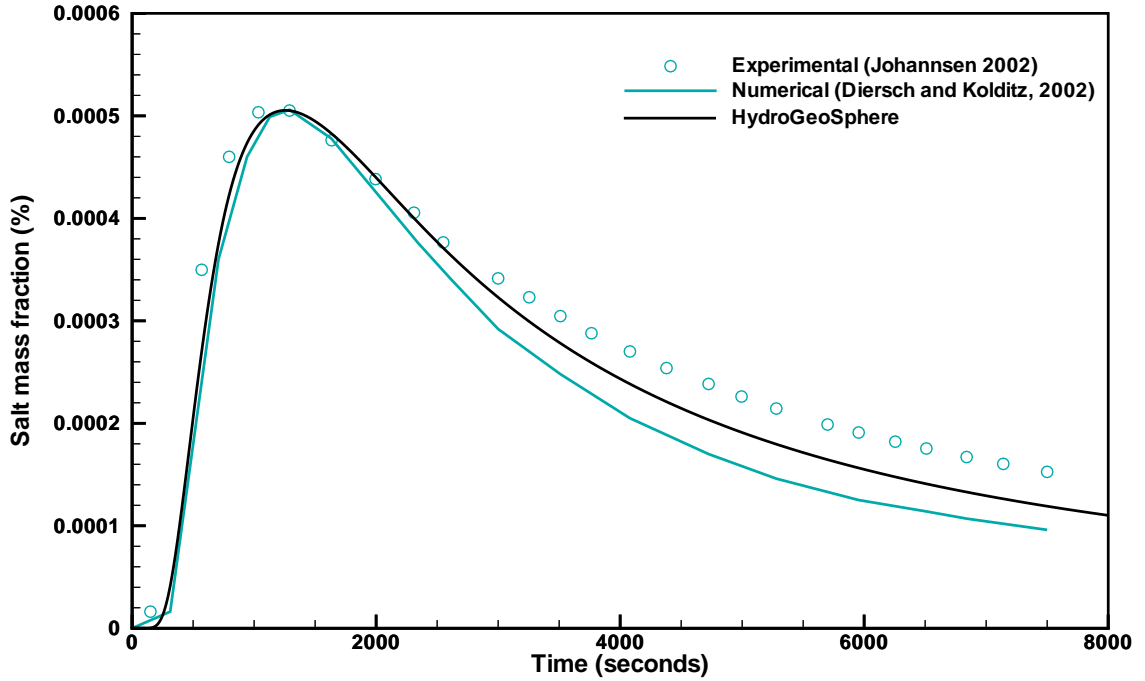
Figure 4.51: Results of three-dimensional variable-density transport simulations in porous media.

274,625 grid points [*Johannsen et al.*, 2002]). Good agreement between the experimental results from *Oswald* [1999], the numerical results from *Diersch and Kolditz* [2002] and the **HydroGeoSphere** model was obtained (Figure 4.51). The long-term results of this low density case more closely resemble the experimental data than in *Diersch and Kolditz* [2002]; however, differences remain.

4.5.3   Level 2: Variable-Density Flow in Fractured Porous Media

Variable-density flow in vertical fractures was verified by reproducing the results presented by *Shikaze et al.* [1998]. The trial which includes only vertical fractures was used as a test case. The external hydraulic heads on both aquifer top and bottom were set to zero because *Shikaze et al.* [1998] showed that density effects are best accounted for if the imposed head gradient vanishes. Otherwise, the effect of forced convection may suppress free convection. In the numerical simulations, the left and right boundaries were assumed to be impermeable for flow. The top of the domain is assumed to be a salt lake with a constant concentration equal to 1.0. All other boundaries for transport are zero dispersive-flux boundaries. The physical parameters used are identical to those presented in *Shikaze et al.* [1998] and summarized in Table 4.24. The 3D domain is of size $\ell_x = 10$ m, $\ell_y = 1$ m and $\ell_z = 10$ m. The spatial discretization used was 0.025 m in both the x- and the z-direction and unity
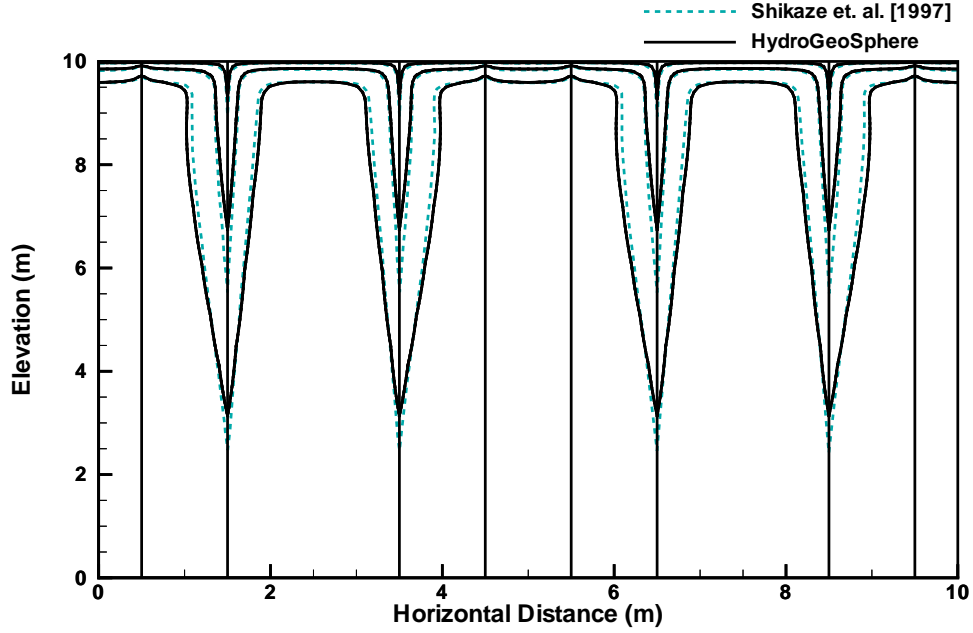
Figure 4.52: Variable-density flow in a set vertical fractures embedded in a porous matrix. Shown are the concentration contours 0.1 to 0.9 with a contour interval of 0.4 at 2 years simulation time.

in the y-direction. Fracture spacings are nonuniform as shown in Figure 4.52. The Figure shows excellent agreement between the concentration distributions calculated by the two numerical models.

4.6 Heat Transfer

4.6.1 Level 1, 2: Heat Transfer in Porous Media

The first test case verifies 1D heat transfer in an unfractured porous matrix. A constant velocity along the flow axis is imposed. The impact of temperature on fluid properties is ignored, which linearizes the problem. Thermal energy is transported by way of conduction, advection and mechanical dispersion. In this case, the governing equation can be written in the form:

$$D_{th}\frac{\partial^2 T}{\partial x^2} - v_{th}\frac{\partial T}{\partial x} = \frac{\partial T}{\partial t} \tag{4.6}$$

Table 4.24: Model parameters used in fractured media studies. All parameters are identical to those used by *Shikaze et al.* [1998].

Parameter	Value	Unit
Free-solution diffusion coefficient (D_d)	5×10^{-9}	$\text{m}^2 \text{ s}^{-1}$
Brine density (ρ_{max})	1200	kg m^{-3}
Reference density (ρ_0)	1000	kg m^{-3}
Fluid compressibility (α_{fl})	4.4×10^{-10}	$\text{kg}^{-1} \text{ m s}^2$
Matrix compressibility (α_m)	1.0×10^{-8}	$\text{kg}^{-1} \text{ m s}^2$
Fluid dynamic viscosity (μ)	1.1×10^{-3}	$\text{kg m}^{-1} \text{ s}^{-1}$
Matrix permeability (κ_{ij})	10^{-15}	m^2
Matrix longitudinal dispersivity (α_l)	0.1	m
Matrix transverse dispersivity (α_t)	0.005	m
Matrix porosity (ϕ)	0.35	
Tortuosity (τ)	0.1	
Fracture dispersivity (α^{fr})	0.1	m
Fracture aperture ($2b$)	50	μm

where D_{th} [$\text{L}^2 \text{ T}^{-1}$] is the thermal dispersion coefficient:

$$D_{th} = \frac{k_b + \phi D_{xx} \rho_l \tilde{c}_l}{\rho_b \tilde{c}_b} \tag{4.7}$$

and v_{th} [L T^{-1}] is the retarded velocity:

$$v_{th} = q \cdot \frac{\rho_l \tilde{c}_l}{\rho_b \tilde{c}_b} = q \cdot \frac{1}{\phi R_{th}} \tag{4.8}$$

with the thermal retardation coefficient R_{th} [–] [*Molson et al.*, 1992]:

$$R_{th} = 1 + \frac{(1 - \phi)\rho_s \tilde{c}_s}{\phi \rho_l \tilde{c}_l} \tag{4.9}$$

Equation (4.6) has the standard parabolic-hyperbolic form of a 1D partial differential equation. Therefore, if (4.6) is subject to the Dirichlet boundary condition, $T = T_1$, the solution is the Ogata-Banks analytical solution:

$$\frac{T - T_0}{T_1 - T_0} = \frac{1}{2}\left[\text{erfc}\left(\frac{x - v_{th}t}{2\sqrt{D_{th}t}}\right) + \exp\left(\frac{v_{th}x}{D_{th}}\right)\text{erfc}\left(\frac{x + v_{th}t}{2\sqrt{D_{th}t}}\right)\right] \tag{4.10}$$

where T_0 is the initial temperature in the domain.

In the numerical simulation, the finite element domain was spatially discretized by using 20 uniform blocks in the flow direction. All simulation parameters are given by Table 4.25. The developed numerical model is compared with the analytical solution (4.10) as well as with numerical results presented by *Ward et al.* [1984] who used the code SWIFT. The results are depicted in Figure 4.53.
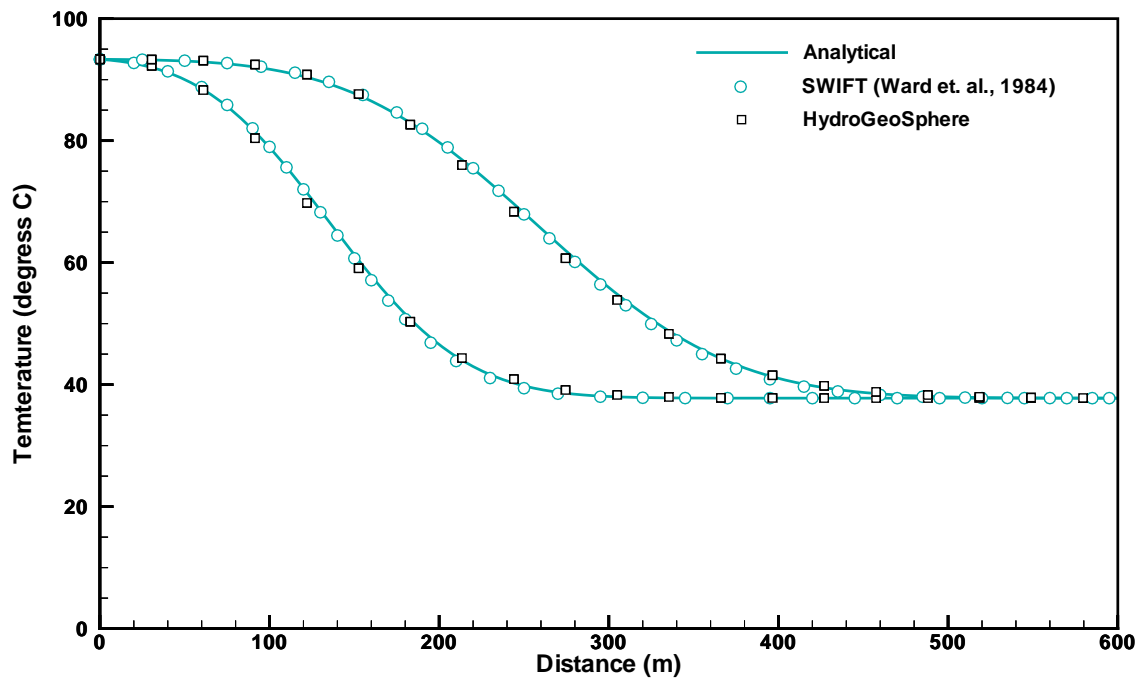
Figure 4.53: Temperature profiles of 1D heat transfer in an unfractured porous matrix (example 1). Shown are the temperatures in the matrix at 2,148 (left) and 4,262 (right) days.

Table 4.25: Model parameters used in the verification example for 1D heat transfer in an unfractured porous matrix. All parameters are identical to those used by *Ward et al.* [1984].

Parameter	Value	Unit
Bulk thermal conductivity (k_b)	2.16	kg m s^{-3} K^{-1}
Heat capacity of solid ($\tilde{c}_s$)	1254.682	m^2 s^{-2} K^{-1}
Solid density (ρ_s)	1602	kg m^{-3}
Heat capacity of water ($\tilde{c}_l$)	4185	m^2 s^{-2} K^{-1}
Fluid density (ρ_l)	1000	kg m^{-3}
Matrix porosity (ϕ)	0.1	
Longitudinal dispersivity (α_l)	14.4	m
Heat dispersion coefficient[a] (D_{th})	1.15×10^{-5}	m^2 s^{-1}
Thermal retardation coefficient[b] (R_{th})	5.323	
Darcy flux (q)	3.53×10^{-7}	m s^{-1}
Retarded velocity[c] (v_{th})	6.63×10^{-7}	m s^{-1}
Initial temperature (T_0)	37.78	°C
Boundary temperature (T_1)	93.33	°C
Domain size (ℓ_x)	600	m
Output times (t_1, t_2)	2148, 4262	days

[a] from relation (4.7)
[b] from relation (4.9)
[c] $q/(\phi R^{th})$

4.6.2   Level 1: Heat Transfer in Fractured Media

The second test case verifies advective-conductive-dispersive 1D heat transfer in a single fracture embedded in an impermeable matrix. As in the previous case, a constant flow velocity along the axis is imposed and fluid properties are kept constant in order to linearize the problem. Hence, the governing equation simplifies to:

$$D_{th}^{fr} \frac{\partial^2 T^{fr}}{\partial z^2} - v^{fr} \frac{\partial T^{fr}}{\partial z} = \frac{\partial T^{fr}}{\partial t} \qquad (4.11)$$

where D_{th}^{fr} [L^2 T^{-1}] is the fracture thermal dispersion coefficient, given by:

$$D_{th}^{fr} = \frac{k_l}{\rho_l \tilde{c}_l} + D_{zz}^{fr} \qquad (4.12)$$

and where v^{fr} is the constant groundwater flow velocity along the fracture. Equation (4.11) is a standard 1D parabolic-hyperbolic partial differential equation. If the Dirichlet boundary condition $T^{fr} = T_1^{fr}$ is imposed on the fracture inlet, the Ogata-Banks analytical solution is now:

$$\frac{T^{fr} - T_0^{fr}}{T_1^{fr} - T_0^{fr}} = \frac{1}{2} \left[\text{erfc} \left(\frac{z - v^{fr}t}{2\sqrt{D_{th}^{fr}t}} \right) + \exp \left(\frac{v^{fr}z}{D_{th}^{fr}} \right) \text{erfc} \left(\frac{z + v^{fr}t}{2\sqrt{D_{th}^{fr}t}} \right) \right] \qquad (4.13)$$

where T_0^{fr} is the initial temperature in the fracture.
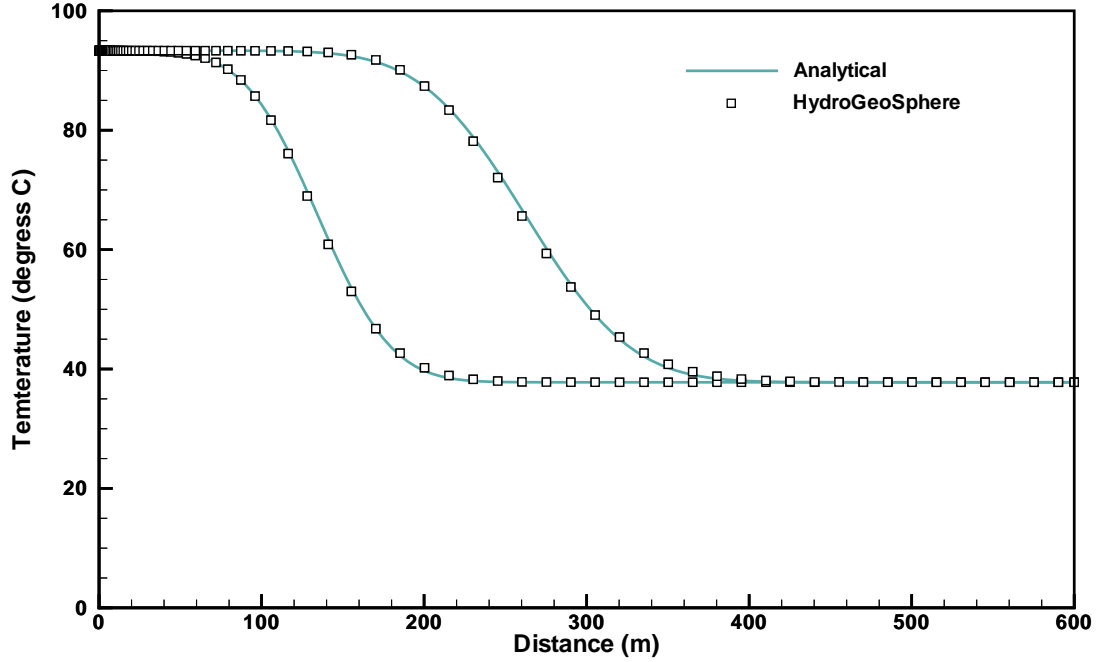
Figure 4.54: Temperature profiles of 1D heat transfer in a single fracture within an impermeable matrix (example 2). Shown are the temperatures in the fracture at 2,148 (left) and 4,262 (right) days.

In the numerical simulation, the finite element domain was spatially discretized along the fracture by using element sizes that gradually increase by the factor 1.1 from $\Delta z = 0.1$ m near the elevated temperature to $\Delta z = 15$ m at the domain boundary. The groundwater velocity in the fracture was set to 7.0×10^{-7} m s^{-1} and the fracture dispersivity used was 5.0 m, giving the fracture thermal dispersion coefficient as 3.62×10^{-6} m^2 s^{-1}. All other parameters are identical to those used in the previous example and given in Table 4.25. The simulation results are depicted in Figure 4.54.

4.6.3 Level 1: Heat Transfer in Fractured Porous Media

The third test case verifies 2D heat transfer in a single fracture embedded in a porous matrix. This verification example is based on analytical results presented by *Meyer* [2004], who investigated advective transient heat transfer in a fracture while in the porous matrix, heat is transported due to conduction alone. Mechanical heat dispersion as well as conduction within the fracture are not considered, making numerical integration unnecessary. The groundwater flow velocity in the fracture is constant. Under these assumptions, the

governing equations of this problem simplify to:

$$\rho_b \tilde{c}_b \ \frac{\partial T}{\partial t} - k_b \ \frac{\partial^2 T}{\partial x^2} = 0 \qquad\qquad b \leq x \leq \infty \qquad\qquad (4.14)$$

and

$$\rho_l \tilde{c}_l \ \frac{\partial T^{fr}}{\partial t} + \rho_l \tilde{c}_l \ v^{fr} \ \frac{\partial T^{fr}}{\partial z} - \frac{k_b}{b} \ \frac{\partial T^{fr}}{\partial x}\bigg|_{x=b} = 0 \qquad 0 \leq z \leq \infty \qquad (4.15)$$

for heat transport in the matrix and in the discrete fracture, respectively. The last term in (4.15) expresses conductive loss of heat from the fracture into the matrix on the fracture-matrix interface. Initially, the entire system has the uniform temperature, T_0. The fluid entering the fracture has the constant temperature, T_1. All boundaries, except the fracture inlet and outlet, are impermeable for groundwater flow and for heat exchange. According to *Meyer* [2004], the transient solution along the fracture is:

$$\frac{T^{fr} - T_0}{T_1 - T_0} = \mathrm{erfc}\left(\frac{z \ \sqrt{k_b \ \rho_b \tilde{c}_b}}{2v^{fr}\rho_l \tilde{c}_l b \ \sqrt{(t - z/v^{fr})}}\right) \qquad\qquad (4.16)$$

Using the analytical results presented by *Tang et al.* [1981], it can be shown that the transient solution along a cross-section from the fracture into the porous matrix is given by

$$\frac{T - T_0}{T_1 - T_0} = \mathrm{erfc}\left(\frac{z \ \sqrt{k_b \ \rho_b \tilde{c}_b}}{2v^{fr}\rho_l \tilde{c}_l b \ \sqrt{(t - z/v^{fr})}} + \frac{\sqrt{\rho_b \tilde{c}_b} \ (x - b)}{2\sqrt{k_b} \ \sqrt{(t - z/v^{fr})}}\right) \qquad (4.17)$$

The fracture-matrix system used is identical to that shown in Figure 4.55. The finite element domain was spatially discretized in the x-direction by gradually increasing Δx with constant factor 1.1 from $\Delta x = 0.01$ m near the fracture to $\Delta x = 0.1$ m at the domain boundary. In the flow direction, Δz also increases gradually from $\Delta z = 0.1$ m near the elevated temperature to $\Delta z = 0.5$ m at the domain boundary. All other parameters are presented in Table 4.26 and the simulation results are exhibited in the Figures 4.56 and 4.57.

4.6.4   Level 2: Heat Transfer in Anisotropic Porous Media

The last verification problem for heat transfer is the 2D field scale example presented by *Yang and Edwards* [2000], and includes all heat transfer mechanisms in both continua with variable fluid properties. This test case represents a realistic scenario of radioactive waste disposal in the low-permeability anisotropic granitic rock of the Canadian Shield [*Davison et al.*, 1994a]. Figure 4.58 shows the conceptual model, a vertical slice of dimensions 2,000 m $\times$ 1,000 m with a unit thickness. Radionuclides are disposed of in a 1,300 m long horizontal vault at a depth of 500 m below surface. The simulation domain consists of three anisotropic porous layers. The radioactive waste represents an exponentially decreasing heat source due to remaining radioactivity [*Davison et al.*, 1994a]. Thus, the term $\Gamma = 11.59$ kg m^{-1} s$^{-3} \cdot$ exp(-5.5$\times 10^{-10}$ s$^{-1} \cdot t$), as given by *Yang and Edwards* [2000], was added as a heat sink term to the left hand side of the governing equation. All boundaries are impermeable for flow.
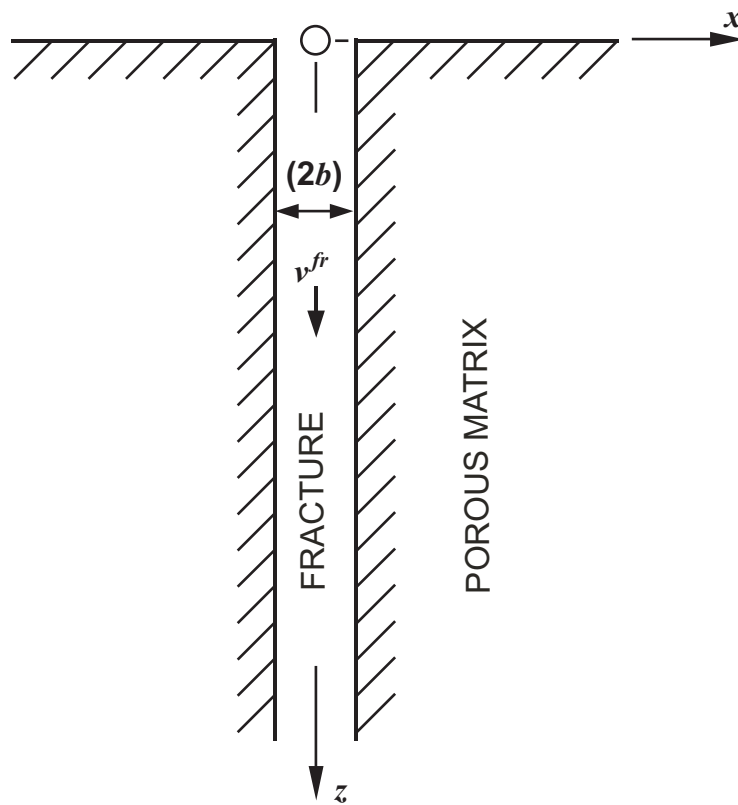
Figure 4.55: Fracture-matrix system used for model verification (Tang et al., 1981).
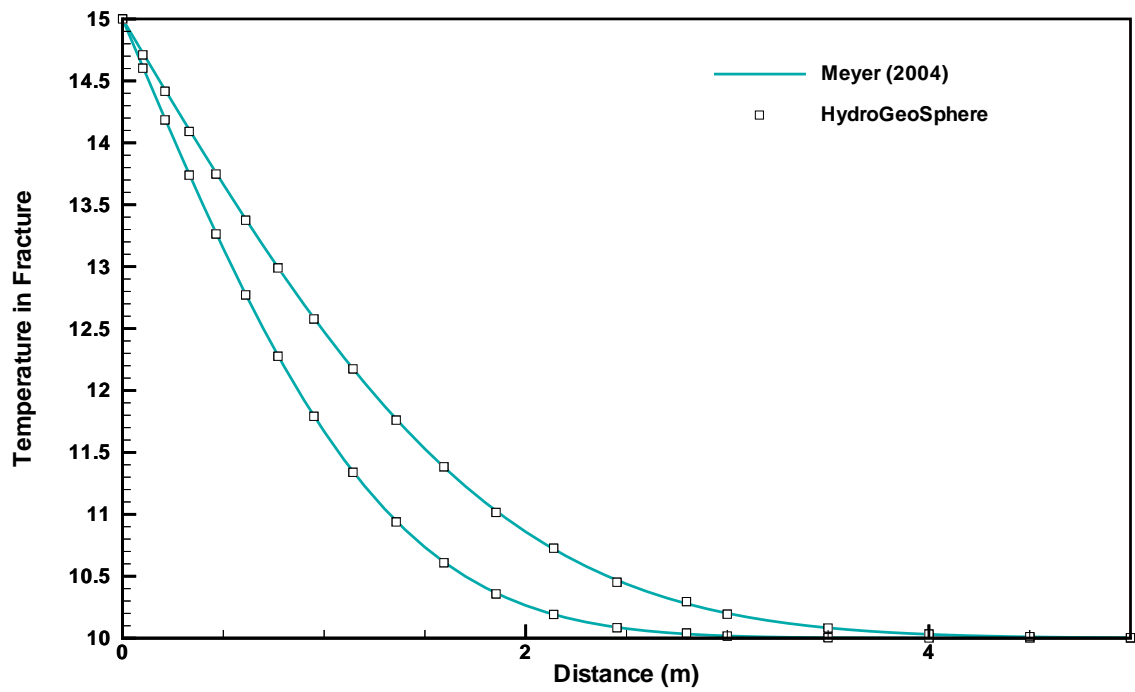
Figure 4.56: Temperature profiles of 1D heat transfer in discretely-fractured porous media. Shown are the temperatures in the fracture at 5,000 (left) and 10,000 (right) seconds.
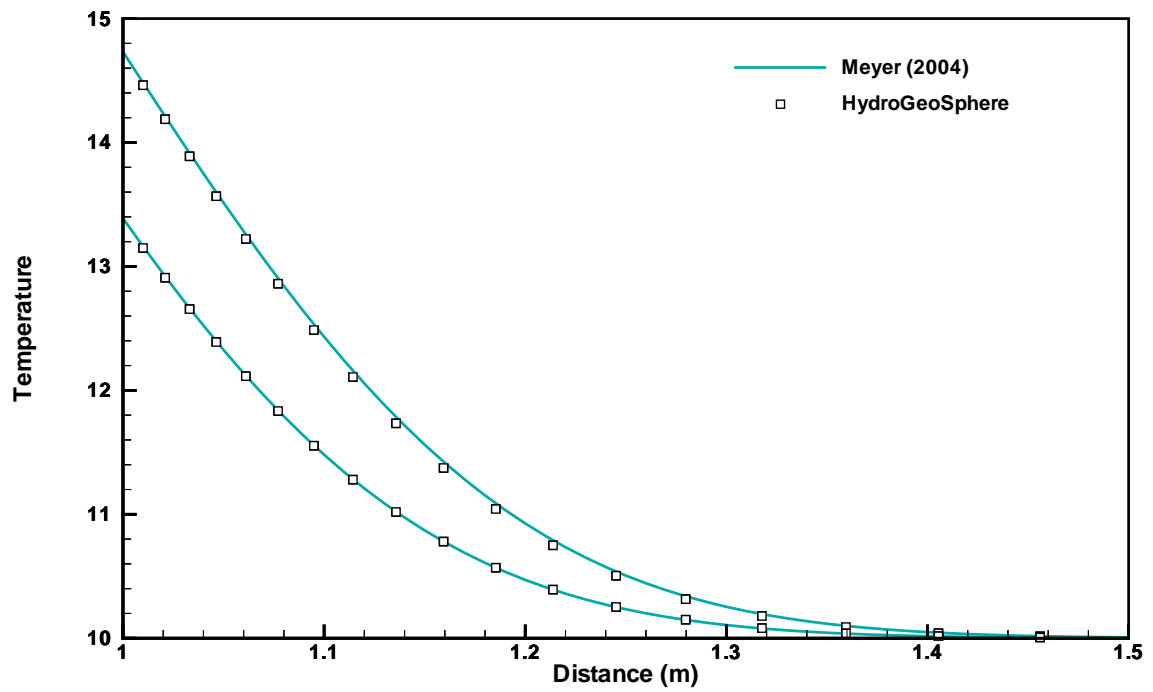
Figure 4.57: Temperature profiles of 1D heat transfer in discretely-fractured porous media. Shown are the temperatures in the matrix at 10,000 seconds simulation time at the distances 0.1 (left) and 0.61 (right) m from the fracture.
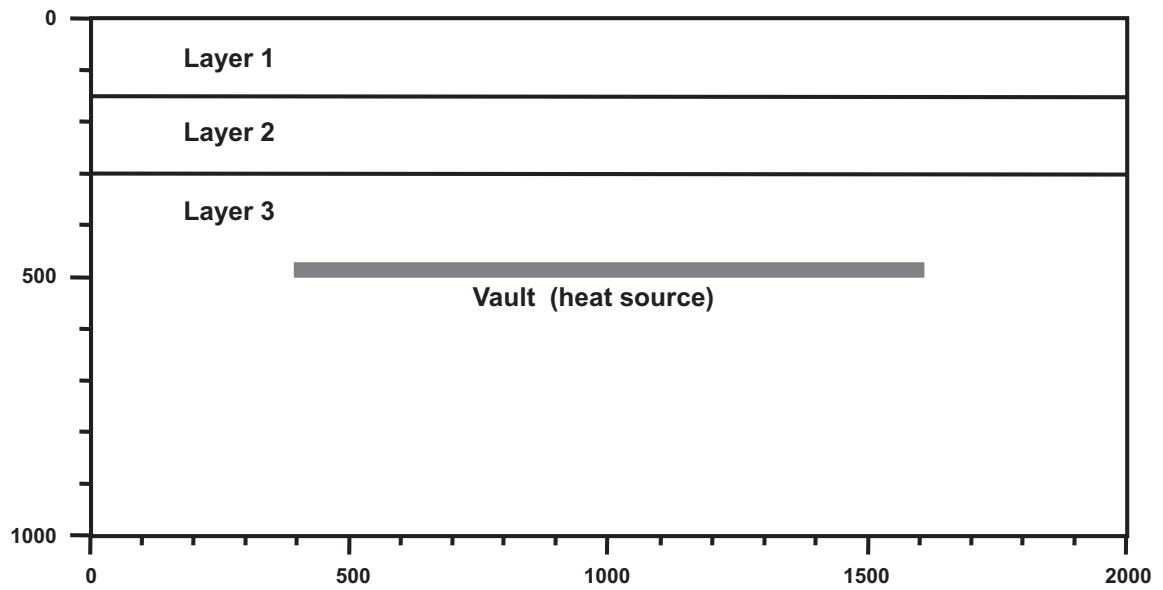
Figure 4.58: The conceptual model for variable-density heat transfer in anisotropic porous media (example 4; Yang and Edwards, 2000). The heat source in the vault is due to the remaining radioactivity of the stored waste. Top and bottom boundaries are assigned the constant temperatures 6ºC and 17.5ºC, respectively, with the corresponding geothermal gradient 11.5 K km^{-1}.

Table 4.26: Model parameters used in the verification example for 2D heat transfer in a single fracture embedded in a porous matrix. All parameters are identical to those used by Meyer [2004].

Parameter	Value	Unit
Bulk thermal conductivity (k_b)	3.4	kg m s^{-3} K^{-1}
Heat capacity of solid $(\tilde{c}_s)$	908	m^2 s^{-2} K^{-1}
Solid density (ρ_s)	2550	kg m^{-3}
Heat capacity of water $(\tilde{c}_l)$	4192	m^2 s^{-2} K^{-1}
Fluid density (ρ_l)	997	kg m^{-3}
Matrix porosity (ϕ)	0.2	
Groundwater flow velocity in the fracture (v^{fr})	0.05	m s^{-1}
Initial temperature (T_0)	10	°C
Boundary temperature (T_1)	15	°C
Domain size (ℓ_x, ℓ_z)	2, 10	m
Location of cross-sections (z_1, z_2)	0.1, 0.61	m
Output times (t_1, t_2)	5000, 10000	s

Top and bottom boundaries have constant temperatures to mimic a geothermal gradient of 11.5 K km^{-1}, which is natural in the study area. All other boundaries are impermeable for heat transfer. Initially, the geothermal field is undisturbed with horizontal isotherms.

In the numerical simulations carried out with **HydroGeoSphere**, the temperature is assumed to have an impact on both fluid properties density and viscosity. This conforms with the assumption made by *Yang and Edwards* [2000]. Chemical reactions are not considered. All model parameters are summarized in Table 4.27. The variable-density, variable-viscosity flow and heat transfer results are exhibited in Figure 4.59, which shows excellent agreement between the two numerical models.

4.6.5 Level 2: Borden thermal injection experiment

Subsurface thermal energy transport is verified by comparing results from HydroGeoSphere with the model verification example used by *Molson et al.* [1992]. This verification example involves a simulation of the Borden thermal injection experiment. A description of the experiment and the observed data are presented by *Palmer et al.* [1992]. Unless otherwise stated, all parameters used in the HydroGeoSphere simulation are identical to those used by *Molson et al.* [1992]. The domain size was $40 \times 30 \times 20$ m, and was discretized in all directions using 0.5 m block elements. The initial temperature of the domain varied from 15 °C at ground surface to 9 °C 6 m below the surface. The injection well was located at x = 12 m, y = 15 m, z = 16 m and water was injected at a temperature of 37 °C for the first 6 days. The flow and transport parameters are given in Table I. The aquifer thermal parameters are listed in Table II. The simulation was run for 76 days, and the results are presented at 9 days, 27 days and 76 days for a 2-D longitudinal cross-section through the injection well (Figure 1). The results from the HydroGeoSphere simulation are compared to
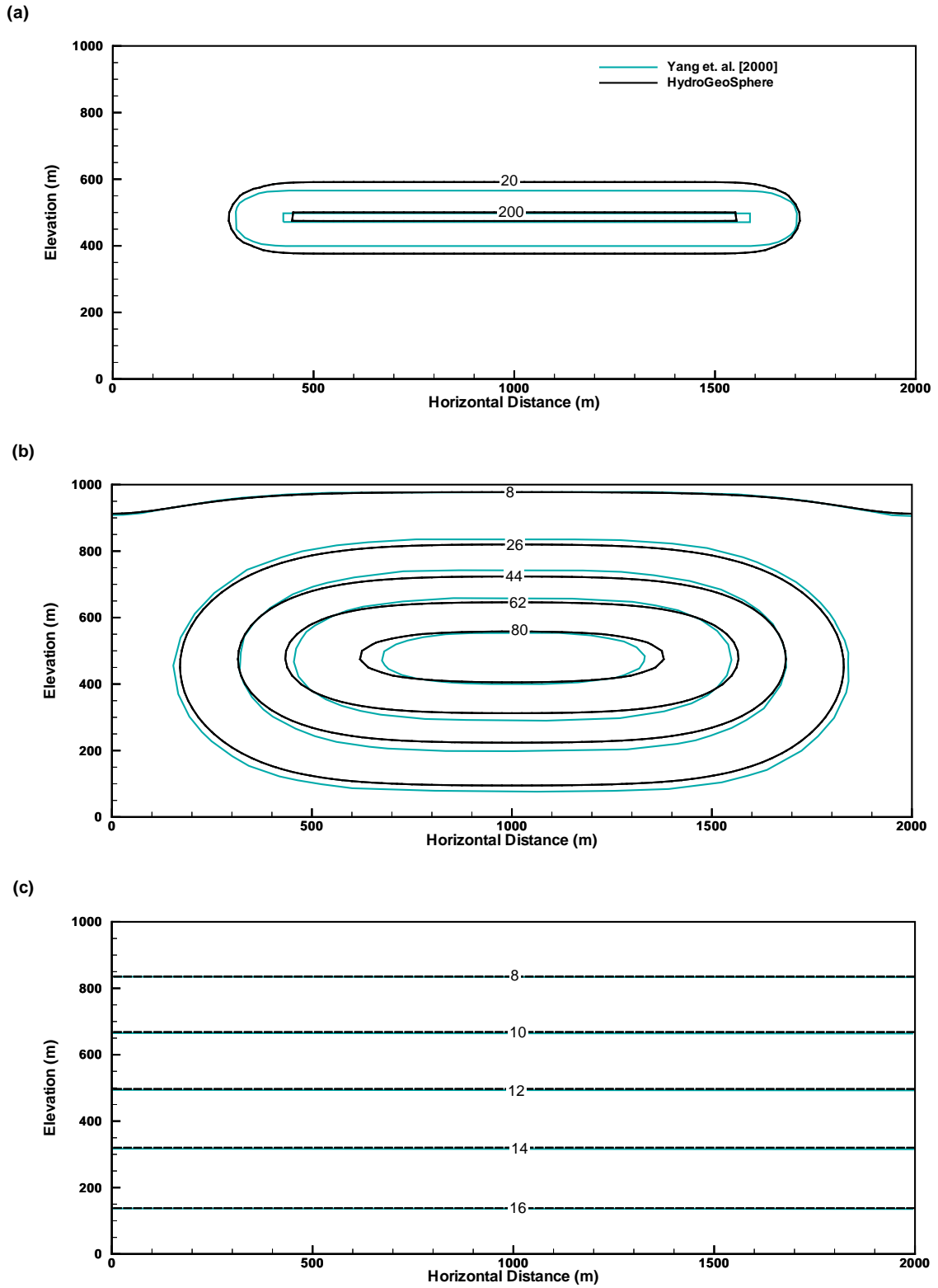
Figure 4.59: Evolution of temperature in anisotropic porous media with an exponentially decreasing heat source. Simulation times are (a) 10^4 days, (b) 3×10^5 days, (c) 7×10^6 days. Shown are isotherms in degrees Celsius.

Table 4.27: Model parameters used in the verification example for 2D variable-density thermal flow and heat transfer in anisotropic porous media. All parameters are identical to those used by *Yang and Edwards* [2000].

Parameter	Value	Unit
Bulk thermal conductivity (k_b)	2.0	kg m s^{-3} K^{-1}
Heat capacity of solid $(\tilde{c}_s)$	800	m^2 s^{-2} K^{-1}
Heat capacity of water $(\tilde{c}_l)$	4174	m^2 s^{-2} K^{-1}
Solid density (ρ_s)	2630	kg m^{-3}
Matrix permeability $(\kappa_{xx}, \kappa_{zz})$	Layer 1: 1.0×10^{-15}, 5.0×10^{-15}	m^2
	Layer 2: 1.0×10^{-17}, 5.0×10^{-17}	m^2
	Layer 3: 1.0×10^{-19}, 1.0×10^{-19}	m^2
Matrix porosity (ϕ)	0.004	
Domain size (ℓ_x, ℓ_z)	2000, 1000	m
Spatial discretization $(\Delta x, \Delta z)$	25, 25	m

those presented in *Molson et al.* [1992]. Figure 1 shows that the temperature results from both models agree. The minor differences that do occur are due to the different treatment of the temperature boundary condition at the surface of the domain. In addition, *Molson et al.* [1992] used temperature-dependent density and viscosity terms, whereas the HydroGeo-Sphere simulation treated these parameters as constant under the given range in subsurface temperatures.

4.7   Travel Time Probability

4.7.1   1D travel time PDF

The travel time PDF for a semi-infinite domain is the flux concentration solution of Eq. (2.128a), which is obtained by applying the boundary conditions $g_t(t,0) = \delta(t)$ and $D\frac{\partial g_t(t,x)}{\partial x}|_{x=\infty} = 0$, with $D = \alpha_L v + D_m$. This solution reads:

$$g_t(t,x) = \frac{x}{\sqrt{4\pi D t}\, t} \exp\left(-\frac{(x-vt)^2}{4Dt}\right) \tag{4.18}$$

Considering a 1D domain with outlet position at $x = L$, the backward travel time PDF is deduced by replacing x by $L - x$, to give:

$$g_t(t,x) = \frac{L-x}{\sqrt{4\pi D t}\, t} \exp\left(-\frac{(L-x-vt)^2}{4Dt}\right) \tag{4.19}$$

In Figure 4.60, the numerical forward and backward travel time PDF's are compared to this analytical solution.
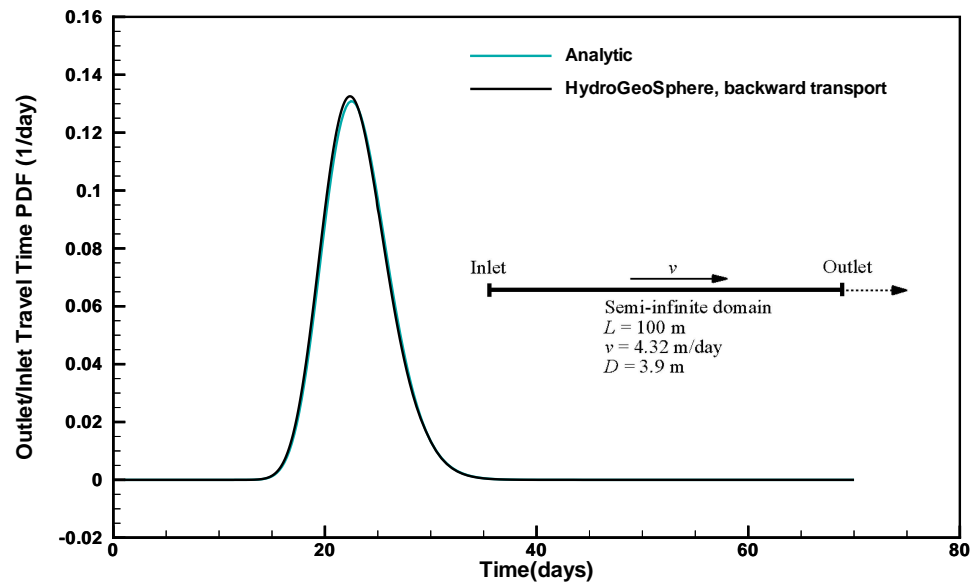
Figure 4.60: Forward and backward travel time PDF's versus analytical solution for a 1D semi-infinite domain.

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**[1]) which is then run to generate the input data files for **HydroGeoSphere**. Second, **HydroGeoSphere** is run to solve the problem and generate output data files.

The **grok** input file name consists of a meaningful prefix of up to 40 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.

Throughout the manual, we will adopt the convention of using *italics* to indicate problem-dependent, user-defined portions of filenames (e.g. prefix, species name etc.) and `typewriter font` to indicate invariant portions generated by **HydroGeoSphere**. For example, in the

---

[1] grok /grok/, var. /grohk/ vt. [from the novel "Stranger in a Strange Land", by Robert A. Heinlein, where it is a Martian word meaning literally 'to drink' and metaphorically 'to be one with'] The emphatic form is 'grok in fullness'. 1. To understand, usually in a global sense. Connotes intimate and exhaustive knowledge. Contrast zen, which is similar supernal understanding experienced as a single brief flash. See also glark. 2. Used of programs, may connote merely sufficient understanding. "Almost all C compilers grok the void type these days."

filename *prefix*o.concentration.*species*.001 the *prefix* and *species* portions would be the user-defined prefix and name of a solute, or species, while the o.concentration. and .001 portions would be generated by **HydroGeoSphere** automatically.

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 *prefix*.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 **HydroGeoSphere**-compatible data files

Tasks 3 and 6 are guided by instructions issued by the user in the *prefix*.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.

Here is an example instruction and some input data which illustrates some common conventions that will be used throughout the manual:

Example instruction text

1. **xl, nbx** Domain length and number of blocks in the *x*-direction

2. **xi(i),i=1,nx** *x*-coordinates of the **nx** nodes.

3. **inode...end** Node numbers.

<div align="center">● ● ●</div>

The pre-processor instruction is separated from the preceding text by a horizontal line, and is written using the sans serif font. It must be typed in the *prefix*.grok file *exactly as shown*, with the exception that it is 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.

If the instruction requires input data, there will follow a series of numbered lines, each containing bolded **variable names** and a description of what is to be read. Each numbered line will correspond to one *or more* FORTRAN read statements.

Usually, 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, but should be avoided when reading character strings (e.g. file names).

In this example, 3 items of input are required. The first item **xl, nbx** requires that the user enter a real value (i.e. domain length) followed by an integer value (i.e. number of blocks) on the first non-blank or uncommented line following the instruction.

The second item **xi(i),i=1,nx** reads **nx** real values into the array **xi**. The size of **nx** is problem dependent (e.g. number of nodes in x, number of species etc.) and it is up to the user to supply enough values to satisfy the read statement. The values may be entered on one line or spread out over multiple lines as desired. If they are entered on one line, they should be separated by spaces or commas.

Finally, the third item **inode...end** indicates a list, in this case of node numbers, that is to be read until an end card is encountered. The list values must be entered one per line.

The end of the documentation that pertains to a specific instruction is designated by 3 dots: ● ● ●.

So for this example instruction, assuming that **nx** is equal to 5, the following statements in the *prefix*.grok file would satisfy the input requirements:

```
Example instruction text
10.0    100
0.0   2.0   4.0   6.0   8.0   10.0
1
2
3
5
6
end
```

Some instructions are controlled by input routines that have their own subset of input instructions, some or all of which may be optional. For example, the instruction Solute is used to define a new solute and in it's simplest form appears as:

```
Solute
end
```

In this case, the End instruction immediately follows the Solute instruction, and no optional instructions have been issued. The End statement is required so that **grok** knows when to exit the solute definition routine. Such instructions will be indicated using the following convention:

Example instruction text...End

• • •

where the text ...End indicates that the instruction (e.g. Solute) will be followed by optional instructions or input and terminated by an End instruction.

Before **grok** processes instructions contained in a *prefix*.grok or a material properties file (see Section 5.8.1.6) it first makes a working copy of the file in which any line which is completely blank or which begins with an exclamation point(!) is removed and in which the contents of any included file are copied. This allows you to include blank lines and comments when and where required to improve the readability and clarity of the input.

Included files can be used to avoid having to cut and paste or comment and uncomment large sections of input instructions. Long lists (e.g. of node numbers or boundary condition data) and cases where various different grid generation approaches are being tried are good candidates for application of the include feature. For example, if we wanted to use include to supply data to the example given above, we could use the following instruction in *prefix*.grok:

```
Example instruction text
10.0    100
0.0   2.0   4.0   6.0   8.0   10.0
include my.node_list
```

and where the file my.node_list could contain, for example:

```
1
2
3
5
6
end
```

If you now wanted to substitute another node list you could, for example, supply different node numbers in the file my_other.node_list and then just change the file name given in the include instruction.

Included files can contain groups of instructions and input, or just bits of input for a single instruction. Only one level of include instruction is allowed, and so included files can not themselves contain include instructions.

As **grok** reads and processes the copy of the *prefix*`.grok` file it also creates the *prefix*`o.eco` file. Results of the **HydroGeoSphere** 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. If an error occurs while reading the input data, then **grok** will halt execution and issue an error message (to the screen and the *prefix*`o.eco` file) of the form:

```
INSTRUCTION: 500


**************************************
*** INPUT ERROR, HALTING EXECUTION ***
**************************************


GRID GENERATION: Unrecognized instruction


Press any key to continue
```

In this case the last instruction (i.e. 500) has, for some reason, caused an error. You should now check the input files to further investigate the cause of the problem, starting with the *prefix*`.grok` and material properties files.

5.1.1   File Process Control Options

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

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.

• • •

Skip rest

grok exits the loop for reading instructions from the *prefix*.`grok` file and proceeds to generate the **HydroGeoSphere** data files.

• • •

Pause

This instruction causes **grok** to pause at the current location in the *prefix*.`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 *prefix*.`grok` file where necessary.

5.1.2   Units and Physical Constants

The units used in the program are not preset, although a default of kilogram-metre-second units is assumed and used to define the values of certain physical constants as discussed below. The user should decide which units will be used for mass (M), length (L) and time (T) for the various input variables, issue the appropriate units instruction (or assign appropriate values for the physical constants) and then consistently use those chosen units for all other input data. For example, if you want to specify the dimensions of your domain in metres and the time at which you want a solution is 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 m s^{-1}, a pumping rate in m^3 s^{-1} etc. The program does not perform any checks to ensure unit consistency.

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}, Equation 2.3.

- Fluid density $\rho = 1000.0$ kg m^{-3}, Equation 2.13.

- Fluid viscosity $\mu = 1.124 \times 10^{-3}$ kg m^{-1} s^{-1}, Equation 2.13.

- Fluid compressibility $\alpha_w = 4.4 \times 10^{-10}$ (m s^2) kg^{-1}, Equation 2.15.

- Fluid surface tension $\chi = 0.07183$ kg s^{-2}, Equation 4.5.

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-minute

Converts the default values given above into the kilogram-metre-minute system. This instruction also converts the porous media, dual continuum, fractured media and surface flow default properties which are defined in the code. NOTE: It does not convert properties specified in any *prefix* `.grok` `.mprops`, etc. file. Similar instructions exist for converting to the following systems:

- Kilogram-metre-hour.
- Kilogram-metre-day.
- Kilogram-metre-year.
- Kilogram-centimetre-second.
- Kilogram-centimetre-minute.
- 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 $[L \ T^{-2}]$, g in Equation 2.3.

• • •

Fluid density

1. **rho** Fluid density $[M \ L^{-3}]$, ρ in Equation 2.13.

• • •

Fluid viscosity

1. **visc** Fluid viscosity $[M \ L^{-1} \ T^{-1})]$, μ in Equation 2.13.

• • •

Fluid compressibility

1. **wcomp** Fluid compressibility [(LT^2) M^{-1}], α_w in Equation 2.15.

$$\bullet \; \bullet \; \bullet$$

Zero fluid compressibility
 Assigns a value of zero for fluid compressibility (i.e. incompressible).

$$\bullet \; \bullet \; \bullet$$

Fluid surface tension

1. **tensn** Fluid surface tension [M T^{-2}], χ in Equation 4.5.

$$\bullet \; \bullet \; \bullet$$

5.1.3   Pre-processor Considerations

5.1.3.1   Array Dimensioning

When performing Task 1, **grok** first checks for the existence of a file `array_sizes.default` in the directory where the *prefix*`.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

...etc...

tiles: flux function panels
        20
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 (to the screen and the *prefix*`o.eco` file) of the form:

```
*******************************************
*** DIMENSIONING ERROR, HALTING EXECUTION ***
```

```
**********************************************

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 `array_sizes.default` file 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. Re-compilation of the code is not necessary, since it uses Fortran 95 ALLOCATE statements to define array sizes at run-time.

HydroGeoSphere does not utilize the file `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 `array_sizes.default` file will be generated with default values.

5.2   Problem Identification

The first section of the *prefix*`.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** (*prefix*`o.eco`) and **HydroGeoSphere** (*prefix*`.lst`).

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

End
This instruction signals the end of the description, and control is then passed back to the pre-processor.

•••

5.3   Grid Generation

The next section of the *prefix*`.grok` file should consist of instructions for grid generation followed by an End instruction.

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.
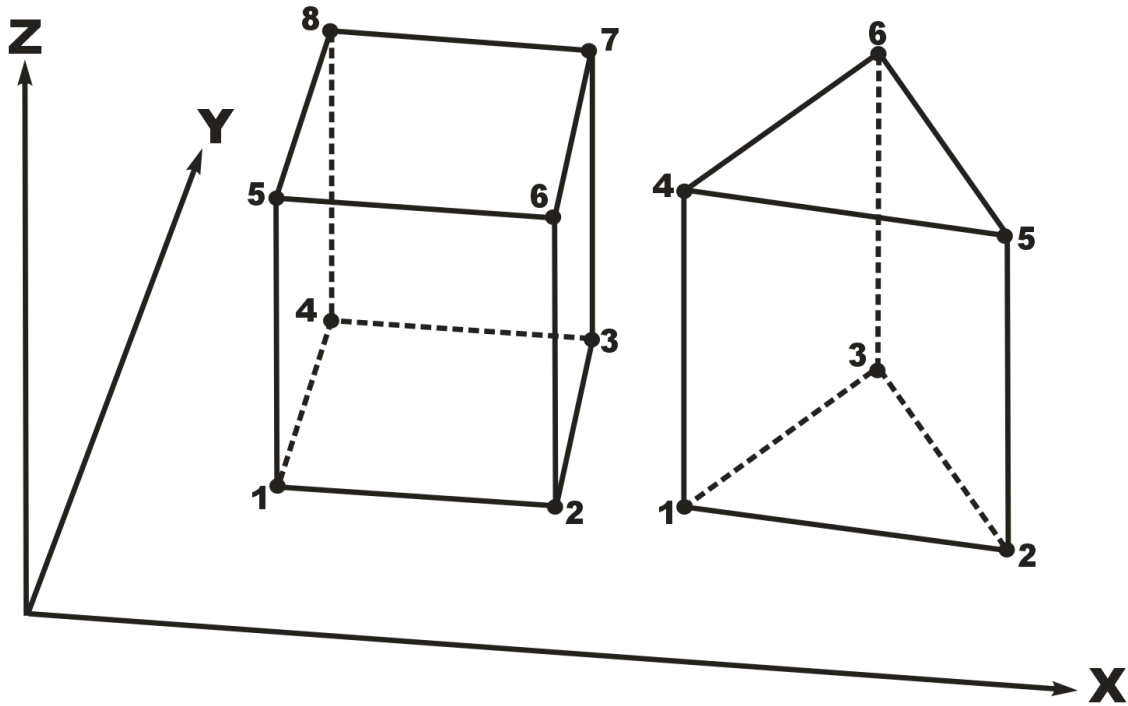
Figure 5.1: Element Types and Local Node Numbering Conventions.

There is also an option for subdividing hexahedral block elements into 4-node tetrahedral elements (see Section 5.3.6). 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

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

Generates a grid for a rectangular domain made up of uniform blocks. 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

1. **nx** Number of nodes in the x-direction x-coordinates of the **nx** nodes.

2. **ny** Number of nodes in the y-direction

3. **yi(i),i=1,ny** y-coordinates of the **ny** nodes.

4. **nz** Number of nodes in the z-direction

5. **zi(i),i=1,nz** z-coordinates of the **nz** nodes.

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

Interactive block 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. This is particularly useful for regions in which fine meshes are required, for example, near a discrete fracture or well.

Note that these instructions cannot be used in conjunction with the other grid generation instructions like Generate uniform/variable blocks/prisms.

Generate blocks interactive...End

Causes **grok** to begin reading a group of interactive block instructions until it encounters an End instruction. The group should contain of at least one instruction for each of the principal directions.

<div align="center">● ● ●</div>

The available instructions are:

Grade x

1. **x1, x2, dxstart, xfac, dxmax** Start and end x-coordinate, starting element size, element size multiplication factor and maximum element size.

Grid lines (i.e. elements) are generated along the x-axis from **x1** to **x2** which grade up in size from **dxstart** to **dxmax**. Element sizes are increased steadily by a factor of **xfac**.

<div align="center">● ● ●</div>

Grade y

As above but for the y-axis.

<div align="center">● ● ●</div>

Grade z

As above but for the z-axis.

<div align="center">● ● ●</div>

The instructions used to generate the mesh shown in Figure 5.2 are:

```
generate blocks interactive

grade x
 75.0      0.0    0.01    1.5  5.
grade x
 75.0    100.0    0.01    1.5  5.
grade x
125.0    100.0    0.01    1.5  5.
grade x
125.0    200.0    0.01    1.5  5.

grade y
100.0      0.0    0.01    1.5  5.
grade y
```
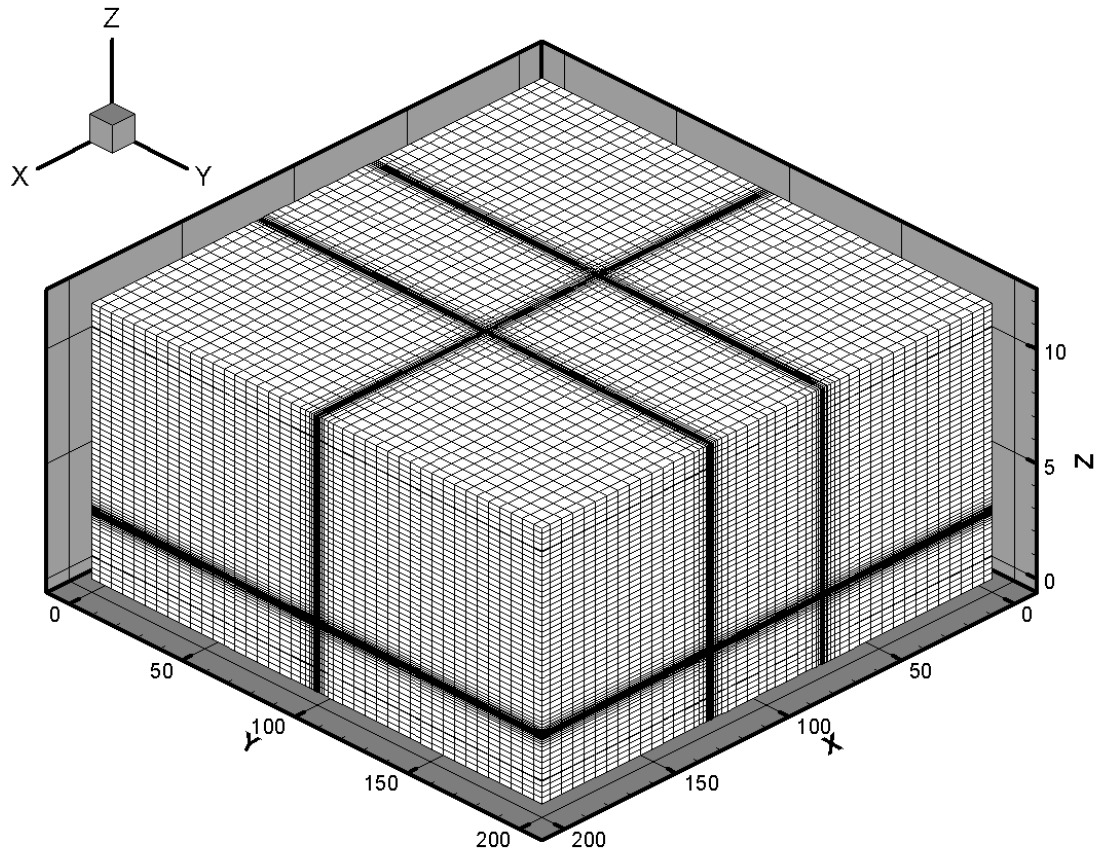
Figure 5.2: Example Grid which was Created Using Generate blocks interactive Instructions.

```
100.0    200.0    0.01    1.5  5.

grade z
   1.0     0.0    0.25    1.0  0.25
grade z
   3.0     1.0    0.01    1.3  0.25
grade z
   3.0    11.0    0.01    1.3  0.25
grade z
  11.0    12.0    0.25    1.0  0.25

end generate blocks interactive
```

5.3.3   3-D Random Fracture Generator for Block Grids

The following command can be used to generate a 3-D random fracture network in an orthogonal domain (i.e. composed of 8-node block elements). Fractures with random locations, lengths and apertures can be generated.

Rfgen driver

1. **rfgfile** Name of the file which contains the random fracture grid and fracture generation information.

The structure of the file is discussed below.

• • •

Grid information

1. **x1, x2** x-range of the domain.

2. **y1, y2** y-range of the domain.

3. **z1, z2** z-range of the domain.

4. **botfracbnd(1)** Elevation of lowest extent of a fracture. No fractures will be generated below this elevation.

5. **nwell** Number of wells. Read the following **nwell** times:

   (a) **xwell, ywell** xy-coordinates of the well. $x-$ and $y-$gridlines will be generated at this point.

6. **xsource1, xsource2** x-coordinates of the source. $x-$gridlines will be generated at these points.

7. **ysource1, ysource2** y-coordinates of the source. As above but for the y-direction.

8. **zsource1, zsource2** z-coordinates of the source. As above but for the z-direction.

9. **mingrspacx,mingrspacy,mingrspacz** Minimum grid spacing in the $x-$, $y-$ and $z-$directions respectively. For example, a **mingrspacx** value of 1.0 would ensure that no gridlines are more than 1.0 length units apart along the x-axis.

10. **fixed_grid** This is a logical variable which controls whether grid lines are generated randomly or according to fixed spacing input parameters. If true read the following:

   (a) **fixed_spac** This is a logical variable which controls whether uniform or variable grid line spacing is applied. If true read the following:

      i. **fixgrspacx,fixgrspacy,fixgrspacz** Fixed spacing values in the $x-$, $y-$ and $z-$directions respectively.

If false read the following:

   i. **nx** Number of nodes in the x-direction
   ii. **xi(i),i=1,nx** x-coordinates of the **nx** nodes.
  iii. **ny** Number of nodes in the y-direction
  iv. **yi(i),i=1,ny** y-coordinates of the **ny** nodes.
   v. **nz** Number of nodes in the z-direction
  vi. **zi(i),i=1,nz** z-coordinates of the **nz** nodes.

This instruction should be placed at the start of the file and should not appear more than once.

•••

Fracture information

1. **seed** Seed for the random number generator. If this number is changed, a new random number sequence is produced, which in turn causes new realizations of fracture location, length and aperture to be generated.

2. **xmeanfreq** Mean fracture frequency in x-direction.

3. **ymeanfreq** As above but in the y-direction.

4. **zmeanfreq** As above but in the z-direction.

5. **zeta** Aperture decay constant. Aperture size can be made to decrease with increasing depth. Set to zero for no decay.

6. **lnsbetween** Minimum number of grid lines between fractures.

7. **cap** cap on the number of times to attempt generating a fracture.

This instruction should follow the Grid information instruction and should not appear more than once.

•••

Fracture location distribution x-axis

1. **type** An integer value indicating the type of function to use to generate the variable fracture locations in the $x-$direction. Acceptable values are:

   1   Uniform.
   2   Normal.
   3   Exponential.

2. **var1, var2** Distribution parameters which control the function.

For a uniform distribution **var1** is the minimum and **var2** is the maximum.

For a normal distribution **var1** is the mean and **var2** is the variance.

For an exponential distribution **var1** is the mean and **var2** is the standard deviation.

The following instructions use the same input data structure except they are applied to the y and z directions:

> Fracture location distribution y-axis
> Fracture location distribution z-axis

The following instructions use the same input data structure to generate fracture lengths in the 3 principal directions:

> Fracture length distribution x-axis
> Fracture length distribution y-axis
> Fracture length distribution z-axis

The following instructions use the same input data structure to generate fracture apertures in the 3 principal orientations:

> Xy fracture aperture distribution
> Xz fracture aperture distribution
> Yz fracture aperture distribution

<div align="center">• • •</div>

The remaining commands are optional but should not be used more than once:

Vertical fracture from top

1. **vertical_frac_top** This is a logical variable which, if true, ensures that all vertical fractures start from the top of the domain.

<div align="center">• • •</div>

Zone fractures how

1. **zone_rfgen_fracs** Controls how fracture zone numbers are assigned. Acceptable values are:

   1   Assign zone numbers by fracture.
   2   Assign zone numbers by orientation.

If zoned by orientation, horizontal fracture are assigned to zone 1, vertical fractures parallel to the xy-axis are in zone 2 and vertical fractures parallel to the xz-axis are in zone 3.

<div align="center">• • •</div>

End

This instruction signals the end of the 3-D random fracture generator input, and control is then passed back to the pre-processor.

● ● ●

Once the 3-D grid is generated, it is possible to change the random fracture apertures to zoned fracture apertures by following the procedures outlined in Section 5.8.1.6.

5.3.4 2-D Random Fracture Generator

The 2D Random Fracture Generator can be used to generate random fracture networks in two dimensions, currently only in the xz-plane. Nevertheless, in the y-direction, more than one block can be used.

Begin 2D random fractures...End

Causes **grok** to begin reading instructions that describe the generation of 2-D random fractures until it encounters an End instruction.

● ● ●

The folowing optional instructions can be used to modify the default behaviour of the fracture generator:

Number of random fractures

1. **n_rfractures** Number of random fractures to generate.

By default, the 2-D random fracture network will consist of 80 fractures.

● ● ●

Use constant seed

1. **the_seed** Seed for the 2-D Random Fracture Generator.

Causes the 2-D Random Fracture Generator to use a constant seed **the_seed** to produce the same random fracture network each time **grok** is run.

By default, the 2-D Random Fracture Generator is seeded with a time-dependent value, based on the current system time. In that case, it produces a <u>different</u> fracture network each time **grok** is run.

In either case, the seed value is written to the *prefix*o.eco file and can be used to generate the same random fracture network many times.

● ● ●

Generate orientation distribution

Table 5.1: Default Values for 2-D Random Fracture Orientation.

Parameter	Value	Unit
Number of orientation classes	13	
Middle of the smallest orientation class	30	degrees
Middle of the largest orientation class	150	degrees
Standard deviation of both classes σ	15	degrees
Mean of the first Gaussian distribution μ_1	60	degrees
Mean of the second Gaussian distribution μ_2	120	degrees

1. **or_n_classes** Number of orientation classes.

2. **or_first_class_middle** Middle of the smallest orientation class.

3. **or_last_class_middle** Middle of the largest orientation class.

4. **or_sigma** Standard deviation σ of both Gaussian distributions.

5. **or_my1** Mean μ_1 of the first Gaussian distribution.

6. **or_my2** Mean μ_2 of the second Gaussian distribution.

Causes **grok** to read the parameters that are used to define the distribution of fracture orientation, which follows a double-peaked Gaussian distribution according to:

$$P(\alpha) = P_1(\alpha) + P_2(\alpha) \tag{5.1}$$

where the normal distribution $P(x)$ for a variable x with mean μ and variance σ^2 has the probability function

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5.2}$$

for the domain $x \in (-\infty, \infty)$.

By default, the values given in Table 5.1 are used to define these relationships.

• • •

Generate aperture distribution

1. **ap_n_classes** Number of aperture classes.

2. **ap_first_class_middle** Middle of the smallest aperture class.

3. **ap_last_class_middle** Middle of the largest aperture class.

4. **ap_lambda** λ of the exponential aperture distribution.

Causes **grok** to read the parameters that are used to define the distribution of fracture aperture, which follows an exponential distribution, according to:

$$P(x) = P_o \cdot e^{-\lambda x} \tag{5.3}$$

Table 5.2: Default Values for 2-D Random Fracture Aperture.

Parameter	Value	Unit
Number of aperture classes	10	
Middle of the smallest aperture class	50	microns
Middle of the largest aperture class	300	microns
λ of the exponential aperture distribution	9000	

Table 5.3: Default Values for 2-D Random Fracture Length, Log-normal Distribution.

Parameter	Value	Unit
Number of length classes	10	
Middle of the smallest length class	$0.1 \cdot \min(L_x, L_z)$ [†]	m [‡]
Middle of the largest length class	$\min(L_x, L_z)$	m
μ of the log-normal distribution	2.9	
σ of the log-normal length distribution	0.45	

[†] The symbols L_x and L_z denote the length of the simulation domain the the x and z directions respectively.

[‡] Although metre units of length are shown in this table they may be defined differently by the user as outlined in Section 5.1.2.

Note that for a high value of λ, the exponential distribution becomes steeper and small apertures are more numerous, whereas a small value of λ favours larger apertures.

By default, the values given in Table 5.2 are used to define these relationships.

• • •

Generate log-normal length distribution

1. **le_n_classes** Number of length classes.

2. **le_first_class_middle** Middle of the smallest length class.

3. **le_last_class_middle** Middle of the largest length class.

4. **lognormal_m** μ of the log-normal distribution.

5. **lognormal_s** σ of the log-normal length distribution.

Causes **grok** to read the parameters that are used to define the distribution of fracture length, which follows a log-normal distribution according to:

$$P(x) = \frac{1}{\sigma x \sqrt{2\pi}} \, e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \tag{5.4}$$

By default, the values given in Table 5.3 are used to define these relationships.

It should be noted that the log-normal distribution $\Lambda(\mu, \sigma^2)$ results from the curve $\Lambda(0, \sigma^2)$ by:

Table 5.4: Default Values for 2-D Random Fracture Length, Exponential Distribution.

Parameter	Value	Unit
Number of length classes	10	
Middle of the smallest length class	$0.1 \cdot \min(L_x, L_z)$ [†]	m [‡]
Middle of the largest length class	$\min(L_x, L_z)$	m
λ of the exponential length distribution	0.05	

[†] The symbols L_x and L_z denote the length of the simulation domain in the x and z directions respectively.

[‡] Although metre units of length are shown in this table they may be defined differently by the user as outlined in Section 5.1.2.

- Stretching of e^{μ} in the x-direction.

- Stretching of $e^{-\mu}$ in the z-direction.

Thus, larger values for μ will move the peak to the right. Altering the standard deviation σ will have an impact on the scattering of the distribution where a small σ leads to less scattering and a sharper peak.

• • •

Exponential length distribution

Causes **grok** to use an exponential distribution of the fracture trace, as opposed to the default log-normal distibution.

• • •

Generate exponential length distribution

1. **le_n_classes** Number of length classes.

2. **le_first_class_middle** Middle of the smallest length class.

3. **le_last_class_middle** Middle of the largest length class.

4. **le_lambda** λ of the exponential length distribution.

Used in conjunction with the Exponential length distribution instruction, this causes **grok** to read the parameters that are used to define the distribution of fracture length, which follows an exponential distribution according to:

$$P(x) = P_o \cdot e^{-\lambda x} \tag{5.5}$$

Note that for a high value for λ, the exponential distribution becomes steeper and short fractures are more numerous whereas a small λ favors longer fractures.

By default, the values given in Table 5.4 are used to define these relationships.

• • •

Output random apertures

Writes the generated aperture distribution data and individual fracture apertures to the output file *prefix*o.rfrac.apertures.

<p align="center">● ● ●</p>

Output random lengths

Writes the generated length distribution data and individual fracture lengths to the output file *prefix*o.rfrac.lengths.

<p align="center">● ● ●</p>

Output random orientations

Writes the generated orientation distribution data and individual fracture orientations to the output file *prefix*o.rfrac.orientations.

<p align="center">● ● ●</p>

Output random fractures

Writes fracture zone aperture, conductivity and location data to the output file *prefix*o.rfrac.fractures.

<p align="center">● ● ●</p>

Figure 5.3 gives an overview of the default distributions which the 2D Random Fracture Generator employs. The orientation distribution (Figure 5.3a) is based on the assumption that tectonic stress results in the creation of two fracture families as depicted in Figure 5.3b. However, upon assigning identical values for μ_1 and μ_2, the distribution collapses to a one-peak distribution. The default distribution for the aperture (Figure 5.3c) is exponential and can be modified by the user. By default, the fracture traces are distributed log-normally (Figure 5.3d), which can be changed to exponential. Note that the fracture trace distribution depends on the domain dimensions. Here, a block has been used with $L_x = 100m$, $L_y = 1m$ and $L_z = 50m$.

The following instructions were used to generate the irregular fracture network shown in Figure 5.4. Note the dominance of the two orientations 80^0 and 135^0.

```
!_____ grid definition

generate uniform blocks
100.0 200
1.0 1
50.0 100

adapt grid to fractures
3
```
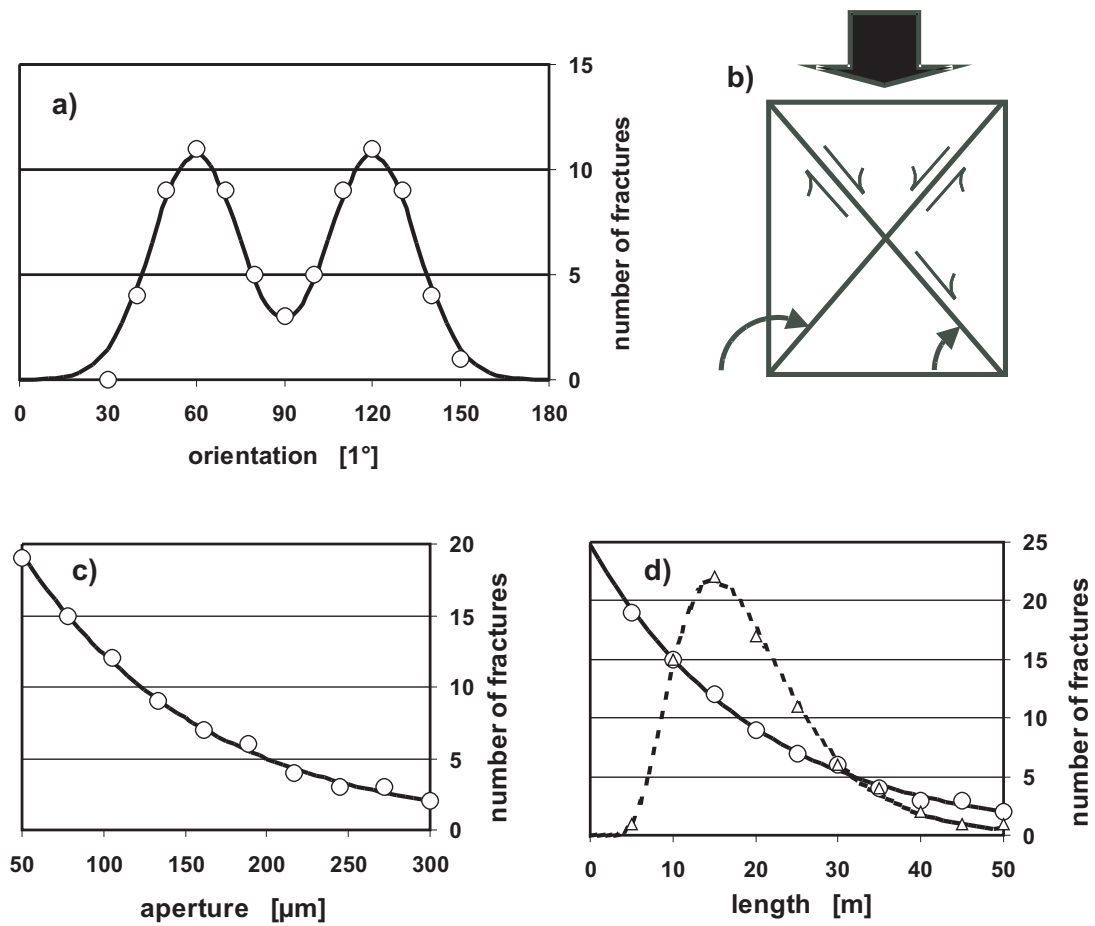
Figure 5.3: Default Random Fracture Distribution

```
end

...etc...

!_____ fracture media properties

use domain type
fracture

properties file
eval.fprops

begin random fractures

use constant seed
0.5

number of random fractures
70

exponential length distribution

generate orientation distribution
10
60.
150.
10.
80.
135.

output random apertures
output random lengths
output random orientations
output random fractures

end

read properties
fracture
```
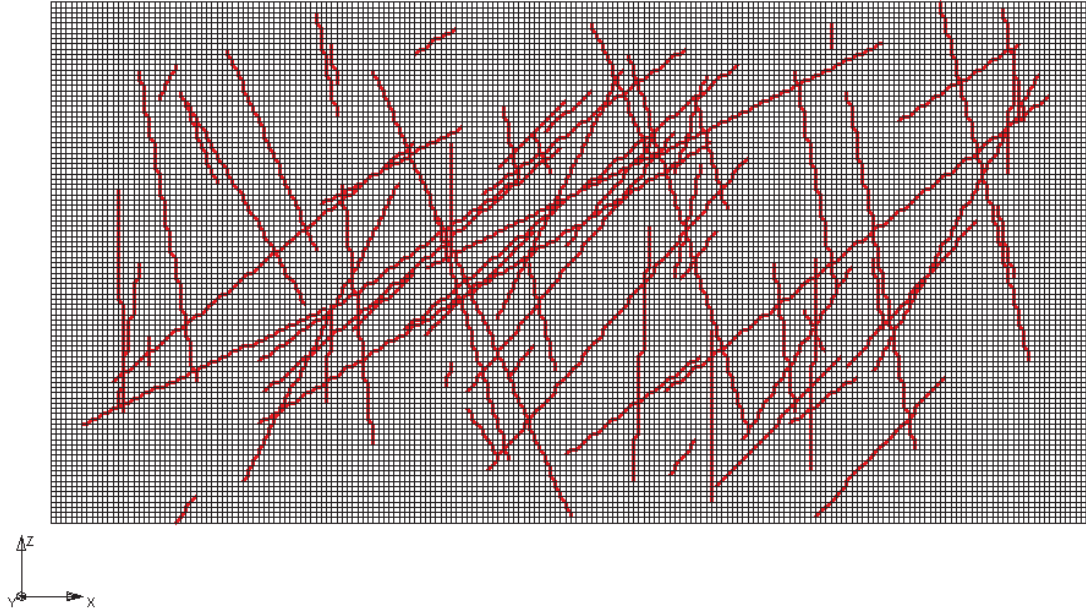
Figure 5.4: Example for an Irregular Fracture Network

5.3.5   Interactive 3-D Mesh Generator

Irregular grids can be generated by supplying nodal coordinates, element incidences and element zones for a 2-D 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 3-D mesh composed of 6-node prisms (from triangles) or 8-node hexahedra (from quadrilaterals).

5.3.5.1   Defining a 2-D Mesh

The following instructions can be used to obtain 2-D slice data.

Generate uniform rectangles

1. **xl, nbx** Length and number of rectangles in the x-direction

2. **yl, nby** Length and number of rectangles in the $y$direction

Generates a 2-D grid for a rectangular domain made up of uniform rectangles. Each rectangular element will be assigned a default zone number of 1. It is identical to the generate uniform blocks instruction except that we drop the z-axis parameters.

● ● ●

Generate variable rectangles

1. **nx** Number of nodes in the x-direction

2. **xi(i),i=1,nx** x-coordinates of the **nx** nodes.

3. **ny** Number of nodes in the y-direction

4. **yi(i),i=1,ny** y-coordinates of the **ny** nodes.

Generates a 2-D grid for a rectangular domain made up of variably-sized rectangles. Each rectangular element will be assigned a zone number of 1. It is almost identical to the **generate variable blocks** instruction except that we drop the z-axis parameters.

• • •

Generate rectangles interactive

This instruction works in exactly the same way as the **Generate blocks interactive** instruction described in Section 5.3.2, except that input is limited to the $x-$ and y-directions and a 2-D mesh of 4-node rectangular elements is generated.

• • •

Read gms 2d grid

1. **gmsfile** Name of the file which contains the 2-D slice data.

Reads a file which contains data defining a 2-D slice. The format of this file is described in detail in Section F.1 and 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.

• • •

Read gb 2d grid

1. **prefix** Prefix of the GRID BUILDER files which contain the node coordinates, element incidences and element zone numbers for the 2-D triangular mesh. This is a string variable.

Reads the files which contain data defining a 2-D slice composed of 3-node triangular elements. These files are described in detail in Section G.1 and are compatible with output generated by the GRID BUILDER program.

• • •

Read fractran 2d grid

1. **prefix** Prefix of the FRACTRAN files which contain the node coordinates, element incidences and element zone numbers for the 2-D rectangular element mesh. This is a string variable.

Reads the files which contain data defining a 2-D slice composed of 4-node rectangular elements. These files are compatible with output generated by the FRACTRAN program.

• • • For a 2-D slice made of 4-node rectangular elements, the following instructions can be used to remove elements:

Remove rectangles with shapefile

1. **arcview_prefix** Prefix of the ARCVIEW shape file.

2. **unproject_file** If .true, this logical switch causes **grok** to read grid unprojection data as described in Section 5.3.10 and to apply it to the data read from the ARCVIEW shapefile.

3. **project_file** If .true, this logical switch causes **grok** to read grid projection data as described in Section 5.3.10 and to apply it to the data read from the ARCVIEW shapefile.

4. **outside** If .true, elements located outside the area defined in the ARCVIEW shapefile are removed. If .false, elements located inside the area are removed.

• • •

Remove rectangles with blanking file
As above except a blanking file in surfer format is used instead of an ARCVIEW shape file.

• • •

Raster to scl

1. **arcview_file_name** Name of the Arcview ASCII file.

2. **bandwidth** Cell bandwidth used for averaging.

Reads an ARCVIEW ascii file and interpolates a value for each 2-D mesh node. The results are written to a GMS formatted scalar file called `output.scl`.

• • •

Raster to nprop

1. **arcview_file_name** Name of the Arcview ASCII file.

2. **bandwidth** Cell bandwidth used for averaging.

Reads an ARCVIEW ascii file and interpolates a value for each 2-D mesh node. The results are written to a Grid Builder compatible binary file called `raster2nprop.output.nprop`.

<center>• • •</center>

Raster to element

1. **arcview_file_name** Name of the Arcview ASCII file.

2. **Statistic** Statistic to be used to interpolate values. Acceptable values for the variable **statistic** are:

   > max count
   > nearest

   If variable **statistic** is set to `max count` read the following:

   (a) **bandwidth** Cell bandwidth used for averaging.

Reads an ARCVIEW ascii file and interpolates a value for each 2-D mesh element. The results are written as a list of element number and value to a file called `output.el`.

<center>• • •</center>

5.3.5.2   3-D Mesh Generation

Once you have a 2-D slice, you have the option of exiting the grid definition procedure, which will cause **grok** to automatically generate a unit thickness 3-D grid. It does this by duplicating the 2-D slice and constructing the appropriate 6-node prism or 8-node hexahedral element incidences and assigning a unit element length perpendicular to the slice. The element zone numbers for the slice are used to assign default zone numbers for each element. Such a grid could be used to simulate 2-D cross-sectional problems.

More often, you will want to generate a 3-D layered grid, perhaps with topography defined by a DEM (Digital Elevation Model) and/or uneven layer contacts based on the observed hydrostratigraphy.

To do so you should start by issuing the following instruction:

Generate layers interactive...End

Causes **grok** to begin reading a group of 3-D grid generation instructions until it encounters an End instruction.

<center>• • •</center>

The basic procedure is to build up the 3-D mesh by defining the base, then adding layers one at a time from the base to ground surface.

By default, the domain will contain a single layer, one element high with a base elevation of zero and a top elevation of 1, and the element zone numbering scheme from the 2-D slice will be used to assign the 3D mesh element zone numbers. Instructions that change the default behaviour are described below:

These commands are optional but should not be used more than once:

Zone by layer
Causes **grok** to assign the 3D mesh layer number to the element zone number.

By default, the element zone numbering scheme from the 2-D slice is used to assign the 3D mesh element zone number.

• • •

Base elevation…End
Causes **grok** to begin reading a group of base elevation instructions until it encounters an End instruction. Available instructions are described in Section 5.3.5.4.

By default, the base elevation of the domain will be set to zero.

• • •

5.3.5.3   Adding a New Layer

For each layer in the domain, you should include a single instance of the following instruction:

New layer…End
Causes **grok** to begin reading a group of new layer instructions until it encounters an End instruction.

By default, the top elevation of the layer will be set to zero by assigning all nodes in the lowermost 2-D slice to have a z-coordinate value of zero. You can change the layer top elevation using the instructions described in Section 5.3.5.4.

• • •

The following instructions are all optional:

Layer name
   1. **layer_name** Layer name.

Changes the layer name, which defaults to **Layer n**, where n is the current layer number.

• • •

Minimum layer thickness

1. **z_added** Minumum thickness value[L].

This instruction causes **grok** to enforce a minimum thickness constraint for the current layer. At nodes where the computed layer top elevation is less than or equal to the current base elevation, **z_added** will be added to the current base elevation to get the top elevation.

If this constraint is not enforced, **grok** will stop and issue a warning message if the computed top elevation is less than or equal to the current base elevation.

$$\bullet \ \bullet \ \bullet$$

By default, a new layer will not be subdivided vertically unless one of the following two instructions is issued:

Uniform sublayering

1. **nsublayer** Number of sublayers.

This instruction divides the layer vertically into **nsublayer** elements, which will each have the same height, equal to the top elevation minus the current base elevation divided by **nsublayer**.

$$\bullet \ \bullet \ \bullet$$

Proportional sublayering

1. **nsublayer** Number of proportional sublayers.

2. **sub_thick(i),i=1,nsublayer** Proportional thiknesses in order from top to bottom.

This instruction can be used if you want to refine the mesh vertically, for example in the subsurface near the surface flow domain or a fracture.

It is important to understand that the variable **sub_thick** is not a true thickness, but is instead a relative thickness, which is used along with the layer thickness to determine the element heights in the current column.

For example, these instructions would subdivide the current layer vertically into three elements, between the current base and top elevation, with element height proportions of .1, 1 and 10 from top to bottom:

```
Proportional sublayering
   3
   0.1
   1.0
   10.0
end
```

● ● ●

Offset top

1. **value** Thickness value (L) by which to offset the layer top elevation.

This instruction causes the elevation of the top layer to be offset vertically by the given value. This can be used to create a surface a given distance below another surface.

For example, these instructions create a layer with a top elevation 1 metre below the elevation defined in the raster file `gs.asc`:

```
new layer
   Uniform sublayering
   10

   elevation from raster file
   gs.asc

   offset top
   -1.0

end

new layer

   Uniform sublayering
   3

   elevation from raster file
   gs.asc
end
```

● ● ●

5.3.5.4   Elevation Instructions

These instructions are used to define 3-D mesh base elevations and new layer top elevations.

Elevation constant

1. **elev** Elevation value [L].

• • •

Elevation from gms file

1. **basefile** Name of the data file containing the elevation values for each node in the 2-D grid. This is a string variable. The file should be formatted as outlined in Section F.2.

• • •

Elevation from gb file

1. **basefile** Name of the data file containing the base elevation values for each node in the 2-D grid. This is a string variable. The file should be formatted as outlined in Section G.2.

• • •

Elevation from raster file

1. **rasterfile** Name of the raster file containing the base elevation values. This is a string variable. The file should be formatted as outlined in Section H.

• • •

Elevation from bilinear function in xy

1. **xfrom, xto, yfrom, yto** x and y ranges.

2. **a1,a2,a3,a4,a5** Constants for bilinear function.

For nodes falling within the given x and y range, the z-coordinate is computed according to the following function:

$$z = a1 + a2(x - \mathbf{xfrom}) + a3 * (x - \mathbf{xfrom})^2 + a4(y - \mathbf{yfrom}) + a5(y - \mathbf{yfrom})^2$$

• • •

Elevation from sine function in xy

1. **xfrom, xto, yfrom, yto** x and y ranges.

2. **zz0** Elevation at **xfrom, yfrom**.

3. **num_sw_x,amplitude_x,slope_x** Number of sine wave cycles, sine wave amplitude and surface slope in the $x-$direction.
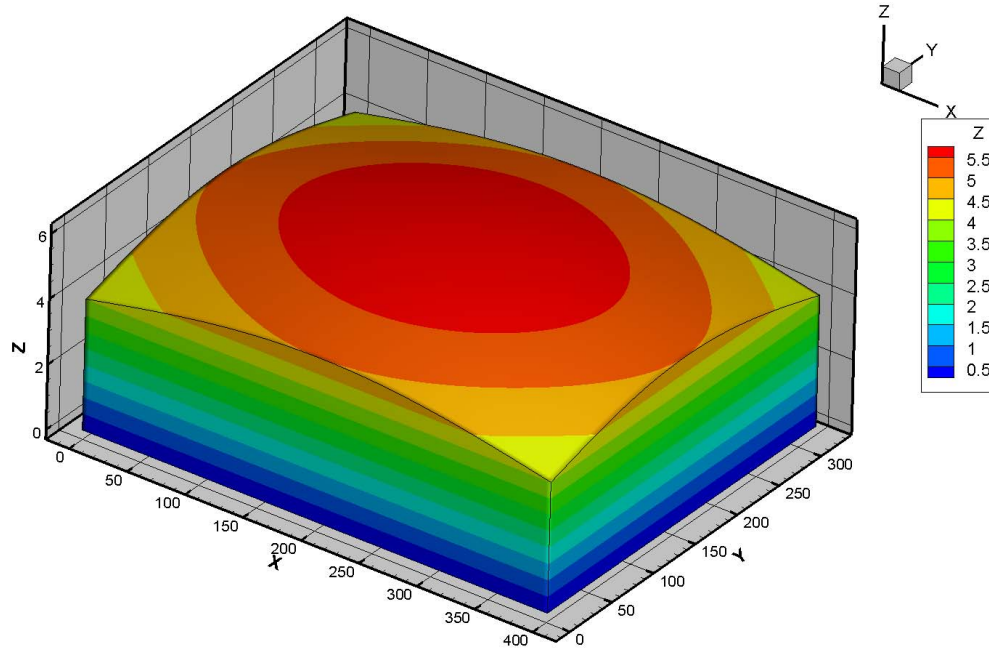
Figure 5.5: Surface elevation example using Elevation from sine function in xy.

4. **num_sw_y,amplitude_y,slope_y** As above but in the $y-$direction.

For nodes falling within the given x and y range, the z-coordinate is computed according to the following function:

$$z = \textbf{zz0} + \textbf{amplitude\_x}(1 + \sin(f(x))) + \textbf{slope\_x}(x - \textbf{xfrom})$$
$$+\textbf{amplitude\_y}(1 + \sin(f(y))) + \textbf{slope\_y}(y - \textbf{yfrom})$$

where:

$$f(x) = (x - \textbf{xfrom})/(\textbf{xto} - \textbf{xfrom}) * 2\pi * \ \textbf{num\_sw\_x}$$
$$f(y) = (y - \textbf{yfrom})/(\textbf{yto} - \textbf{yfrom}) * 2\pi * \ \textbf{num\_sw\_y}$$

• • •

Figure 5.5 shows that the number of cycles of the sine wave can be a fraction, the sine function rises from a value of **zz0** at **(xfrom, yfrom)** as x and y values increase, and that where the peaks coincide, the maximum elevation, in this case 6, is the sum of **zz0** + **amplitude_x** + **amplitude_y**. It was produced using the following input:

```
elevation from sine function in xy
0.0 400.0 0.0 320.0
4.
```
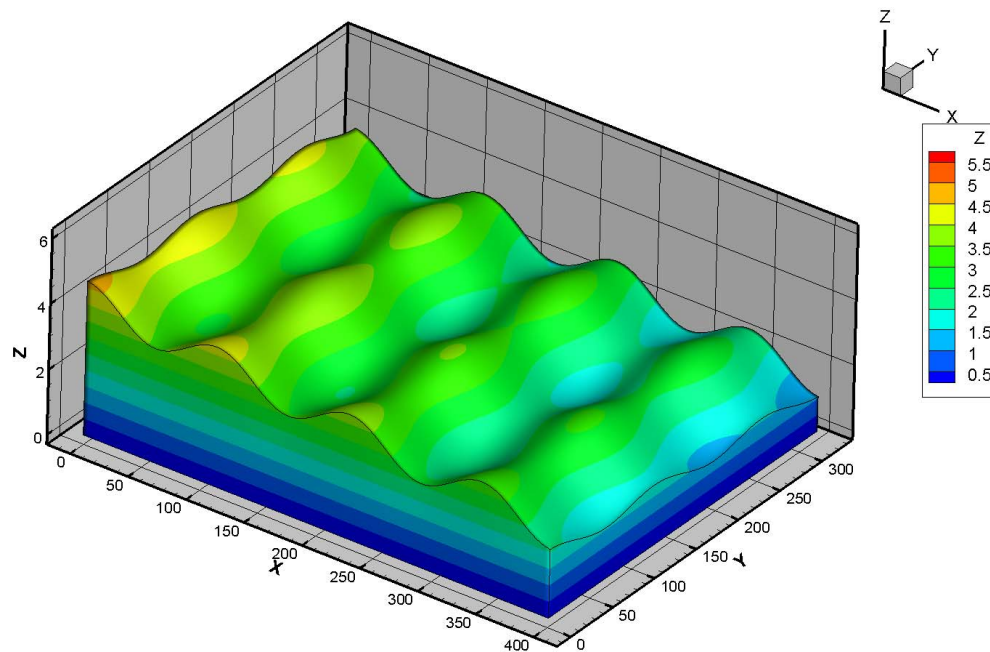
Figure 5.6: Surface elevation example using Elevation from cosine function in xy.

```
0.5 1. 0.0
0.5 1. 0.0
```

Elevation from cosine function in xy

As above but uses the cosine function instead of the sine function.

• • •

Figure 5.6 shows the use of multiple cycles, different degrees of slope in x and y and that the cosine function starts with a peak value at (**xfrom, yfrom**) and decreases as x and y values increase. It was produced using the following input:

```
elevation from cosine function in xy
0.0 400.0 0.0 320.0
4.
3.5 .5 -0.004
2.5 .2 -0.002
```

Elevation from xz pairs

1. **xval, zval** xz pair 1.

2. **xval, zval** xz pair 2.

3. **...etc...**

4. **xval, zval** xz pair n.

5. **end** Signals end of list.

Listed xz coordinate pairs are read until an End instruction is encountered. They should be given in order from smallest to largest x. For each node in the 2-D grid, the x coordinate of the node is used to determine it's position in the list, and a z coordinate is then interpolated from the neighbouring xz pairs.

• • •

5.3.6   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 **HydroGeoSphere** 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.

5.3.7   Axisymmetric Flow

Axisymmetric coordinates

This instruction is used for simulating radial flow to a well. *It should only be applied to 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).

• • •

5.3.8   Reading an Existing 3-D 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 *prefix*.`grok` file will be read in.

These instructions show how to set up the grid generation section of the *prefix*.`grok` file to use the Read 3d grid instruction:

```
!  Generate the grid for first run
   !  skip on

   read gb 2d grid
   laurel

   generate layers from gb 2d grid
   .true.                           !  zone by layer?
   .false.                          !  base elevation constant?
   laurel.nprop.Bedrock
   1                                !  layer
   Whole domain
   20                               !  sublayer
   .false.                          !  top elevation constant?
   laurel.nprop.Topography

   write faces and segments
   !skip off

   skip on
   !  Read previously defined grid for subsequent runs
   read 3D grid
   skip off

end 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 instructions later in the *prefix*.`grok` file.

On subsequent runs, one can skip over the grid generation commands and use the read 3D grid instruction instead.

● ● ●

5.3.9   Manipulating the 3D Grid

Tilt grid x
 Tilts a grid by 90°around the x-axis. This is especially suitable in conjunction with the instruction read slice which reads a two-dimensional grid (parallel to the xy-plane) in gms format.

<div align="center">● ● ●</div>

Tilt grid y
As above but for the y-axis.

<div align="center">● ● ●</div>

Adapt grid to fractures

1. **adapt_g2f_mode** An integer value indicating how the grid is adapted to inclined fractures.

If block elements are used, two inclined fractures may intersect in the middle of an element instead of on a grid node, so the fractures will not be connected unless additional nodes are specified.

Acceptable values for the variable **adapt_g2f_mode** and the actions taken in each case are:

0	No action is taken.
1	New grid lines are added.
2	The block element is substituted by four prisms.
3	Inclined faces are not selected in the block element where the problem occurs.

The default value is 1.

<div align="center">● ● ●</div>

5.3.10   Grid Projection

Figure 5.7 illustrates the degree of distortion that can occur (which is especially severe at the poles) when a spherical geographic coordinate system, such as longitude and latitude, is mapped directly into a planar 2-D coordinate system. The following command can be used to correct the distortion which occurs when modelling large continental scale problems.

Project grid

1. **projt** The projection type to be applied. Currently, the only valid input for **projt** are `albers equal-area` or `lambert azimuthal equal-area`, which apply the Albers

conical or Lambert azimuthal equal-area projections respectively. For the Albers projection, the following additional data are required:

(a) **lon0, lat0** The longitude and latitude for the origin of the projection.

(b) **lat1, lat2** The latitude of the standard parallels.

(c) **datum** The datum used for the projection. Acceptable values for the variable **datum** are:

    1   Sphere.
    2   Ellipsoid (Clarke 1866).

For the Lambert projection, the following additional data are required:

(a) **lon0, lat0** The longitude and latitude for the origin of the projection.

(b) **datum** The datum used for the projection. Acceptable values for the variable **datum** are:

    1   Sphere.

Transforms the current mesh from a spherical system into a planar 2-D coordinate system. It must be inserted just before the End grid generation command.

The following example uses the Project Grid instruction:

```
generate blocks interactive
    grade x
    -150 -50 5 1 5
    grade y
    40 85 5 1 5
    grade z
    0 10 10 1 10
    end

    project grid
    albers equal-area
    -100 60                            !  projection origin
    55 70                              !  standard parallels
    1                                  !  1=sphere, 2=ellipsoid
end
```

The grid produced by the Generate blocks interactive instruction is shown on Figure 5.7 and once projected, the grid looks like the one on Figure 5.8.

• • •
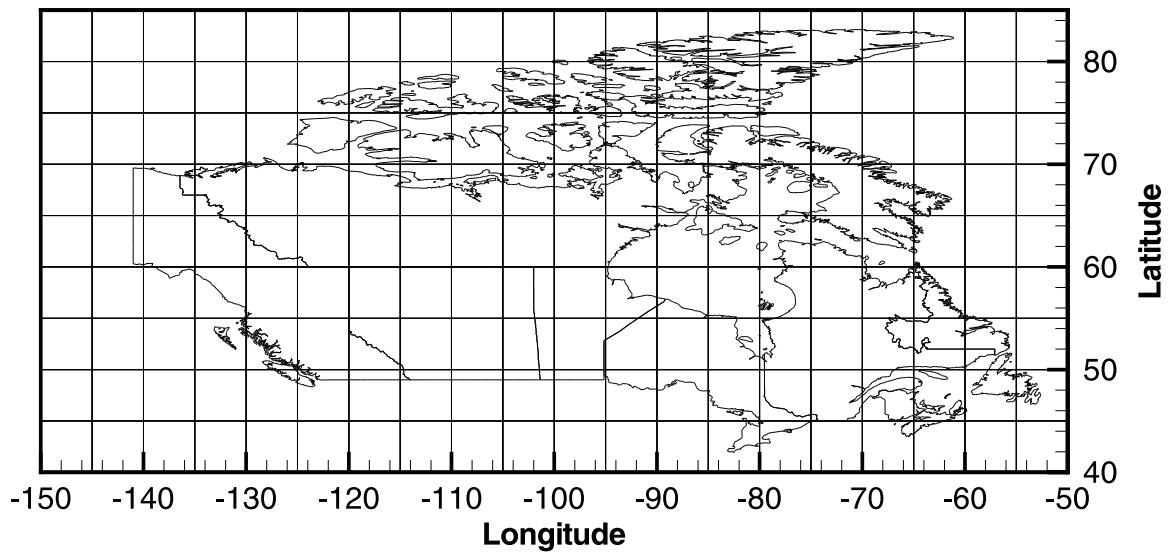
Figure 5.7: Mesh in Geographic Coordinates (lat/long). The map of Canada is shown as a reference.
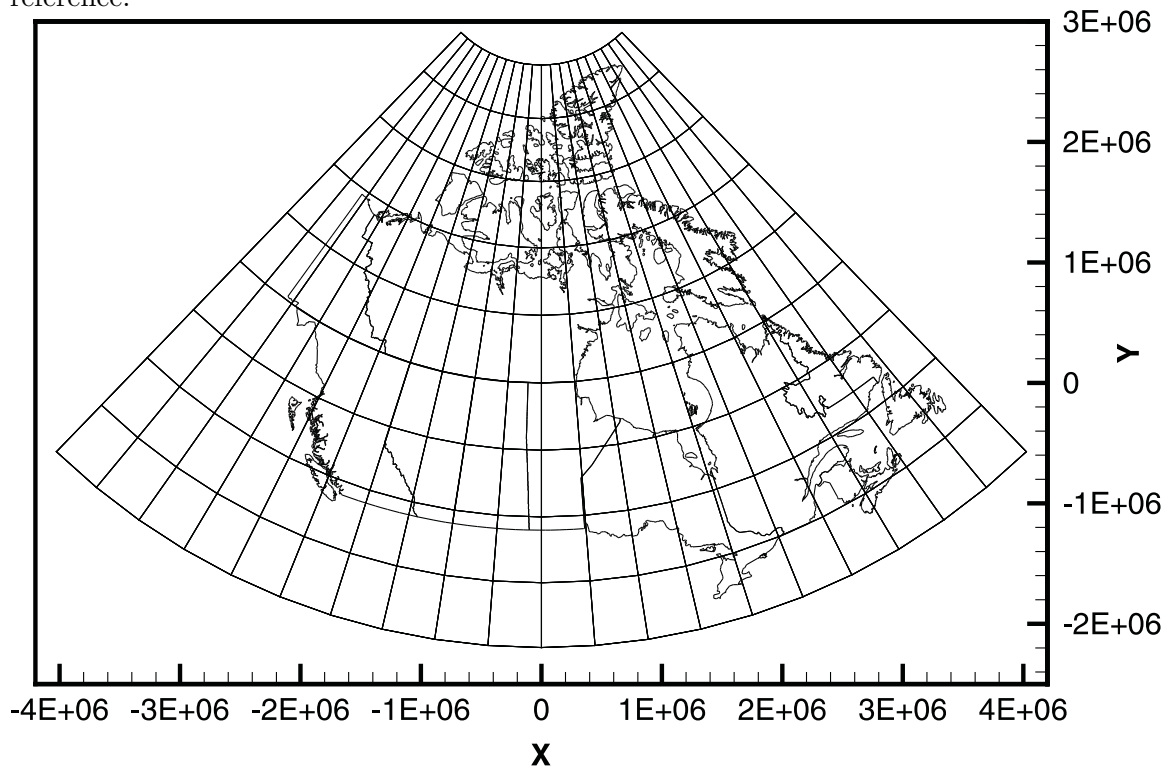
Figure 5.8: Mesh using Albers Equal-area Projection. The map of Canada now shows much less distortion but the elements are deformed.

Project 2d grid

The input data for this instruction is identical to that required for the Project grid instruction described above.

Transforms the current mesh from a planar 2-D coordinate system into a spherical system. It must be inserted just before the End grid generation command.

• • •

Unproject 2d grid

The input data for this instruction is identical to that required for the Project 2d grid instruction described above.

Transforms the current mesh from a spherical coordinate system into a planar 2-D system. It must be inserted just before the End grid generation command.

• • •

5.3.11 Ending Grid Generation

End

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 2-D slice data but did not specify layer information using for example, the generate layers interactive instruction, the pre-processor would generate a default 3-D system by duplicating the 2-D slice to form a single layer of unit-thickness elements.

• • •

5.4 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 3-D 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 *prefix*.grok file would initially turn off all chosen nodes (i.e. 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 $x = 0.0$ (i.e. 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 chosen nodes 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 instruction 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.

5.4.1 Selecting Nodes

We can use the following instructions to alter the set of chosen nodes.

Clear chosen nodes

All nodes in the domain will be flagged as *not* chosen. This is recommended if you are unsure of which nodes are chosen due to previously issued instructions.

● ● ●

Choose nodes all

All nodes in the domain will be chosen. 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

1. **x1, y1, z1** *xyz*-coordinate.

The node closest to the given coordinate will be chosen.

● ● ●

Choose node number

1. **i** Node number.

The node with this number will be chosen. You should use this instruction with caution since node numbering will change if the grid structure changes.

● ● ●

Choose nodes x plane

1. **x1** *x*-coordinate of the plane.

2. **ptol** Distance from the plane.

Nodes within distance **ptol** of the plane defined by the equation $x = $ **x1** will be chosen. This command is particularly useful when assigning boundary conditions to a specific side of a rectangular domain.

● ● ●

Choose nodes y plane

As above but for the *y*-plane.

● ● ●

Choose nodes z plane

As above but for the z-plane.

$\bullet\bullet\bullet$

Choose nodes 3pt plane

1. **x1, y1, z1** xyz-coordinate of the first point.

2. **x2, y2, z2** xyz-coordinate of the second point.

3. **x3, y3, z3** xyz-coordinate of the third point.

4. **ptol** Distance from the plane.

Nodes within distance **ptol** of the plane defined by the 3 points will be chosen. This allows you to choose planes of nodes with an arbitrary orientation.

$\bullet\bullet\bullet$

Choose nodes 3pt plane bounded

1. **x1, y1, z1** xyz-coordinate of the first point.

2. **x2, y2, z2** xyz-coordinate of the second point.

3. **x3, y3, z3** xyz-coordinate of the third point.

4. **ptol** Distance from the plane.

5. **x4, x5** x-range of the block.

6. **y4, y5** y-range of the block.

7. **z4, z5** z-range of the block.

Nodes within distance **ptol** of the plane defined by the 3 points and within the rectangular block defined by the 3 ranges will be chosen.

$\bullet\bullet\bullet$

Choose nodes block

1. **x1, x2** x-range of the block.

2. **y1, y2** y-range of the block.

3. **z1, z2** z-range of the block.

Nodes within the rectangular block defined by the 3 ranges are chosen. 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

All nodes in the top sheet of the domain are chosen.

• • •

Choose nodes top block

1. **x1, x2** x-range of the block.

2. **y1, y2** y-range of the block.

3. **z1, z2** z-range of the block.

Nodes in the top sheet whose centroids are within the rectangular block defined by the 3 ranges are chosen. 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 bottom

All nodes in the bottom sheet of the domain are chosen.

• • •

Choose nodes sheet

1. **nsheet** Sheet number.

All nodes in this sheet are chosen.

• • •

Choose nodes top gb

1. **fname** Name of the GRID BUILDER chosen nodes file *gb_prefix*.`nchos`.*description*.

Nodes flagged as .TRUE. in the file and that are in the top sheet are chosen.

• • •

Choose nodes gb

1. **fname** Name of the GRID BUILDER chosen nodes file *gb_prefix*.`nchos`.*description*.

2. **nsheet_bot,nsheet_top** Bottom and top sheet numbers.

Nodes flagged as .TRUE. in the file and that are between the bottom and top sheet (inclusive) are chosen.

<center>• • •</center>

Choose nodes list

1. **fname** Name of the file which contains the list of node numbers.

The file consists of a list of node numbers, one entry per line. The procedure exits automatically when end-of-file is reached. The nodes in the list are chosen.

<center>• • •</center>

Choose nodes xyz list

1. **fname** Name of the file which contains the list of xyz-coordinates.

The file consists of a list of xyz-coordinates, one entry per line. The node nearest the given coordinate will be chosen. The procedure exits automatically when end-of-file is reached.

<center>• • •</center>

Choose surface flow nodes

Nodes currently flagged as surface flow nodes are chosen.

<center>• • •</center>

Choose nodes top boundary

Nodes around the top edge of the surface flow domain are chosen.

<center>• • •</center>

Choose nodes top overlay file

1. **fname** Name of the GRID BUILDER overlay file.

Only the first group of entries in the overlay file are considered. If an overlay line segment intersects an element edge, then the node in the top sheet closest to the point of intersection is chosen.

<center>• • •</center>

Choose nodes between gb surfaces

1. **surffile** Name of the GRID BUILDER nodal properties file *gb_prefix*.`nprop`.*description* for the bottom surface.

2. **surffile** Name of the GRID BUILDER nodal properties file *gb_prefix*.`nprop`.*description* for the top surface.

For the 3-D mesh node with the same $x-$ and $y-$coordinate as a 2-D mesh node, if the $z-$coordinate of the 3-D node is greater than the nodal property value given in the bottom surface file and less than the nodal property value given in the top surface file the 3-D node is chosen.

• • •

Choose nodes tecplot geometry

1. **geofile** Name of the TECPLOT geometry file which contains polyline data.

The node nearest to each point on the polyline is chosen.

• • •

Choose nodes horizontal circle

1. **x_mid, y_mid, z_mid** xy-coordinates of the centre of the circle and elevation of the circle.

2. **radius** Radius of the circle.

3. **ptol** Vertical tolerance.

4. **rtol** Horizontal tolerance.

Nodes within a vertical distance **ptol** of elevation **z_mid**, and within a horizontal distance **rtol** of the circle with centre **x_mid, y_mid** and radius **radius** are chosen. This allows you to choose nodes in a domain that has a circular ground-plan.

• • •

Choose nodes active/inactive boundary
Nodes which lie on the boundary between active and inactive elements will be chosen.

This instruction was intended to be used to select the surface of nodes on top of the set of active elements so that flow boundary conditions (e.g. head equals elevation) could be assigned.

• • •

Write chosen nodes

1. **fname** Name of the file to which the chosen node numbers will be written.

• • •

Write chosen nodes xyz

1. **fname** Name of the file to which the chosen node information will be written.

As above except xyz-coordinates are written instead of node numbers.

<div align="center">● ● ●</div>

5.4.2   Selecting Segments

We can use the following instructions to alter the set of chosen segments.

Clear chosen segments

All segments in the domain are flagged as *not* chosen. This is recommended if you are
unsure of which segments are chosen due to previously issued instructions.

<div align="center">● ● ●</div>

Choose segments all

All segments in the domain will be chosen. This is useful if you wish to assign a property
to all segments in the grid.

<div align="center">● ● ●</div>

Choose segments line

1. **x1, y1, z1** xyz-coordinates of the first end point of the line.

2. **x2, y2, z2** xyz-coordinates of the second end point of the line.

Segments which fall on or close to the line are chosen. The routine finds the two nodes
closest to the end points of the line and then finds the group of connected line segments
which form the shortest path between the two nodes.

<div align="center">● ● ●</div>

Choose segments polyline

1. **npts** The number of points defining the polyline, which should be entered in order
   from one end of the polyline to the other. Read the following **npts** times:

   (a) **x1, y1, z1** xyz-coordinates of a point on the polyline.

Segments which fall on or close to the polyline are chosen. The routine proceeds along the
polyline, considering two points at a time, finds the two nodes closest to the two current
points and then finds the group of connected line segments which form the shortest path
between the two nodes.

<div align="center">● ● ●</div>

5.4.3 Selecting Faces

Allow internal faces
Causes **grok** to define internal faces, which cut through elements.

By default, only the external faces (6 orthogonal faces for 8-node blocks and 5 faces for 6-node prisms) are defined for the mesh.
● ● ●

The following instructions are used to alter the set of chosen faces:

Clear chosen faces
All faces in the domain are flagged as *not* chosen. This is recommended if you are unsure of which faces are chosen due to previously issued instructions.
● ● ●

Choose faces all
All faces in the domain will be chosen. This is useful if you wish to assign a property to all faces in the grid. Rarely used.
● ● ●

Choose faces x plane

1. **x1** x-coordinate of the plane.

2. **ptol** Distance from the plane.

Faces within distance **ptol** of the plane defined by the equation $x = $ **x1** will be chosen. This command is particularly useful when assigning boundary conditions to a specific face of a rectangular domain.
● ● ●

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

1. **x1, y1, z1** xyz-coordinate of the first point.

2. **x2, y2, z2** xyz-coordinate of the second point.

3. **x3, y3, z3** xyz-coordinate of the third point.

4. **ptol** Distance from the plane.

Faces within distance **ptol** of the plane defined by the 3 points will be chosen. This allows you to choose planes of faces with an arbitrary orientation, and is particularly useful for setting up a set of sloping fractures.

• • •

Choose faces 3pt plane bounded

1. **x1, y1, z1** xyz-coordinate of the first point.

2. **x2, y2, z2** xyz-coordinate of the second point.

3. **x3, y3, z3** xyz-coordinate of the third point.

4. **ptol** Distance from the plane.

5. **x4, x5** x-range of the block.

6. **y4, y5** y-range of the block.

7. **z4, z5** z-range of the block.

Faces within distance **ptol** of the plane defined by the 3 points and within the rectangular block defined by the 3 ranges will be chosen.

• • •

Choose faces block

1. **x1, x2** x-range of the block.

2. **y1, y2** y-range of the block.

3. **z1, z2** z-range of the block.

Faces whose centroids are within the rectangular block defined by the 3 ranges are chosen. 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

1. **x1, x2** x-range of the block.

2. **y1, y2** y-range of the block.

3. **z1, z2** z-range of the block.

4. **nlaybot, nlaytop** Bottom and top element layer numbers.

Faces whose centroids are within the rectangular block which is defined by the 3 coordinate ranges, and which lie within the element layers defined by **nlaybot** and **nlaytop** are chosen. 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 (2-D 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 3-D element layer vary spatially.

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

1. **nsheet_bot,nsheet_top** Bottom and top sheet numbers.

Faces which are between the two specified sheets (inclusive) and are not oriented perpendicular to the sheet will be chosen.

● ● ●

Choose faces top
All faces in the top sheet of the domain will be chosen.

● ● ●

Choose faces top block

1. **x1, x2** x-range of the block.

2. **y1, y2** y-range of the block.

3. **z1, z2** z-range of the block.

Faces in the top layer whose centroids are within the rectangular block defined by the 3 ranges are chosen. 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 bottom
All faces in the bottom sheet of the domain will be chosen.

● ● ●

Choose faces front

Faces on the front of the domain will be chosen. This instruction can only be applied to meshes composed of block elements. Front faces are parallel to the xz coordinate plane and have small y coordinates.

• • •

Choose faces back

Faces on the back of the domain will be chosen. This instruction can only be applied to meshes composed of block elements. Back faces are parallel to the xz coordinate plane and have large y coordinates.

• • •

Choose faces left

Faces on the left side of the domain will be chosen. This instruction can only be applied to meshes composed of block elements. Left side faces are parallel to the yz coordinate plane and have small x coordinates.

• • •

Choose faces right

Faces on the right side of the domain will be chosen. This instruction can only be applied to meshes composed of block elements. Right side faces are parallel to the yz coordinate plane and have large x coordinates.

• • •

Choose faces top gb

1. **fname** Name of the GRID BUILDER chosen elements file *gb_prefix*`.echos.`*description*.

Faces flagged as .TRUE. in the file, are in the top sheet and are not oriented perpendicular to the sheet are chosen.

• • •

Choose faces top for chosen elements

Faces are chosen if they are in the top sheet and the 3-D element they belong to is chosen.

• • •

Choose faces gb

1. **fname** Name of the GRID BUILDER chosen elements file *gb_prefix*`.echos.`*description*.

2. **nsheet_bot,nsheet_top** Bottom and top sheet numbers.

Faces flagged as .TRUE. in the file, that are between the bottom and top sheets (inclusive), and that are not oriented perpendicular to the sheet are chosen.

● ● ●

Choose faces vertical from gb nodes

1. **fname** Name of the GRID BUILDER chosen nodes file *gb_prefix*.`nchos`.*description.*

2. **nsheet_bot,nsheet_top** Bottom and top sheet numbers.

This instruction is intended for use with meshes that are generated from GRID BUILDER 2-D meshes, and is used to choose faces that are oriented perpendicular to the mesh.

If a node is chosen in the 2D mesh, then the nodes in the 3D mesh that have the same XY-coordinate (i.e. that fall in the same column of nodes as the 2D node) and between the top and bottom sheets will be chosen. A face is then chosen if all of its four nodes are chosen.

● ● ●

Choose horizontal faces on layer

1. **nlayer** Element layer number.

2. **x1, x2** The x-range of the block.

3. **y1, y2** The y-range of the block.

Horizontal faces which are in the layer of elements numbered **nlayer** and within the rectangular block which is defined by the x and y ranges are chosen. This instruction can be used to select horizontal faces (e.g. to make fractures) when the elevation of a given layer of nodes is irregular.

● ● ●

Choose faces stairway

1. **x1, y1, z1** xyz-coordinates of the first point.

2. **x2, y2, z2** xyz-coordinates of the second point.

3. **x3, y3, z3** xyz-coordinates of the third point.

Horizontal and vertical faces of an inclined plane that is defined by three points are chosen. This instruction is mainly designed for verification purpose of modelling results using inclined fractures. Note that if using this instruction, fracture velocities are multiplied by a correction factor that accounts for the longer path that contaminants have to travel from node to node.

● ● ●

Choose fracture faces block

1. **x1, x2** x-range of the block.

2. **y1, y2** y-range of the block.

3. **z1, z2** z-range of the block.

Faces which are fracture elements and whose centroids are within the rectangular block defined by the 3 ranges are chosen. 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 face by nodes

1. **n1, n2, n3, n4** Node numbers of the face to be chosen.

The rectangular face which is made up of the 4 given nodes will be chosen.

● ● ●

Clear chosen faces by nodes
As above except the face will be cleared (i.e. not chosen).

● ● ●

Choose faces horizontal circle

1. **x_mid, y_mid, z_mid** xy-coordinates of the centre of the circle and elevation of the circle.

2. **radius** Radius of the circle.

3. **ptol** Vertical tolerance.

Faces within a vertical distance **ptol** of elevation **z_mid**, and within the circle with centre **x_mid, y_mid** and radius **radius** are chosen. This allows you to choose faces in a domain that has a circular ground-plan.

● ● ●

Write chosen faces

1. **fname** Name of the file to which the chosen face information will be written.

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. This instruction is intended to be used in conjunction with the following instruction.

● ● ●

Write chosen faces and host element numbers

1. **fname** Name of the file to which the chosen face and host element information will be written.

For each currently chosen face, this instruction writes the face number and associated 3-D element numbers. If the second element number is zero, the face is on the outside of the 3-D domain.

● ● ●

Read chosen faces

1. **fname** Name of the file from which the chosen face information will be read.

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.

● ● ●

Echo chosen faces

Causes the current set of chosen face numbers to be written to the *prefix*o.eco file.

● ● ●

5.4.4   Selecting Inclined Faces

These instructions only work for rectangular meshes with the standard element numbering scheme.

For each block element, there are 6 potential inclined faces which may be selected. These are given ID numbers according to the following convention:

```
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
```

Clear chosen inclined faces

All faces in the domain are flagged as *not* chosen. This is recommended if you are unsure of which inclined faces are chosen due to previously issued instructions.

Note that this instruction also clears chosen regular (horizontal and vertical) faces. This is necessary because a previously defined inclined plane may also consist of horizontal or vertical faces which have to be unselected as well.

• • •

Choose faces 3pt inclined plane

1. **nplane** Plane ID number, as defined above.

2. **x1, y1, z1** xyz-coordinates of the first point on the plane.

3. **x2, y2, z2** xyz-coordinates of the second point on the plane.

4. **x3, y3, z3** xyz-coordinates of the third point on the plane.

5. **ptol** Distance from the plane.

6. **xmin, xmax** x-range of the block.

7. **ymin, ymax** y-range of the block.

8. **zmin, zmax** z-range of the block.

Faces which have the appropriate plane ID, whose centroids lie within the distance **ptol** of the plane which is defined by the 3 points, and whose centroids are within the rectangular block defined by the 3 ranges are chosen.

Note that if the plane defined by the 3 points is parallel to one coordinate axis, the preprocessor will automatically use the ID of the plane parallel to that axis, and the user-defined plane ID will be ignored.

• • •

5.4.5   Selecting Elements

We can use the following instructions to alter the set of chosen elements.

Clear chosen elements

All elements in the domain will be flagged as *not* chosen. This is recommended if you are unsure of which elements are chosen due to previously issued instructions.

• • •

Choose elements all

All elements in the domain will be chosen. This is useful if you wish to assign a property to all elements in the grid.

• • •

Choose elements by zone

1. **nval** Zone number.

Elements which have zone numbers equal to the given value are chosen.

• • •

Choose elements by zone, within overlay

1. **nval** Zone number.

2. **fname** Name of the GRID BUILDER overlay file.

Elements which have zone numbers equal to the given value and whose centroid lies within the first group of entries in the overlay file are chosen.

• • •

Choose elements x plane

1. **x1** The x-coordinate of the plane.

2. **ptol** Distance from the plane.

Elements within distance **ptol** of the plane defined by the equation $x = \mathbf{x1}$ 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

1. **x1, y1, z1** xyz-coordinate of the first point.

2. **x2, y2, z2** xyz-coordinate of the second point.

3. **x3, y3, z3** xyz-coordinate of the third point.

4. **ptol** Distance from the plane.

Elements whose centroids are within distance **ptol** of the plane defined by the 3 points will be chosen. This allows you to choose planes of elements with an arbitrary orientation.

• • •

Choose elements block

1. **x1, x2** x-range of the block.

2. **y1, y2** y-range of the block.

3. **z1, z2** z-range of the block.

Elements whose centroids are within the rectangular block defined by the 3 ranges are chosen. 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

1. **x1, x2** x-range of the block.

2. **y1, y2** y-range of the block.

3. **z1, z2** z-range of the block.

4. **nlaybot, nlaytop** Bottom and top element layer numbers.

Elements whose centroids are within the rectangular block which is defined by the 3 coordinate ranges, and which lie within the element layers defined by **nlaybot** and **nlaytop** are chosen. 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 (2-D 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 3-D element layer vary spatially.

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 by layer

1. **nlaybot,nlaytop** Bottom and top element layer numbers.

Elements which are between the bottom and top layers are chosen.

• • •

Choose elements top

All elements which are in the top layer are chosen.

• • •

Choose elements list

1. **fname** Name of the file which contains the list of element numbers.

The file consists of a list of element numbers, one entry per line. The procedure exits automatically when end-of-file is reached. The elements in the list are chosen.

• • •

Choose elements xyz list

1. **fname** Name of the file which contains the list of xyz-coordinates.

One set of coordinates should be entered per line. If a coordinate falls within an element, then that element is chosen. *This instruction only works for rectangular structured meshes with the standard element numbering scheme, where the number of nodes in each of the x-, y-, z-directions (nx, ny, nz) is defined. It does not work for irregular grids generated with GridBuilder or GMS.*

• • •

Choose elements gb

1. **fname** Name of the GRID BUILDER chosen elements file *gb_prefix*.echos.*description*.

2. **nsheet_bot,nsheet_top** Bottom and top sheet numbers.

Elements flagged as .TRUE. in the file and that are between the bottom and top sheet are chosen.

• • •

The following instructions allow you to choose elements based on raster surfaces, as described in Section H. Since rasters are independent of the 3-D mesh, they do not have to be re-generated if the mesh changes, as is the case with Grid Builder or GMS formatted surfaces that have values corresponding to each node in the 2-D mesh.

Choose elements between raster surfaces

1. **rasterfile** Name of the raster file for the bottom surface.

2. **rasterfile** Name of the raster file for the top surface.

The xy-coordinates of the element centroid are used to determine if it falls in a valid cell (i.e. a cell that has no missing values) in both raster files. If it does, the coordinates are used to interpolate a raster value from the cell's four corner values. If the $z-$coordinate of the element centroid is greater than the raster value of the bottom surface, and less than the raster value of the top surface, then the element is chosen.

<div align="center">• • •</div>

The following instructions use the same input data structure except that a single raster file is read and used for comparison:

> Choose elements above raster surface
> Choose elements below raster surface

The following instructions use the same input data structure except that there is an additional constraint that the element is chosen only if it's current zone number is zero. They are intended to be used in conjunction with the instruction Assign zone zero:

> Choose elements between raster surfaces, iprop zero
> Choose elements above raster surface, iprop zero
> Choose elements below raster surface, iprop zero

The following instructions use the same input data structure except that GRID BUILDER nodal property files (i.e gb_prefix.`nprop`.$description$) are read instead of raster files. The element is chosen based on the z-coordinate of the first node in it's node list: Choose elements between gb surfaces

> Choose elements above gb surface
> Choose elements below gb surface

The following instructions use the same input data structure except that GMS scalar files are read instead of raster files. The element is chosen based on the z-coordinate of the first node in it's node list:

> Choose elements between gms surfaces
> Choose elements above gms surface
> Choose elements below gms surface

Choose elements from arcview ascii thickness map

1. **arcfile** Name of the Arcview ASCII file.

Reads an ARCVIEW ascii file containing thickness values (e.g. overburden thickness). For

each 3-D element, it's position (row and column) in the raster is determined and it's centroid is calculated. It is chosen if its centroid is greater than the surface elevation minus the thickness at that location.

$\bullet\bullet\bullet$

Choose elements horizontal circle

1. **x_mid, y_mid, z_mid** *xy*-coordinates of the centre of the circle and elevation of the circle.

2. **radius** Radius of the circle.

3. **ptol** Vertical tolerance.

4. **rtol** Horizontal tolerance.

Elements whose centroids are within a vertical distance **ptol** of elevation **z_mid**, and within a horizontal distance **rtol** of the circle with centre **x_mid, y_mid** and radius **radius** are chosen. This allows you to choose elements in a domain that has a circular ground-plan.

$\bullet\bullet\bullet$

5.5 Simulation Control Options

5.5.1 General

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.* The porous medium properties for the default problem, which are hard-wired in the code, are listed in Table 5.5. By default, the finite-element approach is used and a transport simulation is not done. If the default problem setup and material properties are acceptable, it is likely that the only additional data required to complete the definition of the problem are some flow boundary conditions, which can be assigned as described in Section 5.7.1.

Transient flow
Causes **HydroGeoSphere** to perform a time-stepping, transient flow solution.

$\bullet\bullet\bullet$

Unsaturated
Causes **HydroGeoSphere** to perform a variably-saturated flow solution.

$\bullet\bullet\bullet$

Do transport

Causes **HydroGeoSphere** to perform a transport solution.

• • •

If surface loading with hydromechanical coupling is required, you must issue the following instruction:

Surface loading
Causes **HydroGeoSphere** to consider surface loading with hydromechanical coupling.

• • •

The following instructions are used to define the behaviour of the Travel Time Probability Package described in Section 2.8.

Travel time PDF
Defines if a travel time PDF computation is to be performed.

• • •

Travel time CDF
Defines if a travel time CDF computation is to be performed.

• • •

Travel time PDF from CDF
Defines if the travel time PDF must be deduced from the CDF at observation points.

• • •

Mean age
Defines if a mean age/mean life expectancy computation is to be performed.

• • •

Backward-in-time
Defines if flow is to be reversed (backward transport solution).

• • •

Evaluate capture zone
Defines if an outlet capture zone is to be evaluated.

• • •

Species attribution
Defines the species number (absolutely necessary in the case of multi-species transport).

• • •

These instructions are of general interest:

Y vertical

Causes **HydroGeoSphere** to assume that the y-coordinate points in the vertical direction (i.e. instead of the z-coordinate).

This instruction *does not switch coordinates*, but merely cause **HydroGeoSphere** to use the y-coordinate of a node to calculate the total hydraulic head (pressure + elevation) for variably-saturated simulations. It is intended to be used for variably-saturated flow problems when using triangular prism elements, when one wants to have the triangular mesh (which is defined in the xy plane) to be oriented along the vertical direction.

● ● ●

Data check only

Causes **HydroGeoSphere** to halt execution after reading all data files, initializing arrays etc. but prior to the start of the solution procedure

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.

● ● ●

5.5.1.1 Finite-difference Options

Finite difference mode

Causes **HydroGeoSphere** to use the finite difference approach instead of the default finite element method.

● ● ● The following instructions affect the transport simulation in a general way when finite difference method is being used.

Compute fd cross terms

Causes **HydroGeoSphere** to compute cross terms explicitly when a finite difference representation is chosen, which, by default, are ignored.

● ● ●

Control volume

Causes the control-volume finite difference approach to be used instead of the default standard finite difference approach. This instruction has no effect when the finite element approach is being used because the control volume approach is always used in that case.

● ● ●

5.5.1.2   Matrix Solver

The matrix solution procedure consists of three phases: initialization, preconditioning and solution. The instructions presented here can be used to control the preconditioning and solution phases in order to increase the efficiency of the model.

The default preconditioning scheme is level-based factorization without red/black system reduction. Improving model performance requires a good understanding of these options, and so if you are unsure you should just use the default settings. However, if you decide to experiment with the solver preconditioning parameters, here are a few suggestions for doing so:

1. For transient, simple (i.e. small number of nodes) problems, level-based preconditioning works better, because the static data structure analysis does not need to be done for each time step.

2. For steady state or complex problems, drop tolerance preconditioning works better, because WATSIT spends most of its time in the solution phase, not the preconditioning phase.

3. For a very smoothly varying solution (e.g. a weakly stressed, homogeneous property field) red/black reduction will speed convergence.

Level of fill

1. **level** Level of fill.

Assigns the level of fill to be preserved in level-based factorization, which defaults to 0. If drop tolerance preconditioning is used, this value is not used.

● ● ●

Red black reduction

Tells the solver to use red black reduction. This can be used either using level-based or drop tolerance preconditioning.

● ● ●

Drop tolerance preconditioning

Tells the solver to use drop tolerance preconditioning. This will remove elements based on how small they are. The default threshold is 0.1.

● ● ●

Drop tolerance threshold

1. **thres** Drop tolerance threshold.

Assign a new drop tolerance threshold.

• • •

Once the preconditioning phase is complete, the matrix can be solved using several acceleration techniques. The following command can be used to change the solver procedure:

Solver acceleration technique

1. **iaccel** Type of acceleration to use.

Assigns a new value for the acceleration technique for the linear solver, which defaults to 3 (CGSTAB-P). Appropriate values are 0 (CG, for symmetric matrices only), 1 (OrthoMin), 2 (CGS), 3 (CGSTAB-P) or 4 (GMRES). If unsure, don't use this command.

• • •

No matrix scaling
Tells the solver not to use matrix scaling preconditioning.

• • •

5.5.2 Timestep Control

Before we discuss the instructions which are available for controlling the behaviour of a transient 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 **HydroGeoSphere** which uses it to produce timestep values.

As well, adaptive timestepping, as discussed in Section 5.5.2.1, can be used to 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 time-stepping behaviour of a transient solution:

Initial time

1. **tinit** Initial time.

Assigns a new value for the initial time, which defaults to zero. This is useful if you are restarting the simulation and want to index the times used to an earlier run.

• • •

Initial timestep

1. **val** Initial timestep size.

Assigns a new value for the initial timestep, which defaults to 0.01 time units.

• • •

Maximum timestep

1. **val** Maximum timestep size.

Assigns a new value for the maximum timestep size, which defaults to 10^{25} time units.

• • •

Minimum timestep

1. **val** Minimum timestep size.

Assigns a new value for the minimum timestep size, which defaults to 1×10^{-10} time units. If the timestep becomes smaller than this value as a result of the adaptive timestepping procedure, **HydroGeoSphere** will stop and issue a diagnostic message.

• • •

Target times

1. **target_time...end** Target times.

Listed times are added to the current set of target times.

• • •

Generate target times

1. **tstart** Start time [T].

2. **delta** Initial time step size [T].

3. **tinc** Time step multiplier.

4. **dtmax** Maximum time step size allowed [T].

5. **tend** End time [T].

New target times are generated from the start time **tstart** to end time **tend** by repeatedly adding the time step **delta**, which is increased each time by the multiplier **tinc** until it reaches a maximum size of **dtmax**.

• • •

Output times

1. **output_time...end** Output time.

Listed times are added to the current set of output times (i.e. times for which you want detailed output). Note that these values will automatically become part of the target time array.

• • •

5.5.2.1 Adaptive Timesteps

If required, **HydroGeoSphere** can modify timestep values as the solution proceeds, based on the transient behaviour of the system (see Equation 3.106). The following instructions can be used to activate this feature, and set targets for specific variables (e.g. pressure head, saturation etc.). These targers are used to modify timestep size as the solution proceeds.

Head control

1. **dhead_allowed** Maximum allowed absolute change in nodal head during any time step [L].

• • •

Water depth control

1. **ddepth_allowed** Maximum allowed absolute change in nodal surface water depth during any time step [L].

• • •

Saturation control

1. **dsat_allowed** Maximum allowed absolute change in nodal saturation during any time step [L].

• • •

Newton iteration control

1. **nnri_allowed** Maximum allowed number of Newton-Raphson iterations during any time step [L].

• • •

Concentration control

1. **dconc_allowed** Maximum allowed absolute change in nodal concentration during any time step [L].

• • •

Mass change control

1. **dmass_change_allowed** Maximum allowed absolute change in mass during any time step [L].

• • •

Mass error control

1. **dmass_error_allowed** Maximum allowed absolute mass error during any time step [L].

• • •

The following controls are used to generate a timestep multiplier according to Equation 3.106. There are limits to how large or small the multiplier can be, and these are set by default to be in the range of 0.5 to 2 respectively. If you want to modify these limits you can do so using the following instructions:

Maximum timestep multiplier

1. **val** Maximum timestep multiplier value.

Assigns a new value for the maximum timestep multiplier, which defaults to 2.

• • •

Minimum timestep multiplier

1. **val** Minimum timestep multiplier value.

Assigns a new value for the minimum timestep multiplier, which defaults to 0.5. Currently, **HydroGeoSphere** ignores this parameter and allows *any non-zero minimum, no matter how small.*

• • •

5.5.3 Saturated Flow

Pressure head input
Causes all heads which are input to be treated as pressure heads instead of hydraulic heads.

<div align="center">● ● ●</div>

Freshwater pressure head
Causes all heads which are input and output to be treated as freshwater pressure heads instead of hydraulic heads.

<div align="center">● ● ●</div>

Fluid pressure
Causes all heads which are input and output to be treated as fluid pressures instead of hydraulic heads.

<div align="center">● ● ●</div>

No fluid mass balance
This instruction suppresses the calculation of fluid mass balance information which is, by default, computed at each time step.

<div align="center">● ● ●</div>

Flow time weighting

1. **tw** Time-weighting factor for the flow solution.

Assigns a new value for the time-weighting factor for the flow solution, which defaults to 1.0. Values must be greater than 0.0 and less than or equal to 1.0.

<div align="center">● ● ●</div>

Flow solver maximum iterations

1. **maxfit** Maximum number of flow solver iterations.

Assigns a new value for the maximum number of flow solver iterations, which defaults to 2000.

<div align="center">● ● ●</div>

Flow solver convergence criteria

1. **restol** Flow solver convergence criteria.

Assigns a new value for the flow solver convergence criteria, which defaults to 1×10^{-10}.

Convergence 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:

$$r = b - [A]x$$

• • •

Flow solver detail

1. **isolv_info** Flow solver detail level.

Assigns a new value for the flow solver detail level, which defaults to 0. This value controls the level of detail of solver performance information printed to the screen and listing file, and can have values of 0 (no information) or 1 (full information).

• • •

5.5.4   Variably-saturated Flow

By default, **HydroGeoSphere** uses the upstream weighting scheme (weighted harmonic mean) for relative permeability, with an upstream weighting factor value of 1.0. The default behaviour can be changed using the following commands:

Upstream weighting factor

1. **upwfactor** Upstream weighting factor.

Assigns a new value for the upstream weighting factor, which defaults to 1.0. This value should be in the range from 0.5 (central weighting) to 1.0 (upstream weighting).

• • •

Central weighting
Causes **HydroGeoSphere** to use central weighting (weighted arithmetic mean) instead of upstream weighting.

• • •

Primary variable switching

Causes **HydroGeoSphere** to use the primary variable substitution technique outlined in Section 3.12.3. If desired, the default values of tol_f and tol_b, the upper and lower limits for variable substitution (see Equation 3.103) can be changed using the following commands:

• • •

Upper limit

1. **switch_t** Upper limit of the variable substitution approach, tol_f in Equation 3.103. The default value is 0.99.

• • •

Lower limit

1. **switch_f** Lower limit of the variable substitution approach, tol_b in Equation 3.103. The default value is 0.89.

• • •

5.5.4.1 Newton Iteration Parameters

The following parameters can be used to control the Newton-Raphson iteration scheme for solution of the variably-saturated flow problem as described in Section 3.12.2.

Newton maximum iterations

1. **maxnewt** Maximum number of Newton iterations.

Assigns a new value for the maximum number of Newton iterations, which defaults to 15. If this number is exceeded during a time step, the current time step value is reduced by half and a new solution is attempted.

• • •

Jacobian epsilon

1. **epsilon** Jacobian epsilon.

Assigns a new value for the Jacobian epsilon, which defaults to 1×10^{-4}. The Jacobian epsilon is the shift in pressure head used to numerically compute the derivatives in the Jacobian matrix. As a rule of thumb, a value equal to 10^{-5} times the average pressure head in the domain is recommended.

• • •

Newton absolute convergence criteria

1. **delnewt** Newton absolute convergence criteria.

Assigns a new value for the Newton absolute convergence criteria, which defaults to 1×10^{-5}. 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.

● ● ●

Newton residual convergence criteria

1. **resnewt** Newton residual convergence criteria.

Assigns a new value for the Newton residual convergence criteria, which defaults to 1×10^{-8}. Convergence of the solution occurs when the maximum absolute nodal residual (see Section 5.5.3) in the domain for one Newton iteration exceeds this value.

● ● ●

Newton maximum update for head

1. **NR_dhtol** Newton maximum update for head.

Assigns a new value for the Newton maximum update for head, which defaults to 1.0. This is used to calculate the underrelaxation factor ω_r in such a way that:

$$\omega_r = \textbf{NR\_dhtol}/\max(dh_r) \tag{5.6}$$
$$h_r = h_{r-1} + \omega_r * dh_r \tag{5.7}$$

where $\max(dh_r)$ is the computed maximum update for head in r^{th} Newton iteration, and h_r is the head flow solution after r iterations. As **NR_dhtol** becomes smaller, the Newton solution becomes more stable but with possibly more iterations. For highly nonlinear problems for which Newton linearization easily fails to converge, it is recommended to set this value smaller.

● ● ●

Newton maximum update for depth

1. **NR_ddtol** Newton maximum update for depth.

Assigns a new value for the Newton maximum update for water depth, which defaults to 1×10^{-2}. The same as **NR_dhtol** above but applies only to water depth.

● ● ●

Newton maximum residual increase

1. **NR_resnorm_fac** Newton maximum residual increase.

Assigns a new value for the Newton maximum residual increase, which defaults to 1×10^{30}.

If the Newton maximum residual increases by more than the value .The same as **NR_resnorm_fac**, the Newton loop is restarted with a smaller time step.

$\bullet \bullet \bullet$

Remove negative coefficients

Forces negative inter-nodal conductances to zero. Negative inter-nodal conductances result in inter-nodal flow from lower to higher heads and can cause oscillatory behavior during Newton iterations [*Letniowski and Forsyth*, 1991].

$\bullet \bullet \bullet$

Nodal flow check tolerance

1. **n_flow_check_tol** Assigns a new value for the nodal flow check tolerance, which defaults to 1×10^{-2}.

This checks that the relative local (nodal) fluid mass balance is acceptable for all nodes:

$$[(Q_{in} - Q_{out} + dM)/Q_{in}] < \textbf{n\_flow\_check\_tol}$$

Absolute fluid mass balance can always be satisfied against the global convergence criteria, if local inflows and outflows, and mass accumulation are very small.

$$(Q_{in} - Q_{out} + dM) < \text{global\_tolerance}$$

and where Q_{in}, Q_{out}, and dM (mass accumulation) are much less than 1.0. This can deteriorate the transport solution, as the concentration is defined as solute mass per unit fluid mass.

$\bullet \bullet \bullet$

No nodal flow check

Turns off the nodal flow check feature. In cases where a transport solution is not required, the nodal flow check is not necessary.

$\bullet \bullet \bullet$

Underrelaxation factor

1. **under_rel** Underrelaxation factor.

Assigns a new value for the underrelaxation factor for the Newton iteration, which defaults to 1. This value can range from 0 (full underrelaxation) to 1 (no underrelaxation).

$\bullet \bullet \bullet$

Compute underrelaxation factor

Causes the underrelaxation factor ω to be computed according to the following method described by *Cooley* [1983]:

$$
\begin{aligned}
\omega_{r+1} &= \frac{3+s}{3+|s|} & if\ s \geq -1 \\
&= \frac{1}{2|s|} & if\ s < -1
\end{aligned}
\tag{5.8}
$$

where:

$$
\begin{aligned}
s &= \frac{e_{r+1}}{e_r \omega_r} & if\ r > 1 \\
&= 1 & if\ r = 1
\end{aligned}
\tag{5.9}
$$

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

1. **dellim** Upper limit on the computed underrelaxation factor.

Assigns a new value for the upper limit on the computed underrelaxation factor, which defaults to 1000. A suggested value is 10 times the system domain thickness.

• • •

Minimum relaxation factor allowed

1. **min_relfac_allowed** Minimum value allowed for computed underrelaxation factor.

Assigns a new value for the lower limit on the computed underrelaxation factor, which defaults to 0.001. If not the first timestep, and the computed underrelaxation factor is less than this value, the current timestep is cut in half and the Newton Raphson iteration loop is re-started.

• • •

Newton information
Causes **HydroGeoSphere** to write more detailed information to the listing file about the performance of the Newton iteration process.

• • •

5.5.5   Discrete Fracture Flow

By default, **HydroGeoSphere** does not simulate discrete fracture flow unless a discrete fracture flow zone is created using the methods and instructions outlined in Section 5.8.2.

Also by default, **HydroGeoSphere** uses the common node approach to define the discrete fracture flow domain. If the dual-node approach is required, you must issue the following instruction:

> Dual nodes for fracture flow

5.5.6 Surface Flow

By default, **HydroGeoSphere** does not simulate surface flow unless a surface flow zone is created using the methods and instructions outlined in Section 5.8.2.

Dual nodes for surface flow

Causes **HydroGeoSphere** to use the dual-node approach to define the discrete surface flow domain. By default, the common node approach is used.

• • •

5.5.7 Transport

Transport time weighting

1. **twc** Time-weighting factor for the transport solution.

Assigns a new value for the time-weighting factor for the transport solution, which defaults to 0.5. Values can range from 0.0 (explicit) to 0.5 (central or Crank-Nicholson) or 1.0 (fully implicit). Fully implicit time-weighting is less prone to exhibit oscillations but more prone to numerical smearing than central time-weighting.

• • •

Peclet number

1. **pectol** Peclet number.

Assigns a new value for the Peclet number, which defaults to 1×10^{20}. The Peclet number does not influence the solution in any way, but is merely used to generate warning messages. The Peclet number can be computed as:

$$P_e = \mathbf{v} \frac{\Delta x}{\mathbf{D}}$$

where $\mathbf{v}$ is a flow velocity, Δx is an element length, in this case in the x-direction, and $\mathbf{D}$ is a dispersion coefficient. It is a measure of the adequacy of mesh fineness, with large numbers indicating poor spatial discretization.

The large default value suppresses the generation of warning messages.

$$\bullet \bullet \bullet$$

Output peclet number

This instruction causes **HydroGeoSphere** to write the Peclet number for each element to
the file *prefix*o.peclet_number, at the first timestep only.

$$\bullet \bullet \bullet$$

Courant number

1. **courtol** Courant number.

Assigns a new value for the Courant number, which defaults to 1×10^{20}. The Courant
number does not influence the solution in any way, but is merely used to generate warning
messages. The Courant number can be computed as:

$$C_e = \mathbf{v} \frac{\Delta t}{\Delta x}$$

where $\mathbf{v}$ is a flow velocity, Δx is an element length, in this case in the x-direction, and Δt
is the timestep length. It is a measure of the adequacy of timestep size, with large numbers
indicating poor temporal discretization.

$$\bullet \bullet \bullet$$

Transport solver convergence criteria

1. **restolc** Transport solver convergence criteria.

Assigns a new value for the transport solver convergence criteria, which defaults to 1×10^{-10}.
Convergence is obtained when this criteria is less than the absolute maximum value of the
residual (error) of the latest transport solution.

$$\bullet \bullet \bullet$$

Transport solver detail

1. **isolv_infoc** Transport solver detail level.

Assigns a new value for the transport solver detail level, which defaults to 0. This value
controls the level of detail of solver performance information printed to the screen and listing
file, and can have values of 0 (no information) or 1 (full information).

$$\bullet \bullet \bullet$$

Transport solver maximum iterations

1. **maxit** Maximum number of transport solver iterations.

Assigns a new value for the maximum number of transport solver iterations, which defaults to 2000.

$\bullet\bullet\bullet$

Upstream weighting of velocities

1. **almax,btmax,gammax** Upstream weighting factors for the x-, y-, and z-directions respectively.

Causes upstream-weighting of velocities to be used, as opposed to the default, central weighting of velocities. Values can range from 0.0 (no upstream-weighting) to 1.0 (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.

$\bullet\bullet\bullet$

Flux limiter for transport

This instruction causes **HydroGeoSphere** to use the van Leer flux limiter as described in Section 3.8.1.

$\bullet\bullet\bullet$

Iteration parameters flux limiter

1. **maxiter_flim, resmax_flim, delmax_flim** Maximum number of iterations, residual and absolute convergence criteria respectively for the non-linear flux limiter iteration procedure.

Assigns new values to the parameters that affect the behaviour of the van Leer flux limiter, and which default to 15, 1×10^{-5} and 1×10^{-8} respectively.

$\bullet\bullet\bullet$

Overland advective solute exchange only

This instruction only affects transport in coupled surface/subsurface systems, and causes **HydroGeoSphere** to neglect dispersive/diffusive exhange between the two domains. This is intended to be used to gauge the relative importance of advective vs. dispersive exchange or when comparing **HydroGeoSphere** to codes that neglect dispersive exchange.

$\bullet\bullet\bullet$

Fracture advective solute exchange only

This instruction only affects transport in coupled fracture/subsurface systems, and causes **HydroGeoSphere** to neglect dispersive/diffusive exhange between the two domains. This is intended to be used to gauge the relative importance of advective vs. dispersive exchange or when comparing **HydroGeoSphere** to codes that neglect dispersive exchange.

$\bullet\bullet\bullet$

The following three instructions can be used to define a threshold concentration for flagging output and optionally stopping a run.

Detection threshold concentration

1. **detection_threshold_conc** Detection threshold concentration $[\text{M L}^{-3}]$. This instruction sets the value which will be used to control the behaviour of the next two instructions.

<div align="center">● ● ●</div>

Flag observation nodes if exceed detection threshold concentration
Causes **HydroGeoSphere** to tag observation well output with the string `> detection threshold` if the concentration at the node exceeds **detection_threshold_conc**, defined above.

<div align="center">● ● ●</div>

Stop run if flux output nodes exceed detection threshold concentration
Causes **HydroGeoSphere** to halt execution if the concentration at any observation well node exceeds **detection_threshold_conc**, defined above.

<div align="center">● ● ●</div>

5.5.8   Density-dependent Flow and Transport Solution

This option should only be used for fully-saturated flow conditions.

5.5.8.1   Relative concentration as primary variable

The following instruction can be used to enable a density-dependent flow and transport solution. Note that the single species involved in the transport process is described by *RELATIVE CONCENTRATIONS*.

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 without obtaining convergence, the time step is cut in half and the loop is started over.

4. **linear_rho_c** A logical switch which determines how fluid density is calculated from relative concentration. If .true., a linear relationship is used, otherwise nonlinear.

5. **cmax** The maximum relative concentration, corresponding to the fluid with maximum density. This instruction proved to be especially useful to simulate the lab experiments by Oswald and Kinzelbach (2004), where the concentration of the fluid with maximum density is not 1.

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
.true.    !  linear rho-c relationship
1.0       !  maximum concentration
```

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. A linear relationship between fluid density and concentration is used and the maximum relative concentration (associated with the maximum fluid density) is 1.0

• • •

5.5.8.2 Absolute concentration and temperature as primary variables

The following instruction can be used to enable a density-dependent flow and transport solution. In this case, fluid density is directly calculated from individual species concentrations and temperature. The allowed species are Na^+, K^+, Ca^{2+}, Mg^{2+}, Cl^-, SO_4^{2-}, CO_3^{2-} and HCO_3^- and fluid temperature T, which have to be defined within the solute definition block (Chapter 5.6.3.1). Note that the units must be mg l^{-1} and °C, respectively.

Use Pitzer model

With this instructions, fluid density is calculated from species concentrations using Pitzer's ion interaction model. Note that this method can be very time-consuming because fluid density is calculated by iterating between density and molality. The default is a faster and non-iterative empirical approach.

• • •

Picard convergence criteria

1. **toldens** The minimum absolute difference in head, concentration and temperature for convergence of the non linear Picard loop.

2. **maxitdens** The maximum number of iterations for the Picard loop. If the maximum number of iterations is reached without obtaining convergence, the time step is cut in half and the loop is started over.

Similar to the use of the "Density-dependent transport" instruction when using relative concentration, the above instruction assigns new values to the convergence criteria. The defaults are 1.0×10^{-5} and 100, respectively.

<div align="center">● ● ●</div>

5.5.8.3   Salt mass fraction as primary variable

The following instruction can be used to enable a density-dependent flow and transport solution. Note that the single species involved in the transport process is described by *RELATIVE CONCENTRATIONS*.

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 without obtaining convergence, the time step is cut in half and the loop is started over.

4. **linear_rho_c** A logical switch which determines how fluid density is calculated from relative concentration. If .true., a linear relationship is used, otherwise nonlinear.

5. **cmax** The maximum relative concentration, corresponding to the fluid with maximum density. This instruction proved to be especially useful to simulate the lab experiments by Oswald and Kinzelbach (2004), where the concentration of the fluid with maximum density is not 1.

Note that you must set up the problem with *RELATIVE CONCENTRATIONS*. Therefore, rhomax is the fluid density for a relative concentration equal to 1.

<div align="center">● ● ●</div>

As an example:

```
1200.0    rhomax
0.02      tolerance
```

```
100       maximum iterations
.true.    linear rho-c relationship
1.0       maximum concentration
```

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. A linear relationship between fluid density and concentration is used and the maximum relative concentration (associated with the maximum fluid density) is 1.0

5.5.9 Heat transfer

To enable heat transfer, you must define the temperature species in the solute definition block as described in Chapter 5.6.3.1. Note that the unit for temperature must be °C and that all heat transfer properties must be given in the SI-units kilogram-meter-second-Celsius. The thermal properties of the solids are specified in the material property file. The following instructions can be used to control the heat transfer solution:

Do heat transfer...End
Causes **grok** to begin reading a group of heat transfer instructions until it encounters an End instruction.

• • •

The available instructions are:

Thermal conductivity of water

1. **k_l** [W L^{-1} K^{-1}] Thermal conductivity of the liquid phase.

Assigns a uniform value to the thermal conductivity of groundwater. If this instructions is used, the thermal conductivity of water is assumed constant and equal to k_l. It is therefore not calculated from the water temperature, which is the default setting.

• • •

Specific heat capacity of water

1. **c_l** [J kg^{-1} K^{-1}] Specific heat capacity of the liquid phase.

Assigns a uniform value to the specific heat capacity of water. If this instructions is used, the specific heat capacity of water is assumed constant and equal to c_l. It is therefore not calculated from the water temperature, which is the default setting.

• • •

Mechanical heat dispersion

Causes mechanical heat dispersion to be simulated. The default is no mechanical heat dispersion.

• • •

The following instruction can be used to define initial temperatures for the problem:

Initial temperature profile

1. **temp_top** [°C] The temperature at the top of the domain.

2. **temp_grad** [K m^{-1}] The prevailing geothermal gradient.

Calculates a depth profile of temperature and assigns the values to the initial temperature.

• • •

The following instructions can be used to define the heat source boundary condition:

Zero order source

1. **npanel** Number of panels in the time-variable, zero-order source function. For each panel, enter the following:

    (a) **ton_val,toff_val, (bc_val(j),j=1,nspeciesmob)** Time on [T], time off [T] and mass of solute produced per unit volume of porous medium solids per unit time [M L^{-3}].

Nodes in the chosen zones area assigned zero-order source boundary conditions.

A panel is a point in time at which the source term is set to a new value. The first panel would normally start at time zero. The source term given for the last panel will be maintained until the end of the simulation. You can assign a static source term for the duration of the simulation by setting **npanel** to 1, **ton_val** to 0.0 and **toff_val** to a large number.

Note that if **nspeciesmob** is greater than 1, additional values of **bc_val** should be included.

• • •

Exponential zero order source

1. **heat_q_zero** Heat production at t=0.

2. **heat_constant** Constant in the exponential function.

Nodes in the chosen zones area assigned an exponentially decreasing zero-order heat source boundary conditions.

<div align="center">● ● ●</div>

5.5.10   Inactive Elements

These instructions can be used to discretize irregular boundaries with block elements by deactivating portions of the grid, where 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.

<div align="center">● ● ●</div>

Make zone inactive
All elements in the current set of chosen zones will become inactive.

<div align="center">● ● ●</div>

For a 2-D slice made of 4-node rectangular elements, the following instruction can be used to make elements inactive:

Make element inactive using shapefile

1. **arcview_prefix** Prefix of the ARCVIEW shape file.

2. **unproject_file** If .true, this logical switch causes **grok** to read grid unprojection data as described in Section 5.3.10 and to apply it to the data read from the ARCVIEW shapefile.

3. **project_file** If .true, this logical switch causes **grok** to read grid projection data as described in Section 5.3.10 and to apply it to the data read from the ARCVIEW shapefile.

4. **outside** If .true, elements located outside the area defined in the ARCVIEW shapefile will become inactive. If .false, elements located inside the area will become inactive.

<div align="center">● ● ●</div>

Write inactive elements to file

1. **inactive_file** Name of the file to which the inactive element information will be written.

Setting up inactive elements with the Make element inactive using shapefile instruction can be time consuming in **grok** and this step does not need to be repeated as long as the grid structure remains the same. This instruction is intended to be used in conjunction with the following instruction.

<center>• • •</center>

Read inactive elements from file

1. **inactive_file** Name of the file from which the inactive element information will be read.

The results of successive calls to this instruction are cumulative, if different sets of inactive elements are read.

<center>• • •</center>

5.6   Initial Conditions

5.6.1   Subsurface Flow

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

1. **hval** Initial head [L].

Chosen nodes in the currently active media (see Section 5.8.1) are assigned an initial head value.

<center>• • •</center>

Initial head surface elevation
All nodes in the currently active media (see Section 5.8.1) are assigned an initial head value equal to the surface elevation at the same xy location.

<center>• • •</center>

Initial head from depth-saturation table
Scope: .grok

1.   **depth(1), saturation(1)** First entry.
2.   **depth(2), saturation(2)** Second entry.

: etc.

    n.   **depth(n), saturation(n)** n^{th} entry.

    n+1. **end** The string 'end'

Paired values of depth z and saturation S should be entered from smallest to largest depth. The last line of the table must be an **end** card, and the number of entries in the list are counted automatically to determine the table size.

Chosen nodes in the currently active media (see Section 5.8.1) are first assigned a saturation which is interpolated from the depth-saturation table. The nodes depth is calculated relative to the node on ground surface at the same xy location. The computed saturation is then converted to an initial head value using the constitutive relationships (see Section 5.8.3) for the zone number of the element containing the chosen node. In cases where a node is shared by two elements with different zone numbers, then the zone number of the lowest numbered element is used.

<p align="center">• • •</p>

Initial head from file

    1. **fname** Name of the file which contains the initial head data.

All nodes in the currently active media (see Section 5.8.1) are assigned an initial head value which is read from a free-format ascii file. The heads must be listed in the file in order from node 1 to node NN. Each line can contain one or more values.

<p align="center">• • •</p>

Initial head from output file

    1. **fname** Name of the file which contains the initial head data.

All nodes in the currently active media (see Section 5.8.1) are assigned an initial head value which is read from a previously generated output file with a name of the style *prefix*o.head.-**suffix**, where suffix is a 3-digit number identifying the output file. Since it is applied to the current media type you can use, for example, `prefixo.head.003` for the porous media and `prefixo.head_overland.003` for the surface domain. This can be useful, for example, when a run crashes (since you don't have to start simulation from scratch) or if you want to use separate solutions of groundwater and surface flow to start a coupled surface-subsurface simulation.

<p align="center">• • •</p>

Initial head subsurface from surface output file

    1. **fname** Name of the file which contains the initial surface domain head data.

All nodes in the currently active media (see Section 5.8.1) are assigned an initial head value equal to the surface domain head at the same xy location. The surface heads are read from

a previously generated output file with a name of the style *prefix*o.head_overland.**suffix**, where suffix is a 3-digit number identifying the output file.

<div align="center">● ● ●</div>

Function x initial head

1. **x1,h1** The x-coordinate and initial head value for the first point.

2. **x2,h2** The x-coordinate and initial head value for the second point.

Using linear interpolation, chosen nodes in the currently active media (see Section 5.8.1) are assigned initial head values, based on their x-coordinate and given heads at two points.

<div align="center">● ● ●</div>

Function z initial head
As above but as a function of z.

<div align="center">● ● ●</div>

Initial head raster

1. **rasterfile** Name of the raster file containing the initial head data. This is a string variable. The file should be formatted as outlined in Section H.

All nodes in the currently active media (see Section 5.8.1) are assigned an initial head value equal to the raster value at the same xy location.

<div align="center">● ● ●</div>

Map initial head from raster
Scope: .grok
1. **rasterfile** Name of the raster file containing the initial head values. This is a string variable. The file should be formatted as outlined in Section H.

For each node in each element in the set of currently chosen zones, a value for the initial head will be interpolated from the raster file data.

This instruction currently only generates initial head values for the porous medium.

<div align="center">● ● ●</div>

Restart file for heads

1. **flow_restart_file_name** Name of the file which contains the results of the previous flow solution.

All nodes are assigned an initial head value which is read from the file. This allows heads from a previous run to be used 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.

Since, by default, head values for the last time step are output in BINARY format to a file with the suffix *prefix*o.hen, these are always available for restarting a flow run. It is recommended that this file be renamed in order to avoid accidentally overwriting it and changing the restart conditions for a later run.

<div align="center">● ● ●</div>

Compute velocity field from head

1. **v_head_file** Name of the file which contains the results of the previous flow solution.

A flow solution is not performed. Instead, the velocity field is computed from the previous flow solution, a steady-state flow field is assumed, and the transport solution proceeds.

The heads are read from a previously generated output file with a name of the style *prefix*-o.head.**suffix**, where suffix is a 3-digit number identifying the output file.

<div align="center">● ● ●</div>

Compute velocity field from head and conc.

1. **v_head_file** Name of the file which contains the results of the previous flow solution.

2. **v_conc_file** Name of the file which contains the results of the previous transport solution.

3. **vrhomax, vcmax** Assumed maximum density and concentration.

A flow solution is not performed. Instead, the velocity field is computed from the previous flow solution and transport solution, a steady-state flow field is assumed, and the density-dependent transport solution proceeds.

The heads are read from a previously generated output file with a name of the style *prefix*-o.head.**suffix**, where suffix is a 3-digit number identifying the output file.

The concentrations are read from a previously generated output file with a name of the style *prefix*o.concentration.*species*.**suffix**, where suffix is a 3-digit number identifying the output file.

<div align="center">● ● ●</div>

5.6.2 Surface Flow

Initial water depth

1. **val** Initial water depth.

Chosen surface flow nodes are assigned an initial depth value.

• • •

Initial water depth from gb file

1. **fname** Name of the GRID BUILDER ascii-formatted nodal properties file *gb_prefix*-`.ascii_nprop.`*description* which contains the initial water depth data.

This instruction assumes that the surface flow domain covers the entire top of the 3-D domain, and contains NN2D (number of nodes in the 2-D slice) nodes. All surface flow media nodes (see Section 5.8.1) are assigned an initial water depth value which is read from file **fname**. This is an ascii formatted file, with the water depth values listed in order from node 1 to NN2D.

• • •

5.6.3   Transport

It should be noted that the instructions given here affect the behaviour of one or more solutes, which should already have been defined according to the instructions given in Section 5.6.3.1.

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

1. **bc_val(i), i=1,nspeciesmob** Initial concentration $[\text{M L}^{-3}]$ for each solute.

Chosen nodes in the currently active media (see Section 5.8.1) are assigned an initial concentration value.

• • •

Initial concentration from output file

1. **fname** Name of the file which contains the initial concentration data.

All nodes in the currently active media (see Section 5.8.1) are assigned an initial concentration value which is read from a previously generated output file with a name of the style *prefix*`o.concentration.`**suffix**, where suffix is a 3-digit number identifying the output file. Since it is applied to the current media type you can use, for example, `prefixo.concentration.003` for the porous media and `prefixo.concentration_overland.003` for the surface domain. This can be useful, for example, when a run crashes (since you don't have to start simulation from scratch).

• • •

Initial immobile concentration from output file

1. **fname** Name of the file which contains the initial immobile zone concentration data.

As above for the instruction Initial concentration from output file except immobile zone initial concentrations are assigned. This insures that the mobile and immobile zones will be in equilibrium at the start of a simulation if the same file concentration output file is used to define the initial conditions.

$$\bullet \bullet \bullet$$

Initial concentration from file

1. **fname** Name of the file which contains the initial concentration data.

All nodes in the currently active media (see Section 5.8.1) are assigned an initial concentration value which is read from a free-format ascii file. The concentrations must be listed in the file in order from node 1 to node NN. Each line can contain one or more values.

$$\bullet \bullet \bullet$$

Initial concentration for zones

1. **izone, czone...end** Zone number, initial concentration.

This instruction takes as input a list of zone numbers and initial concentrations. Nodes that are in the currently active media (see Section 5.8.1), and that are also in elements whose zone number is in the list will be assigned the listed initial concentration value. All other nodes will be assigned an initial concentration of zero. If a node straddles the boundary between one or more zones, an average initial concentration value is assigned.

$$\bullet \bullet \bullet$$

Initial immobile zone concentration

1. **val** Initial concentration [M L^{-3}].

All nodes are assigned an initial concentration value for the first solute only.

$$\bullet \bullet \bullet$$

Initial immobile zone concentration from file

1. **fname** Name of the file which contains the initial immobile zone concentration data.

All nodes are assigned an initial concentration value, for the first solute only, which is read from a free-format ascii file. The concentrations must be listed in the file in order from

node 1 to node NN. Each line can contain one or more values.

<center>● ● ●</center>

Restart file for concentrations

1. **transport_restart_file_name** Name of the file which contains the results of the previous transport solution.

All nodes are assigned an initial concentration value which is read from the file. This allows concentrations from a previous run to be used 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.

Since, by default, concentration values for the last time step are output in BINARY format to a file with the suffix *prefix*o.cen, these are always available for restarting a transport run. It is recommended that this file be renamed in order to avoid accidentally overwriting it and changing the restart conditions for a later run.

<center>● ● ●</center>

5.6.3.1 Solute Definition

These instructions can be used to add a new solute (i.e. species) to the system. **HydroGeoSphere** 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 of a straight decay chain see Section 4.4.1. 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.

Note that a solute can have different values for the decay constant and distribution coefficient (retardation factor for fractured media) in porous, dual or fractured media or from zone to zone in a single medium.

Solute
Causes **grok** to begin reading a group of solute definition instructions until it encounters an End instruction.

<center>● ● ●</center>

The available instructions are:

Name

1. **spname** Solute name.

Changes the solute name, which defaults to **Species n**, where n is the current solute number.

• • •

Free-solution diffusion coefficient

1. **diffrac** Free-solution diffusion coefficient [$L^2 \; T^{-1}$], D_{free} in Equation 2.81.

Assigns a new value for the free-solution diffusion coefficient, which defaults to 0.0 (zero).

• • •

Parents

1. **npa** Number of parent species for the current species. If the current species has one or more parents, enter the following **npa** times (i.e. once for each parent):

    (a) **kparen, aparen** Parent species number and the mass fraction.

Assigns a value for the number of parent species, which defaults to 0. The mass fraction is a number between 0 and 1 which defines how much of the parent species transforms into the daughter species (i.e. the current species).

• • •

The following parameters affect porous media solute properties:

Decay constant

1. **clambda** First-order decay constant [T^{-1}], λ in Equation 2.79.

Assigns a uniform value for the solute first-order decay constant for all porous media zones in the domain. The default value is 0.0 (no decay).

• • •

Zoned decay constant

1. **clambda(i,j),j=1,nzones_prop** First-order decay constant [T^{-1}] for species **i** for each porous media zone **j**, λ in Equation 2.79.

Assigns a unique value for the solute first-order decay constant to each porous media zone in the domain. The default value is 0.0 (no decay).

• • •

Distribution coefficient

1. **dkd** Distribution coefficient, K' in Equation 2.80.

Assigns a uniform value for the solute distribution coefficient for all porous media zones in the domain. The default value is 0.0 (no attenuation).

• • •

Zoned distribution coefficient

1. **dkd(i,j),j=1,nzones_prop** Distribution coefficient for species **i** for each porous media zone **j**, K' in Equation 2.80.

Assigns a unique value for the distribution coefficient to each porous media zone in the domain. The default value is 0.0 (no attenuation).

• • •

Dual decay constant

1. **clambda** First-order decay constant $[\text{T}^{-1}]$, λ_d in Equation 2.87.

Assigns a uniform value for the solute first-order decay constant for all dual continuua zones in the domain. The default value is 0.0 (no decay).

• • •

Zoned dual decay constant

1. **clambda(i,j),j=1,nzones_prop** First-order decay constant $[\text{T}^{-1}]$ for species **i** for each dual continuua zone **j**, λ_d in Equation 2.87.

Assigns a unique value for the solute first-order decay constant to each zone in the domain. The default value is 0.0 (no decay).

• • •

Dual distribution coefficient

1. **dkd** Distribution coefficient, K'_d in Equation 2.88.

Assigns a uniform value for the solute distribution coefficient for all dual continuua zones in the domain. The default value is 0.0 (no attenuation).

• • •

Zoned dual distribution coefficient

1. **dkd(i,j),j=1,nzones_prop** Distribution coefficient for species **i** for each dual continuua zone **j**, K'_d in Equation 2.88.

Assigns a unique value for the distribution coefficient to each dual continuua zone in the domain. The default value is 0.0 (no attenuation).

● ● ●

The following parameters affect fractured media solute properties:

Fracture Decay constant

1. **clambda_f** First-order decay constant $[\text{T}^{-1}]$, λ_f in Equation 2.83.

Assigns a uniform value for the solute first-order decay constant for all discrete fracture zones in the domain. The default value is 0.0 (no decay).

● ● ●

Zoned fracture decay constant

1. **clambda_f(i,j),j=1,nzones_prop** First-order decay constant $[\text{T}^{-1}]$ for species **i** for each discrete fracture zone **j**, λ_f in Equation 2.83.

Assigns a unique value for the solute first-order decay constant to each discrete fracture zone in the domain. The default value is 0.0 (no decay).

● ● ●

Fracture retardation factor

1. **rfrac** Fracture retardation factor, R_f in Equation 2.84.

Assigns a uniform value for the fracture retardation factor for all discrete fracture zones in the domain. The default value is 1.0 (no attenuation).

● ● ●

Zoned fracture retardation factor

1. **rfrac(i,j),j=1,nzones_prop** retardation factor for species **i** for each discrete fracture zone **j**, R_f in Equation 2.84.

Assigns a unique value for the fracture retardation factor to each discrete fracture zone in the domain. The default value is 1.0 (no attenuation).

● ● ●

The following parameters affect overland media solute properties:

Overland Decay constant

1. **clambda_o** First-order decay constant $[\text{T}^{-1}]$, λ_f in Equation 2.94.

Assigns a uniform value for the solute first-order decay constant for all overland flow zones in the domain. The default value is 0.0 (no decay).

<div align="center">• • •</div>

Zoned overland decay constant

1. **clambda_o(i,j),j=1,nzones_prop** First-order decay constant $[\mathrm{T}^{-1}]$ for species **i** for each overland flow zone **j**, λ_f in Equation 2.83.

Assigns a unique value for the solute first-order decay constant to each overland flow zone in the domain. The default value is 0.0 (no decay).

<div align="center">• • •</div>

Overland retardation factor

1. **rolf** Overland flow retardation factor, R_o in Equation 2.84.

Assigns a uniform value for the overland retardation factor for all overland zones in the domain. The default value is 1.0 (no attenuation).

<div align="center">• • •</div>

Zoned overland retardation factor

1. **rfrac(i,j),j=1,nzones_prop** retardation factor for species **i** for each overland flow zone **j**, R_o (similar to R_f in Equation 2.84).

Assigns a unique value for the overland flow retardation factor to each overland zone in the domain. The default value is 1.0 (no attenuation).

<div align="center">• • •</div>

The following instructions can be used to identify certain species. This is especially important to calculate fluid density and viscosity (for variable-density transport) from individual species concentrations and temperature. Note that fluid temperature is treated as a mobile species.

Sodium species
The presently defined species is identified as sodium, Na^+.

<div align="center">• • •</div>

The following instructions can be used likewise to identify other species:

> Potassium species
> Calcium species
> Magnesium species
> Chloride species

> Sulphate species
> Hydrogencarbonate species
> Carbonate species
> Salt mass fraction
> Temperature species

By default, no species impacts fluid density or viscosity. This default can be changed with the following instruction:

Affects fluid properties

With this instruction, the presently defined species has an impact on both fluid density and viscosity. This instruction can only be applied to the following species: Na^+, K^+, Ca^{2+}, Mg^{2+}, Cl^-, SO_4^{2-}, CO_3^{2-} and HCO_3^-, fluid temperature T and salt mass fraction smf. Note that, for the moment, if salt mass fraction affects fluid properties, no other species can impact fluid properties.

• • •

Note that instructions like 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 distribution coefficient definitions for all types of media. 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 (e.g. make fractures, specified concentration, specified third-type concentration etc. which depend on it should be placed after it in the *prefix*.grok file.

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
```

An example of a more complex system with two solutes and 7 material zones is shown in Figure 5.9 for the first solute, called DCB, which only decays in zone 1, and has distribution coefficients which vary from zone to zone. Figure 5.10 shows how to define the 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.

5.6.3.2 Travel Time Probability

Find zero age zones

HydroGeoSphere will automatically find the inlet (or outlet for the backward problem)

```
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 5.9: 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                 ! i.e. 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 5.10: Definition of a Daughter Solute With Zoned Properties.

nodes and assign the proper initial condition (or boundary condition for the moment equations). This option is valid for the case of age/life expectancy computations at aquifer scale, in which case each node belonging to the inlet/outlet zone is to be assigned a zero age/life expectancy condition.

<div align="center">• • •</div>

Zero travel time
Assigns a zero travel time condition for the chosen nodes.

<div align="center">• • •</div>

5.6.3.3   Heat Transfer

To enable heat transfer, you must define the temperature species in the solute definition block as described in Chapter 5.6.3.1. Note that the unit for temperature must be °C and that all heat transfer properties must be given in the SI-units kilogram-meter-second-Celsius. The thermal properties of the solids are specified in the material property file. The following instructions can be used to control the heat transfer solution:

Do heat transfer...End
Causes **grok** to begin reading a group of heat transfer instructions until it encounters an End instruction.

<div align="center">• • •</div>

The available instructions are:

Thermal conductivity of water
   1. **k_l** [W L^{-1} K^{-1}] Thermal conductivity of the liquid phase.

Assigns a uniform value to the thermal conductivity of groundwater. If this instructions is used, the thermal conductivity of water is assumed constant and equal to k_l. It is therefore not calculated from the water temperature, which is the default setting.

<div align="center">• • •</div>

Specific heat capacity of water
   1. **c_l** [J kg^{-1} K^{-1}] Specific heat capacity of the liquid phase.

Assigns a uniform value to the specific heat capacity of water. If this instructions is used, the specific heat capacity of water is assumed constant and equal to c_l. It is therefore not calculated from the water temperature, which is the default setting.

<div align="center">• • •</div>

Mechanical heat dispersion

Causes mechanical heat dispersion to be simulated. The default is no mechanical heat dispersion.

• • •

The following instruction can be used to define initial temperatures for the problem:

Initial temperature profile

1. **temp_top** [°C] The temperature at the top of the domain.

2. **temp_grad** [K m^{-1}] The prevailing geothermal gradient.

Calculates a depth profile of temperature and assigns the values to the initial temperature.

• • •

The following instructions can be used to define the heat source boundary condition:

Zero order source

1. **npanel** Number of panels in the time-variable, zero-order source function. For each panel, enter the following:

   (a) **ton_val,toff_val, (bc_val(j),j=1,nspeciesmob)** Time on [T], time off [T] and mass of solute produced per unit volume of porous medium solids per unit time [M L^{-3}].

Nodes in the chosen zones area assigned zero-order source boundary conditions.

A panel is a point in time at which the source term is set to a new value. The first panel would normally start at time zero. The source term given for the last panel will be maintained until the end of the simulation. You can assign a static source term for the duration of the simulation by setting **npanel** to 1, **ton_val** to 0.0 and **toff_val** to a large number.

Note that if **nspeciesmob** is greater than 1, additional values of **bc_val** should be included.

• • •

Exponential zero order source

1. **heat_q_zero** Heat production at t=0.

2. **heat_constant** Constant in the exponential function.

Nodes in the chosen zones area assigned an exponentially decreasing zero-order heat source boundary conditions.

• • •

5.7   Boundary Conditions

5.7.1   General considerations

The boundary condition input routines have been completely rewritten in this version of **HydroGeoSphere**. The old routines were developed over the years by many different developers and for a host of reasons. There was a lack of continuity between the routines that has made it increasingly difficult to update and maintain **HydroGeoSphere**. With this in mind, a more well defined data structure has been developed for defining boundary conditions and, where possible, the functionality of the old routines has been incorporated. For the end user, this means having to learn a new approach to defining boundary conditions, but one that we hope will be more logical and therefore easier to understand and apply.

In it's simplest form, a boundary condition is defined by a value that is associated with a node, for example, a specified head. In some cases, such as a well that has a variable pumping rate, the value may change temporally. If the well were turned off abruptly then started up again at a later time, the value would not be applied continuously. Some inputs are not defined by a single value, but rather by a 2 or 3-dimensional field. For example, rainfall may be given in the form of 2-D raster data defined in a certain region. Other inputs might be in the form of tables of values at defined locations, which is often the case when we incorporate information from another model. As you can see, we need a combination of data structure and input format that are general, yet flexible.

All flow boundary conditions are now defined using the following instruction:

Boundary condition...End

grok reads instructions that define a new boundary condition until it encounters an End instruction.

<div align="center">●●●</div>

Before issuing it, you need to specify which domain to apply the condition to using the use domain type instruction e.g.:

```
use domain type
porous media
```

You also need to choose the nodes or faces to apply the condition to, and save them as a named set, as outlined in section 5.4. For example, here we choose nodes having X-coordinate 0.0 and save them in the node set named 'inflow'.

```
clear chosen nodes
choose nodes x plane
0.0
1.e-5
```

```
    create node set
    inflow
```

Now we can define the Boundary condition...End group which should contain, at a minimum, input that defines the boundary condition type and the name of the mesh subset to which it applies. In many cases, a time-value table is also supplied. For example, given the domain and node set information defined above, the following list of instructions defines a boundary condition of type head (i.e. specified head) for the nodes in the porous medium contained in the node set 'inflow':

```
    boundary condition
        type
        head

        node set
        inflow

        time value table
        0.0     100.0
        end

    end ! new specified head
```

This example also uses the time value table instruction to define the head value of 100.0 at time 0.0.

As you can see from the example, the Boundary condition...End group contains instructions which define the nature of the new boundary condition. We will now discuss those instructions in more detail.

5.7.1.1   Type

To define what type of boundary condition (e.g. head, flux etc.) is to be applied use:

Type

1. **bc_type** The type of boundary condition to be applied.

The various allowable boundary condition types are described below in more detail and in relation to the domains where they may be applied.

● ● ●

5.7.1.2   Node and Face Sets

Some boundary condition types are applied to nodes (e.g. head) while others are applied to faces (e.g. flux). To define the group of nodes use:

Node set

1. **bc_set_name** Name of the node set to apply the boundary condition to.

● ● ●

To define the group of faces use:

Face set

1. **bc_set_name** Name of the face set to apply the boundary condition to.

● ● ●

5.7.1.3   Time-varying Inputs

To define more complex spatial and temporal properties for a boundary condition use:

Time value table

1. **bc_time(i), bc_val(i)...end** The time and boundary condition value list.

At time **bc_time(i)** the value **bc_val(i)** is applied and maintained until time **bc_time(i+1)**. The last value entered in the list will be applied until the end of the simulation.

● ● ●

Time raster table

1. **bc_time(i), bc_raster(i)...end** The time and raster file name list.

At time **bc_time(i)** the raster file **bc_raster(i)** is applied and maintained until time **bc_time(i+1)**. The last raster file entered in the list will be applied until the end of the simulation. **HydroGeoSphere** uses the node (or face centroid) xy-coordinate for each member of the set to interpolate a value for the boundary condition at that point.

● ● ●

The following instructions use the same input data structure except they are applied to xz or yz coordinates:

```
time raster xz table
time raster yz table
```

Time file table

1. **bc_time(i), bc_file(i)...end** The time and file name list.

At time **bc_time(i)** the values read from file **bc_file(i)** are applied and maintained until time **bc_time(i+1)**. The data from the last file entered in the list will be applied until the end of the simulation. **HydroGeoSphere** reads a list of values from the file and assigns them in order to the current set of nodes or faces, depending on what type of boundary condition is being applied. The number of values in the file must match the number of nodes or faces in the set or **grok** will stop with an error message.

● ● ●

This instruction can be used in conjunction with other instructions to create files of nodal values and then read them in to define boundary conditions. For example, these instructions:

```
clear chosen nodes
choose nodes x plane
0.0
1.e-5

nodal function z to file
    inflow.txt
    ! Z    value
     0.0   0.
     3.    10.
     10.    4.
end

create node set
inflow
```

create a file `inflow.txt` which contains a value for each currently chosen node. It uses the nodal z-coordinate to interpolate the value from the tabulated function of z versus value. We can now use the file like this:

```
boundary condition
    type
    head
```

```
        node set
        inflow

        time file table
        0.0   inflow.txt
        11.   none
        end

    end ! new specified head
```

Note that we use the node set inflow in conjunction with the file inflow.txt. Any mismatch in the number of nodes bwtween the node set file and the data file will cause **grok** to stop with an error message.

5.7.1.4   Interpolation

Time-varying boundary conditions change abruptly from value **bc_val(i)** to value **bc_val(i+1)** at time **bc_time(i+1)**. To have **HydroGeoSphere** use linear interpolation to smooth the values used between time panel endpoints use:

Interpolate

This command causes time-varying values to be interpolated between panel values for the boundary condition currently being defined. This results in a smoother application of the time-varying function.

<p align="center">● ● ●</p>

5.7.1.5   Intermittent Conditions

Boundary conditions can be turned on and off by assigning special **NODATA** values instead of normal input in the time-value, time-raster and time-file tables. For example, the default **NODATA** value is -99999., and the following table:

```
    boundary condition
        type
        head

        ... etc

        time value table
           0.0        100.0
```

```
    1000.0     -99999.
    2000.0       100.0
    end

  end ! new specified head
```

would apply the specified head of 100 from time 0.0 to time 1000.0 and then allow the node to revert to an unconstrained condition until time 2000.0, when a specified head of 100.0 would again be applied for the duration of the simulation.

For rasters and files, the default **NODATA** value is the string 'none'.

If for some reason you need to change the default **NODATA** value you can use:

Nodata value

1. **nodata_value** The value that indicates there is no data.

<p style="text-align:center">● ● ●</p>

The following instructions are used to change the **NODATA** values used for rasters a files :

```
    nodata raster
    nodata file
```

5.7.1.6   Nodal Flux Head Constraints

The old instructions Specified volumetric flowrate and Specified volumetric flowrate, head constrained are no longer used, as they are covered by the boundary condition type Nodal flux described below. If you want to force the nodal flux to operate within a specified hydraulic head range, you can do so using the following instruction:

Nodal flux head constraints

1. **bc_constraint_1, bc_constraint_2** Minimum and maximum constraining head values for positive nodal flux (i.e. inflow).

2. **bc_constraint_3, bc_constraint_4** Minimum and maximum constraining head values for negative nodal flux (i.e. outflow).

If the nodal flux is positive, and the head at the node is less than **bc_constraint_1**, then the flux will be applied until such time as the head at the node exceeds **bc_constraint_2**. Once the head at the node exceeds **bc_constraint_2**, then the flowrate will be set to zero until such time as the head at the node drops below **bc_constraint_1**.

If the flowrate is negative, and the head at the node is greater than **bc_constraint_3**, then the flowrate will be applied until such time as the head at the node drops below **bc_constraint_4**. Once the head at the node drops below **bc_constraint_4**, then the flowrate will be set to zero until such time as the head at the node exceeds **bc_constraint_3**.

• • •

5.7.1.7   Tecplot Output

Tecplot-formatted output files can be created for a specific boundary condition using this instruction:

Tecplot output

This command causes node specific information about the boundary condition to be written at each output time. The results are written to a file called *prefix*o.bc.*bcname*.dat

• • •

5.7.2   Subsurface Flow

5.7.2.1   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 define the subset of nodes for which you want to apply the condition and write them to a nodal data set.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign various specified head boundary conditions:

Head
Sets the input type to be a general specified head boundary condition.

• • •

For example:

```
boundary condition
    type
    head

    node set
    inflow

    time value table
```

```
        0.0     100.0
        end


    end ! new specified head
```

would define a specified head of 100.0 from time zero for the duration of the simulation for all of the nodes contained in the node set `inflow`.

These heads can be interpolated or turned on and off as discussed in Section 5.7.1.

Head equals elevation

Sets the input type to be a special form of the specified head boundary condition where head is set equal to the elevation of the node. This is normally the z-coordinate, unless the instruction y vertical has been issued, in which case the y-coordinate is used instead.

<div align="center">● ● ●</div>

For example:

```
    boundary condition
        type
        head equals elevation

        node set
        inflow

    end ! new specified head
```

This example shows that a time-value table is not required, since heads are derived from the nodal elevation. However, if you wish to turn the boundary condition on or off, then this can be accomplished by including a time value table instruction:

```
    boundary condition
        type
        head equals elevation

        node set
        inflow

        time value table
        0.0    1.0
        10.0   -99999.
        end

    end ! new specified head
```

The value 1.0 at time zero is ignored, but the **NODATA** value -99999 at time 10.0 causes the boundary conditon to be turned off and the nodes become unconstrained.

Head equals initial

Sets the input type to be a special form of the specified head boundary condition where head is set equal to the initial head at the node.

<div align="center">• • •</div>

For example:

```
boundary condition
    type
    head equals initial

    node set
    inflow

end ! new specified head
```

This condition can be turned on and off as discussed above for Head equals elevation.

Each time you issue a Boundary condition...End instruction, a new boundary condition is formed with a unique ID number. **HydroGeoSphere** processes the boundary conditions in order and later ones may take precedence over later ones. The position in the grok file determines the order.

rgm Think about whether head bc override flux bc's because we call the routine to process all heads first, then do fluxes later. If the node was assigned a specified head or fluid flux value by a previous instruction then it might be overwritten by later head boundary conditions, depending on whether or not both nodes are active at a given time.

Note that the definition of a seepage face (see Section 5.7.2.3) may lead to the formation of a specified head boundary condition at the seepage nodes.

5.7.2.2   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 normally be part of the outer boundary of the grid.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign various specified flux boundary conditions:

Flux

Sets the input type to be a general specified flux boundary condition and converts fluxes

to nodal volumetric fluxes [L^3 T^{-1}] by multiplying by the contributing area of the chosen face.

•••

For example:

```
boundary condition
    type
    flux

    face set
    top

    time raster table
    0.0     recharge.asc
    end

end ! new specified head
```

would define a specified flux that is mapped from the raster file recharge.asc and is applied from time zero for the duration of the simulation for all of the nodes contained in the face set `top`.

Rain
Sets the input type to be a general specified flux boundary condition and converts fluxes to nodal volumetric fluxes [L^3 T^{-1}] by multiplying by the contributing area of the chosen face when projected onto the xy plane.

•••

Flux nodal
Sets the input type to be a general specified flux boundary condition and treats the fluxes as nodal volumetric fluxes [L^3 T^{-1}] which are applied directly to nodes in the given node set.

•••

For example:

```
boundary condition
    type
    flux nodal

    node set
    top

    time file table
```

```
        0.0     nflux.txt
        end


    end ! new specified head
```

would assign a specified nodal flux that is read from the input file nflux.txt and is applied from time zero for the duration of the simulation for all of the nodes contained in the node set `top`. Note that the number of entries in the file must be the same as the number of nodes in the set.

These fluxes can be interpolated or turned on and off as discussed in Section 5.7.1.

In cases where flux boundary inputs overlap, fluid fluxes will be accumulated.

Note that the definition of wells or tile drains (see Section 5.8.2.4) in the problem may create a non-zero specified flux boundary condition.

5.7.2.3 Seepage

A seepage boundary condition can be applied to any node, but they are usually restricted to those on the surface of the domain. If the hydraulic head at a seepage node rises above its elevation, it becomes a specified head node, and the pressure head is set to zero and water is allowed to flow out. If, subsequently, the hydraulic head drops below the node elevation, then it reverts to an unconstrained state.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign seepage boundary conditions:

Seepage
Sets the input type to be a seepage boundary condition.

<p align="center">● ● ●</p>

For example:

```
    boundary condition
        type
        seepage

        node set
        outflow


    end ! new specified head
```

This example shows that a time-value table is not mandatory, since a pressure head of zero is applied if the nodal head is above the nodal elevation. However, if you wish to turn the seepage boundary condition on or off, then this can be accomplished by including a time value table instruction:

```
boundary condition
    type
    seepage

    node set
    outflow

    time value table
    0.0    1.0
    10.0   -99999.
    end

end ! new specified head
```

The value 1.0 at time zero is ignored, but the **NODATA** value -99999 at time 10.0 causes the boundary conditon to be turned off and the nodes become unconstrained.

rgm Old text read: 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.*

The overall rate of fluid outflow for all seepage face nodes is given in the mass balance output sections of the *prefix*o.lst file.

Details of the seepage node calculations, including which nodes are currently acting as seepage nodes and the fluid flux exiting the domain at each active seepage node can be written to the .lst file by using the run-time debug utility instruction Write seepage face output to .lst file described in Section C.

5.7.2.4 Free Drainage

Assigns a free drainage boundary condition, as described in Section 3.6.1 to nodes on all faces in the specified set. These faces should be on the surface of the domain.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign free drainage boundary conditions:

Free drainage
Sets the input type to be a free drainage boundary condition.

<div align="center">● ● ●</div>

For example:

```
boundary condition
```

```
        type
        free drainage

        face set
        outflow

    end
```

This example shows that a time-value table is not mandatory. However, if you wish to turn the free drainage boundary condition on or off, then this can be accomplished by including a time value table instruction:

```
    boundary condition
        type
        free drainage

        face set
        outflow

        time value table
        0.0    1.0
        10.0   -99999.
        end

    end
```

The value 1.0 at time zero is ignored, but the **NODATA** value -99999 at time 10.0 causes the boundary condition to be turned off and the nodes become unconstrained.

5.7.2.5   Potential Evapotranspiration

This is a special form of the second-type, Neumann, specified or constant flux boundary condition which is used in conjunction with evapotranspiration properties which can be defined as described in Section 5.8.5. 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 top boundary of the grid, and in this case the top boundary must be coincident with the 2-D slice which was used to define the 3-D mesh. This means that you cannot use the boundary condition if the Y-vertical instruction has been used. This restriction arises because of grid numbering assumptions which are made when applying evapotranspiration as a function of depth.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign various specified flux boundary conditions:

Potential Evapotranspiration

Sets the input type to be a special specified flux boundary condition and converts the evaporative fluxes to nodal volumetric fluxes $[L^3 \ T^{-1}]$ by multiplying by the contributing area of the chosen face.

<div align="center">• • •</div>

For example:

```
boundary condition
    type
    potential evapotranspiration

    face set
    top

    time value table
    0.0     0.001
    end

end ! new specified head
```

would define a potential evapotranspiration of 0.001 from time zero for the duration of the simulation for all of the nodes contained in the face set `top`.

Although this instruction is currently restricted to be applied to porous media only, the potential evapotranspiration is applied to the surface water domain and subsurface domain in a stepwise manner. Nodes in the chosen faces and as a function of depth may be assigned a time-variable flux value, depending on factors such as the current nodal saturation, which may inhibit for example, transpiration from occurring.

5.7.2.6 Specified Flowrate

This is a special case of the second-type, Neumann, specified or constant flux boundary condition. It is a nodal property and so you should first choose the subset of nodes for which you want to apply the condition.

If the node was assigned a specified head or fluid flux by a previous instruction then it will not be modified by specified flowrate instructions.

Specified volumetric flowrate

1. **npanel** Number of panels in the time-variable volumetric flowrate. For each panel, enter the following:

    (a) **ton_val, bc_val** Time on [T] and specified volumetric flowrate $[L^3 \ T^{-1}]$.

Chosen nodes in the currently active media (see Section 5.8.1) are assigned a time-variable flowrate value. A panel is a point in time at which the specified flowrate is set to a new

value. The first panel would normally start at time zero. The flowrate given for the last panel will be maintained until the end of the simulation. You can assign a static flowrate for the duration of the simulation by setting **npanel** to 1 and **ton_val** to 0.0.

Nodal volumetric flowrates $[L^3\ T^{-1}]$ are applied directly to the chosen nodes.

• • •

5.7.2.7   River Flux

NOTE: not tested or fully implemented, but will be something like this

Assigns a river flux boundary condition, as described in Section 3.6.1 to nodes in the specified set. These nodes are normally located on the surface of the domain. For river flux nodes, water may flow in or out of the domain depending on the difference in head between the river node and the specified river head value.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign a river flux boundary condition:

Simple river
Sets the input type to be a river flux drainage boundary condition.

• • •

For example:

```
boundary condition
    type
    simple river

    node set
    my_river

    time value table
    0.0     1.e-5     25.
    end

end
```

This example shows the use of a time value table instruction to define the river conductance (1.e-5) and river head (25.) values. This would not be very useful in most cases, since conductance and head vary from node to node along a river. In such cases, the inputs would be defined by a time file table, with unique values given for each node:

```
boundary condition
    type
```

```
        simple river

        node set
        my_river

        time file table
            0.0    my_river.lst
        1000.      none
        end


    end
```

Here, the river flux boundary condition would become inactive after time 1000 and the nodes would become unconstrained. In this case, the file `my_river.lst` would contain two values per line, the river conductance and head value, and one line for each node in the set.

NOTE: In the old instruction for defining river fluxes, there were inputs for bed thickness, bed conductance, reach length and reach width which were then used to compute a river conductance term. Now, it is left up to the user to define the conductance term in any way they see fit, and to write the final conductance value into the input file.

5.7.2.8 Drain Flux

NOTE: not tested or fully implemented, but will be something like this

Assigns a drain flux boundary condition, as described in Section 3.6.1 to nodes in the specified set. These nodes are normally located on the surface of the domain. The fundamental difference between river and drain flux nodes is that drain nodes only allow water to flow out of the system, depending on the difference in head between the drain node and the specified drain head value.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign a drain flux boundary condition:

Simple drain
Sets the input type to be a drain flux drainage boundary condition.

• • •

For example:

```
    boundary condition
        type
        simple drain

        node set
```

```
        my_drain

        time value table
        0.0      1.e-5      25.
        end


    end
```

This example shows the use of a time value table instruction to define the drain conductance (1.e-5) and drain head (25.) values. This would not be very useful in most cases, since conductance and head would vary from drain node to drain node. In such cases, the inputs would be defined by a time file table, with unique values given for each node:

```
    boundary condition
        type
        simple drain

        node set
        my_drain

        time file table
           0.0    my_drain.lst
        1000.      none
        end


    end
```

Here, the drain flux boundary condition would become inactive after time 1000 and the nodes would become unconstrained. In this case, the file my_drain.lst would contain two values per line, the drain conductance and head value, and one line for each node in the set.

5.7.2.9   Surface loading

It is a nodal property, but because surface loading is applied to all nodes of a vertical column of nodes, only one node of the column needs to be chosen.

Specified stress variation

1. **npanel** Number of panels in the time-variable specified stress function. For each panel, enter the following:

   (a) **ton_val, bc_val** Time on [T] and specified stress variation [L T^{-1}].

Chosen nodes in the currently active media (see Section 5.8.1) are assigned a time-variable specified stress variation value. The specified stress variation value corresponds to the term

$$\frac{\partial \left(\sigma_{zz}/\rho g\right)}{\partial t} \tag{5.10}$$

which has units of $[\text{L T}^{-1}]$ and corresponds to an equivalent freshwater head change per unit time.

• • •

Specified stress variation from file

1. **fname** Name of the file which contains the list of nodes to which specified stress variation will be assigned. It should contain the following data for each node:

2. **nde, npanel** Node number, number of panels

   (a) **ton_val, bc_val** Time on $[\text{T}]$, stress variation $[\text{L T}^{-1}]$.

Nodes listed in the file and in the currently active media (see Section 5.8.1) are assigned a unique stress variation value. The specified stress variation value is described in the previous command.

• • •

Echo 1d loading conditions
If this command is included in the input file, a listing of the loading conditions will be issued in the *prefixo*.eco file.

• • •

5.7.2.10 Hydromechanical Stress

Hydromechanical stresses can be computed externally and used as input to **grok** using the following instruction:

Elemental stress field from files

1. **npanel** Number of panels (i.e. files) defining the time-variable external stress field For each panel, enter the following:

   (a) **ton_val, esvfile** Time on $[\text{T}]$ and name of file containing elemental stress stress values $[\text{L}]$.

This information is passed to **HydroGeoSphere**. At each timestep, the current time is compared to the array **ton_val** to determine whether or not to apply the update the current set of elemental stress values by reading the next file in the list.

Each file must be in ASCII format and contain a list of elemental stress values (i.e. one value for each element in the 3-D domain), expressed as equivalent freshwater head.

● ● ●

5.7.3   Surface Flow

You can assign a critical depth or zero depth gradient boundary condition, as described in Section 3.6.2 to segments in the specified set. These segments should be part of the overland flow domain and are typically located on the outer boundary.

The following instructions can be used as input to the Type instruction inside the Boundary condition...End instruction group to assign critical depth boundary conditions:

Critical depth
Sets the input type to be a critical depth boundary condition.

● ● ●

For example:

```
boundary condition
    type
    critical depth

    segment set
    outflow

end
```

This example shows that a time-value table is not mandatory. However, if you wish to turn the critical depth boundary condition on or off, then this can be accomplished by including a time value table instruction:

```
boundary condition
    type
    critical depth

    segment set
    outflow

    time value table
    0.0    1.0
    10.0   -99999.
    end

end
```

The value 1.0 at time zero is ignored, but the **NODATA** value -99999 at time 10.0 causes the boundary conditon to be turned off and the nodes become unconstrained.

Zero-depth gradient
Sets the input type to be a zero-depth gradient boundary boundary condition.

● ● ●

For example:

```
boundary condition
    type
    zero-depth gradient

    segment set
    outflow

end
```

If you wish to turn the zero-depth gradient boundary condition on or off, then this can be accomplished by including a `time value table` instruction.

NOTE: zero-depth gradient boundaries used to include variables defining the direction of boundary gradient and the bed slope at the boundary.

5.7.4   Transport

There are three basic options available for assigning boundary conditions to the transport solution. These are to specify either first-type (concentration), second-type (mass flux, concentration gradient), or third-type (Cauchy) at a node. Although these are typically applied to nodes located on the surface of the domain, first- and second-type boundary conditions can can also be applied to internal nodes.

Once they are defined, you can check the transport boundary conditions using the following instruction, which causes the current configuration to be written to the *prefix*o.eco file:

Echo transport boundary conditions
Causes the current boundary conditions to be written to the *prefix*o.eco file.

● ● ●

5.7.4.1   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

1. **npanel** Number of panels in the time-variable concentration function. For each panel, enter the following:

    (a) **ton_val,toff_val,(bc_val(j),j=1,nspeciesmob)** Time on [T], time off [T], and specified concentration [M L^{-3}] of each species.

Chosen nodes in the currently active media (see Section 5.8.1) are assigned a time-variable concentration value. If a node was previously assigned a specified concentration, it will remain in effect.

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. You can assign a static concentration for the duration of the simulation by setting **npanel** to 1, **ton_val** to 0.0 and **toff_val** to a large number.

Note that if **nspeciesmob** is greater than 1, additional values of **bc_val** should be included.

● ● ●

Specified concentration from file

1. **fname** Name of the file which contains the time-varying concentration information.

This file should contain the following input data:

1. **nnde** Number of nodes to be assigned concentration data.

2. **npanel** Number of panels in the time-variable concentration function. For each panel, enter the following:

    (a) **ton_val,toff_val** Time on [T], time off [T].

3. **nde, (bc_val(j),j=1,nspeciesmob)** Node number and specified concentration [M L^{-3}] of each species.

Listed nodes are assigned the time-variable concentration values contained in the file. If a node was previously assigned a specified concentration it will remain in effect.

Although all nodes in the file share the same time on/time off panel information the concentration values in each panel can vary from node to node.

● ● ●

Specified well concentration

1. **iw** Identification number of the well to which the concentration is to be applied. For each species defined, enter the following:

    (a) **ninjc** Number of panels in the injection concentration history for this well and species. For each panel enter the following:

        i. **cinjc,toninjc,toffinjc** Injection concentration [M L^{-3}], time on [T] and time off [T].

All nodes in well **iw** are assigned a time-variable first-type concentration value, which is uniform along the length of the well.

Since this instruction loops over all defined species, you must supply input even if it is a zero concentration.

• • •

Specified tile concentration

1. **iw** Identification number of the tile drain to which the concentration is to be applied. For each species defined, enter the following:

    (a) **ntdc** Number of panels in the injection concentration history for this tile and species. For each panel enter the following:

        i. **c_tile,ton_c_tile,toff_c_tile** Injection concentration [M L^{-3}], time on [T] and time off [T].

All nodes in tile drain **iw** are assigned a time-variable first-type concentration value, which is uniform along the length of the tile.

Since this instruction loops over all defined species, you must supply input even if it is a zero concentration.

• • •

5.7.4.2 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.

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

1. **npanel** Number of panels in the time-variable mass flux function. For each panel, enter the following:

   (a) **ton_val,toff_val,(bc_val(j),j=1,nspeciesmob)** Time on [T], time off [T], and specified mass flux [M T^{-1}] of each species.

Chosen nodes in the currently active media (see Section 5.8.1) are assigned a time-variable mass flux value. This is a passive injection of solute mass which has no effect on the flow solution. If a node was previously assigned a first or third-type concentration, it will remain in effect.

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. You can assign a static 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.

Note that if **nspeciesmob** is greater than 1, additional values of **bc_val** should be included.

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.

• • •

Interpolate mass flux
This command causes time-varying mass fluxes to be interpolated between panel values. This results in a smoother application of the mass flux function.

• • •

5.7.4.3   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

1. **calcflux** A logical switch which determines how fluid fluxes are handled for this third-type boundary condition. If false, read the following:

   (a) **userflux** Fluid flux value [L/T].

2. **npanel** Number of panels in the time-variable concentration function. For each panel, enter the following:

   (a) **ton_val,toff_val, (bc_val(j),j=1,nspeciesmob)** Time on [T], time off [T], and specified concentration [M L^{-3}] of each species.

Chosen nodes in the currently active media (see Section 5.8.1) are assigned a time-variable third-type concentration value unless they were previously assigned a first-type concentration.

If the variable **calcflux** is true, nodal fluxes are calculated by **HydroGeoSphere** from the flow solution and used to calculate the third-type boundary condition. Only positive fluxes (i.e. flowing into domain) are used. If false, **HydroGeoSphere** reads a flux value **userflux** which is used instead.

A panel is a point in time at which the specified third-type 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. You can assign a static concentration for the duration of the simulation by setting **npanel** to 1, **ton_val** to 0.0 and **toff_val** to a large number.

Note that if **nspeciesmob** is greater than 1, additional values of **bc_val()** should be included.

• • •

Specified third-type concentration from file

1. **fname** Name of the file which contains the time-varying third-type concentration information.

This file should contain the following input data:

1. **nelem3** Number of elements.

2. **npanel** Number of panels in the time-variable concentration function.

3. **calcflux** A logical switch which determines how fluid fluxes are handled for this third-type boundary condition. If false, read the following:

   (a) **userflux** Fluid flux value [L/T].

For each panel, enter the following:

    (a) **ton_val,toff_val** Time on [T] and time off [T].

For each of the **nelem3** elements, enter the following:

1. **nel, n1, n2, n3, n4, (bc_val(i,j),i=1,npanel, j=1,nspeciesmob)** Element number, 4 node numbers defining face and third-type concentration [M L^{-3}] history of each species.

Faces defined by the 4 nodes in the listed elements in the currently active media (see Section 5.8.1) are assigned a time-variable third-type concentration value unless they were previously assigned a first-type concentration. It is the responsibility of the user to ensure that the nodes define a face that is on the exterior of the domain.

For 3-node faces, enter a value of zero for node **n4**.

If the variable **calcflux** is true, nodal fluxes are calculated by **HydroGeoSphere** from the flow solution and used to calculate the third-type boundary condition. Only positive fluxes (i.e. flowing into domain) are used. If false, **HydroGeoSphere** reads a flux value **userflux** which is used instead.

If **nspeciesmob** is greater than 1, additional values of **bc_val** should be included.

Although all elements in the file share the same time on/time off panel information the concentration values in each panel can vary from element to element.

$$\bullet \ \bullet \ \bullet$$

Specified third-type concentration from face file

1. **fname** Name of the file which contains the time-varying third-type concentration information.

This file should contain the following input data:

1. **nface3** Number of faces.

2. **npanel** Number of panels in the time-variable concentration function.

3. **calcflux** A logical switch which determines how fluid fluxes are handled for this third-type boundary condition. If false, read the following:

    (a) **userflux** Fluid flux value [L/T].

For each panel, enter the following:

    (a) **ton_val,toff_val** Time on [T] and time off [T].

For each of the **nface3** faces, enter the following:

1. **nfce, (bc_val(i,j),i=1,npanel, j=1,nspeciesmob)** Face number and third-type concentration [M L^{-3}] history of each species.

Listed faces in the currently active media (see Section 5.8.1) and which are located on the exterior of the domain are assigned a time-variable third-type concentration value unless they were previously assigned a first-type concentration.

If the variable **calcflux** is true, nodal fluxes are calculated by **HydroGeoSphere** from the flow solution and used to calculate the third-type boundary condition. Only positive fluxes (i.e. flowing into domain) are used. If false, **HydroGeoSphere** reads a flux value **userflux** which is used instead.

If **nspeciesmob** is greater than 1, additional values of **bc_val** should be included.

Although all faces in the file share the same time on/time off panel information the concentration values in each panel can vary from face to face.

• • •

5.7.4.4 Thermal Energy

Specified temperature flux

1. **ntime** Number of panels in the time-variable, specified temperature flux function. For each panel, enter the following:

   (a) **ton_val, toff_val, bc_val, bc_temp** Time on [T], time off [T], specified volume flux (of liquid) [L^3/T] multiplied by the heat capacity [J kg^{-1} K^{-1}] and density [M/L^3] of the boundary medium, temperature of water entering [K].

Nodes in both the chosen faces and the currently active media (see Section 5.8.1) are assigned a time-variable temperature flux value.

Although a fluid volume flux **bc_val** is specified, this does not influence the flow solution in any way. It is merely intended to give the user a straightforward way to input a known amount of energy to the system, as a function of fluid volume and temperature.

• • •

The following instructions are used to define the atmospheric inputs discussed in Section 2.5.1.10, and summarized in Equation 2.105.

Incoming Shortwave Radiation

1. **ntime** Number of panels in the time-variable, incoming shortwave radiation function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], incoming shortwave radiation [M/T^3], $K^{\downarrow}$ in Equation 2.106. Default value is 1.10×10^2 J/(m^2 s).

• • •

Sinusoidal Incoming Shortwave Radiation

1. **ntime** Number of panels in the time-variable, sinusoidal incoming shortwave radiation function. For each panel, enter the following:

   (a) **ton_val, value, amp, phase, period** Time on [T], vertical shift (mid-point incoming shortwave radiation, $K^\downarrow$ in Equation 2.106) [M/T^3], amplitude [M/T^3], phase, period [T].

• • •

Cloud Cover

1. **ntime** Number of panels in the time-variable, cloud cover function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], cloud cover [-], C_c in Equation 2.108. Default value is 0.50.

• • •

Incoming Longwave Radiation

1. **ntime** Number of panels in the time-variable, incoming longwave radiation function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], incoming longwave radiation [M/T^3], $L^\downarrow$ in Equation 2.109. Default value is 3.0×10^2 J/(m^2 s).

• • •

Temperature of Air

1. **ntime** Number of panels in the time-variable, air temperature function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], air temperature [K], T_a in Equation 2.110. Default value is 15°C.

• • •

Sinusoidal Temperature of Air

1. **ntime** Number of panels in the time-variable, sinusoidal air temperature function. For each panel, enter the following:

   (a) **ton_val, value, amp, phase, period** Time on [T], vertical shift (mid-point temperature, T_a in Equation 2.110) [K], amplitude [K], phase, period [T]

<div align="center">• • •</div>

These instructions are used to define the sensible heat flux Q_h in Equation 2.112 and latent heat flux Q_E in Equation 2.113.

Density of Air

1. **ntime** Number of panels in the time-variable, air density function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], density of air [M/L^3], ρ_a in Equation 2.112. Default value is 1.225 kg/m^3.

<div align="center">• • •</div>

Specific Heat of Air

1. **ntime** Number of panels in the time-variable, air specific heat function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], specific heat of air [L^2/(T^2 K)], c_a in Equation 2.112. Default value is 7.17×10^2 J/(kg ^{o}K).

<div align="center">• • •</div>

Wind Speed

1. **ntime** Number of panels in the time-variable, wind speed function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], wind speed [L/T], V_a in Equation 2.112. Default value is 1.0 m/s.

<div align="center">• • •</div>

Sinusoidal Wind Speed

1. **ntime** Number of panels in the time-variable, sinusoidal wind speed function. For each panel, enter the following:

   (a) **ton_val, value, amp, phase, period** Time on [T], vertical shift (mid-point wind speed, V_a in Equation 2.112) [L/T], amplitude [L/T], phase, period [T]

<div align="center">• • •</div>

Drag Coefficient

1. **ntime** Number of panels in the time-variable, drag coefficient function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], drag coefficient [-], c_D in Equation 2.112. Default value is 2.0×10^-3.

   • • •

Latent Heat of Vapourization

1. **ntime** Number of panels in the time-variable, latent heat of vapourization function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], latent heat of vapourization $[L^2/T^2]$, L_V in Equation 2.113. Default value is 2.258×10^6 J/kg.

   • • •

Specific Humidity of Air

1. **ntime** Number of panels in the time-variable, air specific hunidity function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], specific humidity of air [M/M], SH_a in Equation 2.113. Default value is 0.01062.

   • • •

Soil-Water Suction at Surface

1. **ntime** Number of panels in the time-variable, soil-water suction at surface function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], soil-water suction at surface [L], ψ_g in Equation 2.115 . Default value is 0.138 m.

   • • •

Saturation Vapour Pressure

1. **ntime** Number of panels in the time-variable, saturation vapour pressure function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], saturation vapour pressure $[M/(L\ T^2)]$, $e_{sat}[T_g]$ in Equation 2.116. Default value is 1.704×10^3 Pa.

   • • •

Relative Humidity

1. **ntime** Number of panels in the time-variable, relative humidity function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], relative humidity [-]. Default value is 0.75.

   • • •

Pressure of Air

1. **ntime** Number of panels in the time-variable, air pressure function. For each panel, enter the following:

   (a) **ton_val, value** Time on [T], air pressure [M/L T^2], p_a in Equation 2.116. Default value is 1.013×10^5 Pa.

   • • •

5.7.4.5   Imported From GMS

Read gms transport boundary conditions

1. **gmsfile** Name of the file which contains the GMS boundary condition data.

Reads a GMS boundary condition file which has been produced by the FEMWATER module and extracts pertinent transport boundary condition information.

Currently, only Dirichlet and Cauchy transport boundary conditions are recognized by **HydroGeoSphere**. 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 appropriate 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 appropriate 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.

   • • •

5.7.4.6   Immiscible Phase Dissolution Source

An immiscible phase dissolution source is one in which it is assumed that there is dissolution of an immobile liquid or solid phase into the subsurface water until all dissolvable material is exhausted. The following instruction can be used to define the source:

Immiscible phase dissolution data

1. **diss_mass** Mass of dissolvable material per unit volume of porous medium.

2. **diss_conc** The aqueous solubility value for the immiscible phase in mass of solute per unit volume of aqueous solution.

Chosen nodes are assigned first-type concentration values equal to **diss_conc**, which are maintained until all the material associated with a node is dissolved, at which time the node reverts back to an unconstrained condition. Once unconstrained, any water flowing in will generate mass due to the nonzero concentration, and so dissolution nodes should not be located on the outer surface of the domain.

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**.

• • •

5.7.4.7   Zero-order Source

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.

Zero order source

1. **npanel** Number of panels in the time-variable, zero-order source function. For each panel, enter the following:

   (a) **ton_val,toff_val, (bc_val(j),j=1,nspeciesmob)** Time on [T], time off [T] and mass of solute produced per unit volume of porous medium solids per unit time [M L^{-3}].

Nodes in the chosen zones area assigned zero-order source boundary conditions.

A panel is a point in time at which the source term is set to a new value. The first panel would normally start at time zero. The source term given for the last panel will be maintained until the end of the simulation. You can assign a static source term for the duration of the simulation by setting **npanel** to 1, **ton_val** to 0.0 and **toff_val** to a large number.

Note that if **nspeciesmob** is greater than 1, additional values of **bc_val** should be included.

$$\bullet\ \bullet\ \bullet$$

5.8 Materials and Material Properties

5.8.1 General

Currently, the following five basic types of media can be defined in **HydroGeoSphere**:

1. porous

2. dual continuua

3. discretely-fractured

4. surface runoff

5. channel flow

6. et

Porous media and dual continuua are defined by three-dimensional 8-node (brick) or 6-node (prism) elements, discretely-fractured, surface flow and ET media are defined by two-dimensional 4-node (rectangle) or 3-node (triangle) elements and channel flow media are defined by 1-node (line) elements. By default, every 3-D element in the problem domain is a porous media element. Elements of the other four 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 5.5. 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 domain type

1. **zone_type** Can be one of the strings: porous media, dual, fracture, surface, channel or et.

Causes **grok** to read a string defining the type of domain to which additional instructions should be applied.

<div align="center">• • •</div>

5.8.1.1 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 5.4. For example, 3-D block elements must first be selected before a new porous media or dual zone can be defined, 2-D rectangular faces are selected for fracture or surface zones and 1-D segments are selected for channel zones.

In a problem where there are to be dual, fracture, surface or channel 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, where by default all 3-D elements are in porous media zone 1.

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

New zone

1. **num_zone** Zone number.

Chosen elements are assigned a new zone number.

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.

<div align="center">• • •</div>

The following set of instructions, inserted in the *prefix*.grok file would create a new fracture zone.

```
use domain type
fracture

clear chosen faces
choose faces z plane
0.05
1.e-5

new zone
1
```

Assign zone zero

Assign all elements to zone number zero.

$\bullet\bullet\bullet$

This instruction was added for the case where we are using multiple surfaces to choose elements and assign them unique zone numbers. If we first assign all elements to zone zero, we can then find elements that were not chosen by any surface and are still assigned to zone zero.

5.8.1.2 Saving and Retrieving Element Zone Numbers

The following commands can be used to store and retrieve *porous media* element zone numbers.

Write zones to file

1. **zone_file** Name of the file to which element zone numbers will be written.

$\bullet\bullet\bullet$

Read zones from file

1. **zone_file** Name of the file from which zone information will be read.

$\bullet\bullet\bullet$

For example, if you want elements 1, 5, 9, and 44 to have the zone number 2 and element 8 to have the zone number 3, the file would contain:

```
    1   2
    5   2
    9   2
   44   2
    8   3
```

5.8.1.3 Defining New Zones Using ARCVIEW Files

The following commands can be used to assign *porous media and surface flow* element zone numbers using files created by ESRI Arcview. Note that Arcview `.shp` and `.dbf` files should have the same prefix and be in the same directory. Currently, the following Arcview features are not supported:

1. Attributes with a date format.

2. Files containing anything except polygons.

3. Shape files with, for example, geographical projections or coordinate translations that are stored in `.shx` and `.prj` files.

Zones from arcview ASCII grid

1. **arcview_file_name** Name of the Arcview ASCII grid file.

2. **nodata_replace** Zone number to assign to cells with no data.

3. **nzone_add_arcview** A number to be added to zone numbers read from the ASCII grid file.

Chosen elements will be assigned zone numbers from the Arcview ASCII grid file (`.asc`). Since these files are 2 dimensional, elements with the same x and y coordinate will be assigned the same zone number, regardless of their z coordinate.

In cells where there is no data value, the variable **nodata_replace** will replace the Arcview default(usually -9999).

The variable **nzone_add_arcview** can be used to preserve existing zone numbers. For example, if there are zones already defined in the model which are numbered from 1 to 7, and the Arcview file contains zones numbered from from 1 to 4, you can set **nzone_add_arcview** to 7, and zones assigned from the Arcview file would be numbered from 8 to 11.

<p style="text-align:center">• • •</p>

Zones from arcview

1. **arcview_file_name** Prefix of the Arcview shape file.

2. **nodata_replace** Zone number to assign to an element which is not in any polygon

3. **attribute** Field name used to assign the zones, which must be written exactly as in the **.dbf** file (case sensitive).

4. **nzone_add_arcview** A number to be added to zone numbers read from the ASCII grid file.

5. **unproject_file** If true, you must supply input data which is identical to that required for the Project grid instruction described in Section 5.3.10.

6. **project_file** If true, you must supply input data which is identical to that required for the Project grid instruction described in Section 5.3.10.

Chosen elements will be assigned zone numbers from the Arcview shape file (**.shp**). Since these files are 2 dimensional, elements with the same x and y coordinate will be assigned the same zone number, regardless of their z coordinate.

If an element centroid falls within a polygon, it will receive the attribute of that polygon. Since a polygon may have several attributes, the variable **attribute** can be used to specify which one is to be applied. For example, a shapefile of the geology of a region may contain polygons having the attributes age, type, domain, unit, formation and name. Setting the value of **attribute** to 'domain' will assign zone numbers based on the domain number. The choice of a different attribute will result in a different pattern of zone numbering. The variable **attribute** is chosen from the database file (**.dbf**), which can be opened in a spreadsheet program in order to choose the name.

The variable **nzone_add_arcview** can be used to preserve existing zone numbers. For example, if there are zones already defined in the model which are numbered from 1 to 7, and the Arcview file contains zones numbered from from 1 to 4, you can set **nzone_add_arcview** to 7, and zones assigned from the Arcview file would be numbered from 8 to 11.

<div align="center">● ● ●</div>

Since Arcview shape files created on a Windows platform may not be binary compatible with a UNIX platform, the zone numbers can be written to an ASCII file using the write zones to file and then read on the UNIX platform using the read zones from file instruction (as described in Section 5.8.1.2. This can also be useful in cases where the zones from arcview instruction takes a long time to execute, since the results can be stored initially and then read much more quickly in subsequent runs.

5.8.1.4 Defining New Zones Using GridBuilder .GEN Files

The following commands can be used to assign *porous media* element zone numbers using .gen files created by GridBuilder.

Zone by gb gen file

1. **fname** Name of the GridBuilder .gen file.

All elements in the grid will be assigned zone numbers from the GridBuilder .gen file. Since these files are 2 dimensional, elements with the same x and y centroid coordinates will be assigned the same zone number, regardless of their z coordinate.

If elements are found which are outside the region defined in the GridBuilder file, the zone number 0 (zero) will be assigned and a warning will be issued.

In the case of **grok** grids which have been flipped using the Y vertical instruction, the x and z centroid coordinates are used instead to determine the zone number.

<div align="center">● ● ●</div>

5.8.1.5 Selecting Zones

These instructions can be used to alter the set of chosen zones for the current zone type (i.e. porous media, dual, fracture or surface.)

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.

<div align="center">● ● ●</div>

Choose zones all

Selects all zones. This is useful if you wish to assign a property to all zones in the grid.

<div align="center">● ● ●</div>

Choose zone number

1. **num_zone** Zone number.

Selects the zone numbered **num_zone**.

<div align="center">● ● ●</div>

5.8.1.6 Modifying Zoned 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 assign a new porous media hydraulic conductivity to all zones, and thus to all elements in the problem. The following set of instructions, inserted in the *prefix*.`grok` file would accomplish this:

```
use domain 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 zones of type fracture or surface, warning messages will be issued to the screen and *prefix*o.`eco` file and execution will halt.

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 saturated flow properties are described in Section 5.8.2.

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 *prefix*.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. To assign new properties through the use of a material properties file, we first need to issue the following instruction:.

Properties file

1. **props_file_name** Name of the material properties file.

This file will be searched for materials given as input to the Read properties instruction discussed below.

● ● ●

The Properties file instruction has two benefits: first, it allows you to create sets of material properties and give them meaningful file names and second, it allows you to easily switch between material property sets merely by changing the file name given in the *prefix*.grok file.

Any line in a material properties file which is completely blank or which begins with an exclamation point (!) is treated as a comment and is ignored by **grok**. 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 5.8.2.1) may be included. Figure 5.11 shows an example of a material defined for the verification problem discussed in Section 4.4.1.

To make use of the material properties file, you would issue the following instruction:

Read properties

1. **mat_name** Name of the material.

Chosen zones are assigned properties of the material named **mat_name** in the current material properties file as defined in a properties file instruction.

● ● ●

So for example, the following set of instructions could be inserted in the *prefix*.grok file:

```
use domain type
```

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

k isotropic
500.0

specific storage
0.0

porosity
1.0

longitudinal dispersivity
10.0

transverse dispersivity
0.1

transverse vertical dispersivity
0.1

tortuosity
0.1

end material
```

Figure 5.11: Example of a Material Defined in an **.mprops** File

```
porous media

properties file
my.mprops

clear chosen zones

choose zones all

read properties
sand
```

The instruction read Properties would, in this case, search the porous media material properties file my.mprops 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.

A summary of the final data which has been defined for each zone is listed in the *prefix*o.eco file, and an example is shown in Figure 5.12.

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

5.8.2   Saturated Subsurface Flow

5.8.2.1   Porous Medium

HydroGeoSphere 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 5.5. These values are representative of a sand.

Note that the default state of the hydraulic conductivity tensor ($\mathbf{K}$ in Equation 2.2) is that it is isotropic and that all off-diagonal terms are zero.

You can use the general methods and instructions outlined in Section 5.8.1 to modify the default distribution of saturated porous media properties.

Note that each instruction given here has an associated scope of operation. For example, some can only be used in the *prefix*.grok file, in which case they will affect the current set of chosen zones or elements. Other instructions can only be used in a porous media properties (.mprops) file, in which case they affect only the named material of which they

```
-------------------------------------------------------------
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
```

Figure 5.12: Example Output for a Porous Media Material

Table 5.5: Default Values for Porous Media Saturated Flow Properties.

Parameter	Value	Unit
Name	Default Sand	
Hydraulic conductivity terms:		
K_{xx}	7.438×10^{-5}	m s^{-1}
K_{yy}	7.438×10^{-5}	m s^{-1}
K_{zz}	7.438×10^{-5}	m s^{-1}
K_{xy}	0.0	m s^{-1}
K_{xz}	0.0	m s^{-1}
K_{yz}	0.0	m s^{-1}
Specific storage S_s	1.0×10^{-4}	m^{-1}
Porosity θ_s	0.375	
Poisson's ratio ν^*	0.3	-
Solids compressibility γ_s	0.0	kg^{-1} m s^2
Loading efficiency ζ	1.0	-
Unsaturated flow relation type	Pseudo-soil	

are a part. Finally, some instructions can be used in both types of files, in which case their behaviour will be as described above, and will depend on which type of file (i.e. *prefix*`.grok` or `.mprops`) that they are placed in. It is very important that the user understand this behaviour, and the scope of each instruction will be clearly indicated as they are introduced and discussed below.

K isotropic
Scope: .grok .mprops

1. **kval** Hydraulic conductivity [L T^{-1}].

Assign an isotropic hydraulic conductivity (i.e. $K_{xx} = K_{yy} = K_{zz}$).

$\bullet\bullet\bullet$

K anisotropic
Scope: .grok .mprops

1. **kvalx, kvaly, kvalz** Hydraulic conductivities [L T^{-1}] in the x-, y- and z-directions respectively.

Assigns anisotropic hydraulic conductivities.

$\bullet\bullet\bullet$

K tensor
Scope: .grok .mprops

1. **valx, valy, valz** Main-diagonal terms of the hydraulic conductivity tensor K_{xx}, K_{yy} and K_{zz} [L T^{-1}].

2. **valxy, valxz, valyz** Off-diagonal terms of the hydraulic conductivity tensor K_{xy}, K_{xz} and K_{yz} [L T^{-1}].

Assign hydraulic conductivities which include the off-diagonal terms. Note that this option will only work if **HydroGeoSphere** is running in finite element mode and so this switch will automatically be set.

$\bullet\bullet\bullet$

Specific storage
Scope: .grok .mprops

1. **val** Specific storage [L^{-1}], S_s in Equation 2.10.

$\bullet\bullet\bullet$

Porosity
Scope: .grok .mprops

1. **val** Porosity [$L^3\ L^{-3}$], θ_s in Equation 2.1.

• • •

Poisson ratio
`Scope: .grok .mprops`

1. **val** Poisson's Ratio [dimensionless], ν^* in Equation 2.28b.

• • •

Loading efficiency
`Scope: .grok .mprops`

1. **val** Loading efficiency [dimensionless], ζ^* in

Equation 2.28b.

• • •

Compute loading efficiency
`Scope: .grok .mprops` This command should be included in the input file if you want the preprocessor to compute the loading efficiency [dimensionless] based on Equation 2.28b. In this case, values of Poisson's ratio and compressibility of solids and water for the current media will be used. Porous media compressibility will be computed from the specific storage value. If this command is not included, the default or user-defined loading efficiency value is used instead.

• • •

Solids compressibility
`Scope: .grok .mprops`

1. **val** Solids compressibility [$L\ T^2\ M^{-1}$], K_s in Equation 2.22.

• • •

Element K isotropic
`Scope: .grok`

1. **kval** Hydraulic conductivity [$L\ T^{-1}$].

Chosen elements are assigned isotropic hydraulic conductivities (i.e. $K_{xx} = K_{yy} = K_{zz}$).

• • •

Element K anisotropic
`Scope: .grok`

1. **kvalx, kvaly, kvalz** Hydraulic conductivities $[\text{L T}^{-1}]$ in the x-, y- and z-directions respectively.

Chosen elements are assigned anisotropic hydraulic conductivities.

• • •

Read elemental k from file
Scope: .grok

1. **input_k_file_name** Name of the file which contains the variable K data supplied by the user.

• • •

This file should contain the following input data:

1. **element_number, kxx, kyy, kzz.** Element number, hydraulic conductivities $[\text{L T}^{-1}]$ in the x-, y- and z-directions respectively.

All elements are assigned a variable K from the file. For example, if there are 4 elements, with $K_{xx} = K_{yy} = 5$ m day^{-1} and $K_{zz} = 2$ m d^{-1}, 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
```

The user can supply variable values for any number of elements in file **input_k_file_name**. **grok** will then produce data for all elements honouring any previous zoned values and the user specified element-variable values, which are written to the file `prefixo.elemental_k`.

Map isotropic k from raster
Scope: .grok

1. **rasterfile** Name of the raster file containing the hydraulic conductivity $[\text{L T}^{-1}]$ values. This is a string variable. The file should be formatted as outlined in Section H.

For each element in the set of currently chosen zones, a value for the isotropic hydraulic conductivity (i.e. $K_{xx} = K_{yy} = K_{zz}$) will be interpolated from the raster file data.

• • •

Map anisotropic k from raster
Scope: .grok

1. **rasterfile_x** Name of the raster files containing the x component (i.e. K_{xx}) hydraulic conductivity [L T^{-1}] values. This is a string variable. The file should be formatted as outlined in Section H.

2. **rasterfile_y** As above but for the y component (i.e. K_{yy}) hydraulic conductivity [L T^{-1}] values.

3. **rasterfile_z** As above but for the z component (i.e. K_{zz}) hydraulic conductivity [L T^{-1}] values.

For each element in the set of currently chosen zones, values for the anisotropic hydraulic conductivity (i.e. K_{xx}, K_{yy} and K_{zz}) will be interpolated from the raster file data.

●●●

Map porosity from raster
Scope: .grok

1. **rasterfile** Name of the raster file containing the porosity values. This is a string variable. The file should be formatted as outlined in Section H.

For each element in the set of currently chosen zones, a value for the porosity will be interpolated from the raster file data.

●●●

Read elemental porosity from file
Scope: .grok

1. **input_por_file_name** Name of the file which contains the variable porosity data supplied by the user. This file should contain the following input data:

    (a) **element_number, por** Element number, porosity.

All elements are assigned a variable porosity from the file. The user can supply variable values for any number of elements in file **input_por_file_name**. **grok** will then produce data for all elements honouring any previous zoned values and the user specified element-variable values, which are written to the file `prefixo.elemental_por`.

●●●

Read elemental specific storage from file
Scope: .grok

1. **input_stor_file_name** Name of the file which contains the variable specific storage data supplied by the user. This file should contain the following input data:

    (a) **element_number, stor** Element number, specific storage [L^{-1}], S_s in Equation 2.10..

All elements are assigned a variable specific storage from the file. The user can supply variable values for any number of elements in file **input_stor_file_name**. **grok** will then produce data for all elements honouring any previous zoned values and the user specified element-variable values, which are written to the file `prefixo.elemental_stor`.

<div align="center">• • •</div>

Map tortuosity from raster
`Scope: .grok`

1. **rasterfile** Name of the raster file containing the tortuosity values. This is a string variable. The file should be formatted as outlined in Section H.

For each element in the set of currently chosen zones, a value for the tortuosity will be interpolated from the raster file data.

<div align="center">• • •</div>

Read elemental tortuosity from file
`Scope: .grok`

1. **input_tort_file_name** Name of the file which contains the variable tortuosity data supplied by the user. This file should contain the following input data:

    (a) **element_number, tort** Element number, tortuosity.

All elements are assigned a variable tortuosity from the file. The user can supply variable values for any number of elements in file **input_por_file_name**. **grok** will then produce data for all elements honouring any previous zoned values and the user specified element-variable values, which are written to the file `prefixo.elemental_tort`.

<div align="center">• • •</div>

Write element k
`Scope: .grok`

1. **filenm** Name of the file in which to write the hydraulic conductivity information.

Writes a file of element hydraulic conductivity values in ascii format.

<div align="center">• • •</div>

The following FORTRAN code segment shows how the file is opened and the data written:

```
open(itmp,file=filenm,status='unknown')
if(k_variable .or. k_rand) then
    write(itmp,'(i10,3e12.5)') (i,kxx(i),kyy(i),kzz(i),i=1,ne)
else
    write(itmp,'(i10,3e12.5)') (i,kxx(iprop(i)),kyy(iprop(i)),kzz(iprop(i)),i=1,
```

```
        end if
```

where `kxx, kyy` and `kzz` are the hydraulic conductivity values in the three principal directions and ne is the number of elements in the mesh. If k values are zoned, then the variable `iprop(i)` refers to the element zone id number of element `i`.

Write element k at z
`Scope: .grok`

1. **zfix** z-coordinate for choosing which element to write.

2. **rtol** Distance from **zfix** for test.

3. **filenm** Name of the file in which to write the hydraulic conductivity information.

This instruction is identical to write element k except that information if only written for elements whose centroid is within a distance of **rtol** of the z-coordinate **zfix** are written to the file.

• • •

Get average k
`Scope: .grok`
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
`Scope: .grok`

1. **aecl_nd_file** Name of the file which contains the nodal coordinates for the AECL Motif mesh.

2. **aecl_ne_file** Name of the file which contains the element incidences and material property numbers for the AECL Motif mesh.

AECL Motif grid element material numbers are mapped onto the existing **HydroGeoSphere** mesh. The mapping is performed based on the proximity of **HydroGeoSphere** and AECL Motif element centroids. Once the material numbers have been mapped, zones should be chosen and appropriate material properties should be assigned.

• • •

Table 5.6: Default Values for Fractured Media Saturated Flow Properties.

Parameter	Value	Unit
Name	Default 100 micron fracture	
Aperture w_f	100×10^{-6}	m
Conductivity (computed) K_f	$(\rho g \, w_f^2)/(12\mu)$	m s^{-1}
Specific storage (computed) S_{sf}	$\rho g \alpha_w$	m^{-1}
Unsaturated flow relation type	Pseudo-fracture	

Random K field from FGEN
Scope: .grok

1. **fgenfile** Name of file which contains the random hydraulic conductivity information generated by FGEN.

A random K field which was generated by the program FGEN [*Robin et al.*, 1993] is mapped onto the current mesh. **HydroGeoSphere** Automatically dimensions and treats the hydraulic conductivity vector as an elemental property, as opposed to a zoned property.

FGEN generates two cross-correlated 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.

The output from FGEN is in the form of values distributed on a rectangular grid which can be either 2-D or 3-D. Normally, a 3-D distribution is used and values are mapped by first determining which rectangular grid block the element centroid falls in, and then generating a value at the centroid by trilinear interpolation of the 8 neighbouring grid values. If an element is located outside of the FGEN grid, it is assigned a missing value, which is read from the FGEN file.

If the FGEN data is 2-D, values are generated using bilinear interpolation. For example, suppose the FGEN values are distributed on a plane parallel to the xy plane. In this case any element whose centroid had xy coordinates that fell inside the range of the FGEN data would receive a value determined by bilinear interpolation from the 4 neighbouring grid values, but independent of the z coordinate of the element centroid.

The resulting element-variable hydraulic conductivity data are written to the file `prefixo.elemental_k`.

● ● ●

5.8.2.2   Discrete Fractures

Unless you modify the default values, all discrete fracture zones (and 2-D fracture elements) in the domain will be assigned the default saturated properties which are listed in Table 5.6. These values are representative of a 100 micron fracture.

You can use the general methods and instructions outlined in Section 5.8.1 to modify the default distribution of saturated fractured media properties.

As was the case for the instructions which modify porous medium properties, these instructions also have a scope of operation, the only difference being that they would appear in the .fprops file instead of the .mprops file.

Aperture
`Scope: .grok .fprops`

1. **val** Fracture aperture [L], w_f in Equation 2.11.

• • •

High-k plane
`Scope:   .fprops`

1. **valx** Thickness [L] for the high-conductivity plane.

2. **valx** Hydraulic conductivity [L T^{-1}] for the high-conductivity plane.

Treat the fracture material as a high-conductivity plane where the hydraulic conductivity and thickness are given by the user.

• • •

Specific storage
`Scope: .grok .fprops`

1. **val** Specific storage [L^{-1}], S_{sf} in Equation 2.14.

• • •

Coupling length
`Scope: .grok .fprops`

1. **val** Coupling length [L].

• • •

Coupling hydraulic conductivity
`Scope: .grok .fprops`

1. **val** Coupling hydraulic conductivity [L/T].

• • •

Impermeable matrix

`Scope: .grok`

This command causes the matrix to be considered impermeable and so flow and transport will only be computed for the fractures. This overrides any values defined in the *prefix*`.grok` or `.fprops` file so you do not have to alter them.

● ● ●

Make fractures from fgen

`Scope:.grok`

1. **mat_name** Name of the fracture material whose properties are to be read and assigned.

2. **fname** Name of the FGEN file which contains the random aperture information.

Chosen faces are assigned to a new fracture zone and given the properties of material **mat_name**, which is read from the `.fprops` file **fname**. Any aperture information contained therein will be overwritten by the information that is then read from the FGEN file. The FGEN file aperture values are distributed on a rectangular grid with uniform spacings in the three principal axis directions. Element aperture values are then interpolated to the element face centroids by bilinear (if the FGEN data is on a 2-D plane) or trilinear (if the FGEN data is fully 3-D) interpolation.

grok assumes that the FGEN file contains log aperture values and so the following conversion is done:

$$w_f = \exp(w_{f_{FGEN}})$$

where $w_{f_{FGEN}}$ is the fracture aperture value read from the FGEN file.

● ● ●

Make fractures from tecplot file

`Scope: .grok`

1. **tecplot_frac_file** Name of the Tecplot file which contains the fracture triangle information.

2. **label** Name of the fracture material whose properties are to be read and assigned.

A new fracture zone is created based on triangle data given in the tecplot file **tecplot_frac_file**. Fracture zone properties are assigned from the material **label**.

The tecplot file must contain one or more tecplot ZONE statements. Each tecplot ZONE statement must contain, or be followed by a statement containing the string `N=` followed by the number of vertices and then by the string `E=` followed by the number of triangles in the tecplot ZONE.

Comments beginning with the character `#` are allowed.

The tecplot variables represent the xyz coordinates of the vertices and must be given in three-column, point format.

The fracture elements are defined so that they conform closely to the triangulated surface given in the tecplot file, and can include diagonal (i.e. inclined) faces.

● ● ●

The following Tecplot file produced the fracture elements shown in Figure 5.13.

```
VARIABLES="X","Y","Z"
ZONE T="Fractures_primary"
N= 5,E= 3,DATAPACKING=POINT,  ZONETYPE=FETRIANGLE
110. -1. -5.
51.  102.   -3.
-3.  49.  99.
0. 100. 100
0. 0. 0.
1 3 5
1 2 3
2 3 4
```

Make recharge spreading layer
Scope: .grok

1. **frack** Hydraulic conductivity $[\text{L T}^{-1}]$ of the recharge spreading layer.

2. **aperture** Thickness [L] of the recharge spreading layer.

3. **remove_fixed_head_conn** A logical switch which determines whether the recharge spreading layer is connected to an existing specified head boundary.

Chosen faces which are on the top of the domain are assigned to a new fracture zone which has the properties of a recharge spreading layer. 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).

A recharge spreading layer may allow water to bypass the rest of the system if it is connected to or in close proximity to a constant head boundary which can act as a discharge point for the system. In such cases you can set the variable **remove_fixed_head_conn** to be true, to prevent such direct connections. 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.

● ● ●
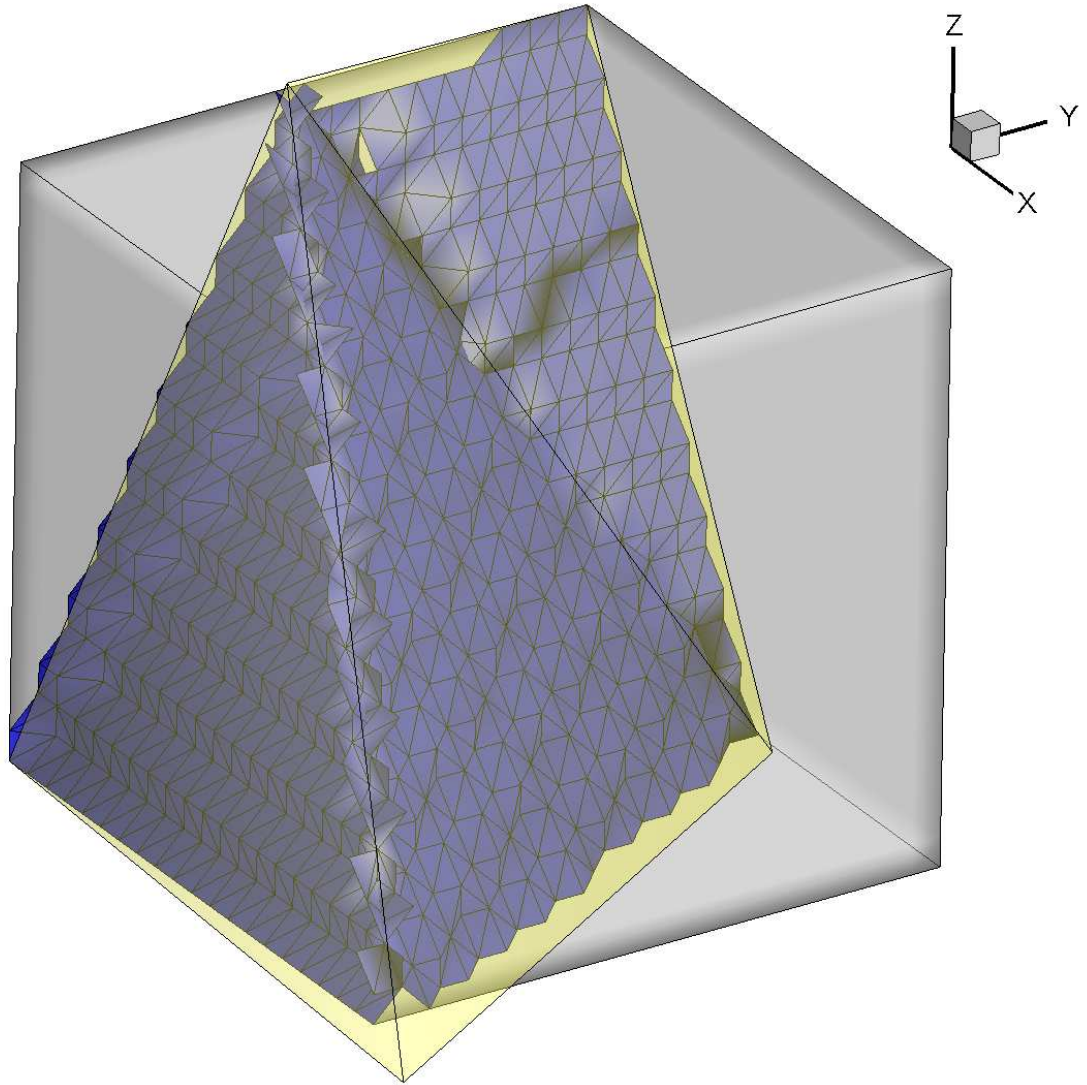
Figure 5.13: Example of fracture generation showing 3D porous medium domain (grey cube), tecplot triangles (yellow triangles) and the resulting fracture elements (blue triangles).

Table 5.7: Default Values for Dual-continuum Saturated Flow Properties.

Parameter	Value	Unit
Name	Default Sand	
Hydraulic conductivity terms:		
$\quad K_{xx}$	7.438×10^{-5}	m s^{-1}
$\quad K_{yy}$	7.438×10^{-5}	m s^{-1}
$\quad K_{zz}$	7.438×10^{-5}	m s^{-1}
Specific storage S_{sd}	1.0×10^{-4}	m^{-1}
Porosity	0.375	
Volume fraction of porous medium w_d	0.01	
Unsaturated flow relation type	Pseudo-soil	

5.8.2.3   Dual Continuum

Unless you modify the default values, all dual-continuum zones (and elements) in the domain will be assigned the default properties which are listed in Table 5.7. These values are representative of a sand.

Note that the default state of the hydraulic conductivity tensor ($\mathbf{K_d}$ in Equation 2.17) is that it is isotropic. It should also be noted that for dual continuua, off-diagonal terms are not considered.

You can use the general methods and instructions outlined in Section 5.8.1 to modify the default distribution of saturated dual-continuum properties.

As was the case for the instructions which modify porous medium properties, these instructions also have a scope of operation, the only difference being that they would appear in the `.dprops` file instead of the `.mprops` file.

K isotropic
`Scope: .grok .dprops`

1. **kval** Hydraulic conductivity [L T^{-1}].

Assign an isotropic hydraulic conductivity (i.e. $K_{xxd} = K_{yyd} = K_{zzd}$).
<p style="text-align:center">• • •</p>

K anisotropic
`Scope: .grok .dprops`

1. **kvalx, kvaly, kvalz** Hydraulic conductivities [L T^{-1}] in the x-, y- and z-directions respectively.

Assigns anisotropic hydraulic conductivities.
<p style="text-align:center">• • •</p>

Specific storage
`Scope: .grok .dprops`

1. **val** Specific storage $[\text{L}^{-1}]$,S_{sd}, but defined in a similar way to $S - s$ in Equation 2.10.

• • •

Porosity
`Scope: .grok .dprops`

1. **val** Porosity $[\text{L}^3\ \text{L}^{-3}]$, $theta_{sd}$ in Equation 2.16.

• • •

Volume fraction dual medium
`Scope:   .dprops`

1. **val** Volume fraction $[\text{L}^3\ \text{L}^{-3}]$, w_d in Equation 2.16.

The volume fractions of the dual medium and porous medium always add up to 1.0.

• • •

First-order fluid exchange coefficient
`Scope:   .dprops`

1. **val** First-order fluid exchange rate, α_{wd} in Equation 2.65.

• • •

Interface k
`Scope:   .dprops`

1. **val** Interface hydraulic conductivity $[\text{L}\ \text{T}^{-1}]$, K_a in Equation 2.65.

• • •

Convert pm k to macropore k
`Scope:   .dprops`

1. **val** Porous medium background hydraulic conductivity $K_{bkgrd}[L/T]$.

We can express the bulk hydraulic conductivity of a dual-continuum K_{bulk} as the sum of the porous media K_{bkgrd} and fracture K_d components:

$$K_{bulk} = K_{bkgrd}\,(1 - w_d) + K_d\,w_d \qquad (5.11)$$

where w_d is the volume fraction $[\text{L}^3\ \text{L}^{-3}]$ in Equation 2.16.

If we assume that the observed (porous medium) hydraulic conductivity is equal to K_{bulk}, and supply an educated guess for K_{bkgrd}, we can rearrange the equation and calculate K_d as :

$$K_d = [K_{bulk} - K_{bkgrd} (1 - w_d)]/w_d \tag{5.12}$$

For all elements in the currently chosen dual zones, the porous medium hydraulic conductivity is replaced by K_{bkgrd} and the fracture hydraulic conductivity K_d is set equal to the calculated value.

<div align="center">• • •</div>

5.8.2.4   Wells

Wells, as outlined in Section 2.1.2.5, can be set up using the following instructions, where the variables **flowrate, radw** and **radc** correspond to Q_w, r_s and r_c in Equation 2.32 respectively.

To neglect the component of well storage arising from water level changes in a well casing that is open to atmosphere, the well casing radius r_c can be set equal to 0.0. The well casing radius r_c only appears in the first term of the right-hand side of equation 2.32 and setting it equal to 0.0 limits storage in the well to water compressibility effects.

Make well

1. **well_name** Descriptive name for the well up to 40 characters.

2. **x1, y1, z1** xyz-coordinates of one end of the well.

3. **x2, y2, z2** xyz-coordinates of the other end of the well.

4. **npanel_well** Number of time panels for which a well flowrate is specified. For each panel, enter the following:

    (a) **ton_val, flowrate** Time on [T] and flowrate [L^3 T^{-1}].

5. **xd, yd, zd** xyz-coordinates of the discharge point of the well.

6. **radw** Well screen radius [L].

7. **radc** Well casing radius [L].

Element face edges (i.e. segments) which form the shortest path between the two nodes closest to the end points of the well (i.e. **x1, y1, z1** and **x2, y2, z2**) will become 1-D line elements which act as an open (i.e. fluid-filled) well.

Water will be extracted from the node in the well closest to the point **xd, yd, zd**.

If the flowrate is set to zero, the well is passive but can still transmit water vertically through its screen.

$$\bullet \; \bullet \; \bullet$$

Make well from element list

1. **well_name** Descriptive name for the well up to 40 characters.

2. **nelemlin(nwell)** Number of node pairs defining the well. For each pair, enter the following:

    (a) **inlin(1,i,nwell),inlin(2,i,nwell)** Node numbers defining the segment.

3. **npanel_well** Number of time panels for which a well flowrate is specified. For each panel, enter the following:

    (a) **ton_val, flowrate** Time on [T] and flowrate [L^3 T^{-1}].

4. **xd, yd, zd** *xyz*-coordinates of the discharge point of the well.

5. **radw** Well screen radius [L].

6. **radc** Well casing radius [L].

Listed node pairs will become 1-D line elements which act as an open (i.e. fluid-filled) well.

Water will be extracted from the node in the well closest to the point **xd, yd, zd**.

If the flowrate is set to zero, the well is passive but can still transmit water vertically through its screen.

$$\bullet \; \bullet \; \bullet$$

Make well from nodes

1. **well_name** Descriptive name for the well up to 40 characters.

2. **nnodes** Number of nodes defining the well.

3. **fname** Name of file containing the list of node numbers.

4. **npanel_well** Number of time panels for which a well flowrate is specified. For each panel, enter the following:

    (a) **ton_val, flowrate** Time on [T] and flowrate [L^3 T^{-1}].

5. **disch_point** Number of node located at the discharge point of the well.

6. **radw** Well screen radius [L].

7. **radc** Well casing radius [L].

Listed nodes will become 1-D line elements which act as an open (i.e. fluid-filled) well.

Water will be extracted from the node defined as the **disch_point**.

If the flowrate is set to zero, the well is passive but can still transmit water vertically through its screen.

<center>• • •</center>

Make infilled well

1. **well_name** A descriptive name for the infilled well. Up to 40 characters.

2. **x1, y1, z1** *xyz*-coordinates of one end of the well.

3. **x2, y2, z2** *xyz*-coordinates of the other end of the well.

4. **npanel_well** Number of time panels for which a well flowrate is specified. For each panel, enter the following:

    (a) **ton_val, flowrate** Time on [T] and flowrate [L^3 T^{-1}].

5. **xd, yd, zd** *xyz*-coordinates of the discharge point of the well.

6. **radw** Well screen radius [L].

7. **radc** Wells casing radius [L].

8. **mat_name** Name of the porous media material to be used for the infill material.

Element face edges (i.e. segments) which form the shortest path between the two nodes closest to the end points of the well (i.e. **x1, y1, z1** and **x2, y2, z2**) will become 1-D line elements which act as an infilled well. The vertical hydraulic conductivity K_{zz} and specific storage S_s of material **mat_name** will be read from the current porous media material properties file `.mprops` and used to define the infill material.

Water will be extracted from the node in the well closest to the point **xd, yd, zd**.

If the flowrate is set to zero, the well is passive but can still transmit water vertically through its screen.

<center>• • •</center>

Make infilled well from element list

1. **well_name** Descriptive name for the well up to 40 characters.

2. **nelemlin** Number of node pairs defining the well. For each pair, enter the following:

    (a) **inlin1,inlin2** Node numbers defining the segment.

3. **npanel_well** Number of time panels for which a well flowrate is specified. For each panel, enter the following:

(a) **ton_val, flowrate** Time on [T] and flowrate [L^3 T^{-1}].

4. **xd, yd, zd** *xyz*-coordinates of the discharge point of the well.

5. **radw** Well screen radius [L].

6. **radc** Well casing radius [L].

7. **mat_name** Name of the porous media material to be used for the infill material.

Listed node pairs will become 1-D line elements which act as an infilled well. The vertical hydraulic conductivity (K_{zz}) and specific storage S_s of material **mat_name** will be read from the current porous media material properties file.mprops and used to define the infill material.

Water will be extracted from the node in the well closest to the point **xd, yd, zd**.

If the flowrate is set to zero, the well is passive but can still transmit water vertically through its screen.

$\bullet \bullet \bullet$

Make well node

1. **well_name** Descriptive name for the well up to 40 characters.

2. **x1, y1, z1** *xyz*-coordinates of the well.

3. **npanel_well** Number of time panels for which a well flowrate is specified. For each panel, enter the following:

(a) **ton_val, flowrate** Time on [T] and flowrate [L^3 T^{-1}].

4. **radw** Well screen radius [L].

5. **radc** Well casing radius [L].

The node closest to point **x1, y1, z1** will act as an open (i.e. fluid-filled) well.

$\bullet \bullet \bullet$

Saturated wells

This instruction causes wells to remain fully saturated.

$\bullet \bullet \bullet$

Allow intersecting 1d elements

By default, **HydroGeoSphere** prevents wells from being connected (i.e. two line elements cannot share a node if in different wells). This instruction overrides the default and allows wells to be connected.

$\bullet \bullet \bullet$

5.8.2.5   Tile Drains

Tile drains, as outlined in Section 2.1.2.6, can be set up using the following instructions. Currently, this option requires that the system be variably-saturated.

Make tile drain

1. **npts** The number of points defining a polyline. Read the following **npts** times:

   (a) **x1, y1, z1** The coordinates of a point on the polyline.

2. **tile_name** Descriptive name for the tile drain up to 40 characters.

3. **npanel_tile** Number of time panels for which a tile discharge rate is specified. For each panel, enter the following:

   (a) **ton_val, flowrate_tile** Time on [T] and tile discharge rate [L^3 T^{-1}], Q_t in Equation 2.35.

4. **xd, yd, zd** The coordinates of the tile discharge point.

5. **wid** Tile width [L]

Element face edges (i.e. segments) which form the shortest path between the points given for the polyline will become 1-D line elements which act as a tile drain. The points **x1, y1, z1** 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.

Water will be extracted from the node in the tile closest to the point **xd, yd, zd**. 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.*

If the discharge rate **qtile** is set to zero, the tile is passive but can still collect or exfiltrate water.

• • •

5.8.2.6   Cutoff Walls

Cutoff walls can be used to represent internal impermeable boundaries which could occur, for example, in a funnel-and-gate system.

Cutoff Walls

1. **nwalls** Number of impermeable cutoff walls to be defined. Read the following **nwalls** times:

(a) **iorient** Wall orientation number.

(b) **avalue,afrom,ato,verfrom,verto** Position, lateral minimum and maximum extent, vertical minimum and maximum extent.

The orientation parameter **iorient** should be set to 1 if the cutoff wall is in the xz plane. In this case **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 the cutoff wall is in the yz plane and **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.

Only vertical cutoff walls are allowed.

Cutoff walls are restricted to rectangular block element grids.

● ● ●

5.8.2.7 Imported from FRACTRAN

Fractran properties

This instruction is to be used after a mesh has been loaded using the `Read fractran 2d grid` described in Section 5.3.5.1. It uses the FRACTRAN prefix defined there to read the associated properties.

● ● ●

5.8.3 Variably-saturated Subsurface Flow

As discussed in Section 2.1, one of the requirements for simulating variably-saturated flow is that we define the constitutive relationships that govern the relation between the pressure head, saturation and relative permeability. These relationships must be defined for the porous medium, discrete fractures and the dual continuum and, for each of these types of media, the choice is to use either pseudo-soil, functional or tabular relationships. Wells are a special case and are handled automatically by **HydroGeoSphere**.

Media specific instructions and default values for porous media, discrete fractures and dual continuua are given in Sections 5.8.3.1, 5.8.3.2 and 5.8.3.3 respectively.

General instructions for modifying the default functional and tabular relationships are given in Sections 5.8.3.4 and 5.8.3.5 respectively.

5.8.3.1 Porous Medium

By default, all porous media zones (and elements) in the domain will use a constitutive relationship based on the pseudo-soil relation, as developed by *Huyakorn et al.* [1994].

Table 5.8: Default Values for Functions Defining the Porous Media Constitutive Relationships, for the Van Genuchten and Brooks-Corey models.

Parameter	Value	Unit
Common		
Residual water saturation, S_{wr}	0.053	
Pore-connectivity, l_p	0.5	
Beta (power index), β	3.1768	
Van Genuchten		
Alpha (power index), α	3.5237	m^{-1}
Gamma (Power index, computed), γ	$1 - 1/\beta$	
Brooks-Corey		
Exponent (computed)	$2/\beta + l_p + 2$	
Air entry pressure, ψ_{ae}	-0.16	m
Alpha (power index, computed), α	$-1/\psi_{ae}$	m^{-1}

Table 5.9: Default Pressure-saturation and Saturation-relative permeability Tables for Variably-saturated Porous Media.

Pressure(m) ψ	Saturation S_w
-10.0	0.053
0.0	1.0

Saturation S_w	Relative permeability k_{rw}
0.053	0.053
1.0	1.0

Essentially, in the pseudo soil relationship, the 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.

If you wish to use Van Genuchten or Brooks-Corey functions to describe the constitutive relationship you can do so using the instructions given in Section 5.8.3.4. Unless you modify them, the default values given in Table 5.8 will be used to define the functional relationships.

If you wish to use tables to describe the constitutive relationships you can do so using the instructions given in Section 5.8.3.5. Unless you modify them, the default values of water saturation versus pressure head and saturation versus relative permeability listed in Table 5.9 will be used to define the tabular relationships.

Relative permeability xy

`Scope: .grok .mprops`

When using functions or tables to define the constitutive relationships, this instruction causes **grok** to apply the relative permeability to horizontal flow only so that water travels vertically under saturated hydraulic conductivity conditions, similar to the behaviour of the pseudo-soil relation. This instruction should be applied to porous media, as discussed in

Table 5.10: Default Values for Functions Defining the Discrete Fracture Constitutive Relationships, for the Van Genuchten and Brooks-Corey models.

Parameter	Value	Unit
Residual water saturation, S_{wr}	0.053	
Power index (alpha), α	3.5237	m^{-1}
Power index (beta), β	3.1768	
Power index (gamma, computed), γ	$1 - 1/\beta$	
Pore-connectivity, l_P	0.5	
Brooks-Corey exponent	$2/\beta + l_p + 2$	

Table 5.11: Default Pressure-saturation and Saturation-relative permeability Tables for Variably-saturated Discrete Fractures.

Pressure(m) ψ	Saturation S_w
-10.0	0.053
0.0	1.0

Saturation S_w	Relative permeability k_{rw}
0.053	0.053
1.0	1.0

Section 5.8.1.

• • •

5.8.3.2 Discrete Fractures

By default, all discrete fracture zones (and elements) in the domain will use a constitutive relationship based on the pseudo-soil relation, as developed by *Huyakorn et al.* [1994]. Essentially, in the pseudo soil relation, the medium is assigned a nodal saturation of 1 above the water table and 0 (zero) below it. Also by default, the effective area available for flow across the fracture-matrix interface is maintained at its maximum value, regardless of the state of fracture saturation.

If you wish to use Van Genuchten or Brooks-Corey functions to describe the constitutive relationships you can do so using the instructions given in Section 5.8.3.4. Unless you modify them, the default values given in Table 5.10 will be used to define the functional relationships.

If you wish to use tables to describe the constitutive relationships you can do so using the instructions given in Section 5.8.3.5. Unless you modify them, the default values of water saturation versus pressure head and saturation versus relative permeability listed in Table 5.11 will be used to define the tabular relationships.

If you wish to modify the default relationship between pressure, saturation and effective area you can do so using the instructions given below. For each instruction we will again

Table 5.12: Default Pressure-Effective Area Table for Variably-saturated Discrete Fractured Media.

Pressure(m)	Effective Area
0.0	1.0

indicate its scope (i.e. `.grok`, `.fprops`). Note that the functional relationships are always applied in a similar way to all fracture zones in the domain, while tabular relationships can vary from fracture zone to fracture zone if so desired.

The following instructions should be applied to discrete fractures, as discussed in Section 5.8.1.

Effective area tables...end

`Scope:.grok .fprops`

Causes **grok** to use tables to describe the pressure-effective area relationship for the fracture and to begin reading a group of effective area table instructions until it encounters an End instruction.

By default values of contact area versus pressure head listed in Table 5.12 will be used.

• • •

These instructions are available for modifying the default pressure-effective area relationship:

Pressure-effective area

`Scope: .grok .fprops`

1. **pressure(1), effective area(1)** First entry.
2. **pressure(2), effective area(2)** Second entry.
    ⋮ etc.
n. **pressure(n), effective area(n)** n^{th} entry.
n+1. **end** The string 'end'

Causes **HydroGeoSphere** to begin reading instructions which describe the pressure-contact area table which will define the relationship for the fracture.

Paired values of pressure ψ and effective area, which varies between 0 (full reduction) and 1 (no reduction), should be entered from lowest pressure (i.e. most negative) to highest pressure, usually zero. The last line of the table must be an **end** card, and the number of entries in the list are counted automatically to determine the table size.

• • •

Effective area Wang-Narasimhan functions

Table 5.13: Default Values for Functions Defining the Dual Continuua Constitutive Relationships, for the Van Genuchten and Brooks-Corey models.

Parameter	Value	Unit
Residual water saturation, S_{wr}	0.053	
Power index (alpha), α	3.5237	m^{-1}
Power index (beta), β	3.1768	
Power index (gamma, computed), γ	$1 - 1/\beta$	
Pore-connectivity, l_p	0.5	
Brooks-Corey exponent	$2/\beta + l_p + 2$	

Table 5.14: Default Pressure-saturation and Saturation-relative permeability Tables for Variably-saturated Dual Continuua.

Pressure(m) ψ	Saturation S_w
-10.0	0.053
0.0	1.0

Saturation S_w	Relative permeability k_{rw}
0.053	0.053
1.0	1.0

Scope:.grok
Causes **HydroGeoSphere** to use the approach of *Wang and Narasimhan*[1985], as discussed in Section 4.1.4, for computing the pressure-effective area relationship for the fractures. These functions will automatically be applied to all fracture zones.

• • •

5.8.3.3 Dual continuum

Dual continuua zones (and elements) in the domain will use a constitutive relationship based on the pseudo-soil relation, as developed by *Huyakorn et al.* [1994]. Essentially, in the pseudo soil relationship, the 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.

If you wish to use Van Genuchten or Brooks-Corey functions to describe the constitutive relationships you can do so using the instructions given in Section 5.8.3.4. Unless you modify them, the default values given in Table 5.13 will be used to define the functional relationships.

If you wish to use tables to describe the constitutive relationships you can do so using the instructions given in Section 5.8.3.5. Unless you modify them, the default values of water saturation versus pressure head and saturation versus relative permeability listed in Table 5.14 will be used to define the tabular relationships.

Relative permeability xy

`Scope: .grok .dprops`

When using functions or tables to define the constitutive relationships, this instruction causes **HydroGeoSphere** to apply the relative permeability to horizontal flow only so that water travels vertically under saturated hydraulic conductivity conditions, similar to the behaviour of the pseudo-soil relation. This instruction should be applied to dual continuua, as discussed in Section 5.8.1.

● ● ●

When simulating a system with porous and dual continuua, the constitutive relationships of the interface between the two continuua must also be defined. The following instructions allow you to do this:

> Interface unsaturated tables
> Interface unsaturated van genuchten functions
> Interface unsaturated brooks-corey functions
> Interface relative permeability xy

Input is identical to the generic forms of the commands discussed in the following sections, except that the scope is restricted to `.dprops`.

We will now discuss instructions of a general nature which can be used to modify the default constitutive relationships for the various media, beginning with the functional relationships.

5.8.3.4   Functional Constitutive Relationships

The instructions described here can be used to modify the default variably-saturated properties for a porous medium, discrete fracture or dual continuum. Before issuing them it is necessary to choose which type of medium they should be applied to, as discussed in Section 5.8.1.

For each instruction we will again indicate its scope (i.e. `.grok .mprops, .dprops, .fprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, while in a properties (e.g. `.mprops`) file, it will only affect the named material of which it is a part.

Unsaturated brooks-corey functions...End

`Scope: .grok .mprops .fprops .dprops`

Causes **grok** to use Brooks-Corey functions (Equation 2.5) to describe the constitutive relationships for the medium and to begin reading a group of instructions that define the function until it encounters an End instruction.

● ● ●

Unsaturated van genuchten functions…End
`Scope: .grok .mprops .fprops .dprops`
Causes **grok** to use Van Genuchten functions (Equation 2.7) to describe the constitutive relationships for the medium and to begin reading a group of instructions that define the function until it encounters an End instruction.

• • •

The previous two instructions are used to choose between the Brooks-Corey of Van Genuchten approaches for defining the functions. In either case, if no further instructions are issued, the default function parameter values given in Tables 5.8, 5.10 and 5.13 will be used for porous media, discrete fractures and dual continuua respectively.

In the case of porous and dual media, these instructions override the pseudo-soil default so that relative permeability factors are applied to both horizontal and vertical flow.

The following instructions can be used to modify the parameters which define the constitutive relationships:

Residual saturation
`Scope: .grok .mprops .fprops .dprops`

1. **val** Residual water saturation S_{wr}.

• • •

Alpha
`Scope: .grok .mprops .fprops .dprops`

1. **val** Power index alpha α $[L^{-1}]$.

For the Brooks-Corey function, this parameter is computed automatically from the air entry pressure. If you use this instruction you will be prompted to enter an air-entry pressure instead and **grok** will stop.

• • •

Beta
`Scope: .grok .mprops .fprops .dprops`

1. **val** Power index beta β.

For the van Genuchten function, this parameter must be greater than 1.0. If you enter a value less than 1.0 you will be warned and **grok** will stop. This value is used to compute ν according to Equation 2.9.

For the Brooks-Corey formulation, this value is used to recalculate the exponent in equation 2.6 unless the instruction Exponent has been used previously for this material.

• • •

Pore connectivity
`Scope: .grok .mprops .fprops .dprops`

1. **val** Pore connectivity l_p. The default value is 0.5.

Note that the default value of pore connectivity of 0.5 is a value recommended for the van Genuchten formulation, so your will probably want to redefine it for the Brooks-Corey formulation. The value recommended in this case is 2.0.

For the Brooks-Corey formulation, this value is used to recalculate the exponent in equation 2.6 unless the instruction Exponent has been used previously for this material.

● ● ●

Air entry pressure
`Scope: .grok .mprops .fprops .dprops`

1. **val** Air entry pressure [L].

For the Brooks-Corey function, this value is used to compute α [L^{-1}] according to Equation 2.5.

For the van Genuchten function, this parameter is not used. If you use this instruction you will be prompted to remove it and **grok** will stop.

● ● ●

Exponent
`Scope: .grok .mprops .fprops .dprops`

1. **val** Exponent in equation 2.6, which is used to computes k_r in the Brooks-Corey function. By default, the exponent is computed automatically from β and l_p. This instruction allows you to enter a different value.

For the van Genuchten function, this parameter is not used. If you use this instruction you will be prompted to remove it and **grok** will stop.

● ● ●

Minimum relative permeability
`Scope: .grok .mprops .fprops .dprops`

1. **val** Minimum relative permeability. During a simulation, the model will choose the maximum value for relative permeability between the minimum value specified here and the value computed from the active function, either Van Genuchten or Brooks-Corey. This option may improve convergence of the non-linear solution.

The following instructions can be used to generate pressure-saturation and saturation-relative permeability tables using the Van Genuchten or Brooks-Corey parameters defined

for the medium. In addition to the instructions given above which define the function, these additional instructions affect the properties of the tabular data.

• • •

Table smoothness factor
`Scope: .grok .mprops`

1. **val** Smaller values cause more points to be generated for a smoother, more accurate table. The default value is 1×10^{-3}.

• • •

Table minimum pressure
`Scope: .grok .mprops`

1. **val** The minimum pressure value in the pressure-saturation table. The default value is -1000.

• • •

Table maximum s-k slope
`Scope: .grok .mprops`

1. **val** The maximum slope of the saturation-relative permeability curve when nearing full saturation. The default value is 100.

• • •

Generate tables from unsaturated functions
`Scope: .grok .mprops`

This instruction generates the tables from the previously defined function parameters and writes the pressure saturation data to the file *prefix*o.p_s_table.*material*.dat and the saturation-relative permeability data to the file *prefix*o.s_k_table.*material*.dat. These files are written in Tecplot-compatible format so they can be easily plotted. The tabular values in the files can be copied into the .mprops file for use with the unsaturated table instructions described in Section 5.8.3.5

• • •

For example, if the following Van Genuchten function parameters were defined in the .mprops file:

```
unsaturated van genuchten functions
alpha
2.25
beta
1.89
```

```
    residual saturation
    0.16

    minimum relative permeability
    1e-2

    table smoothness factor
    1e-2

    table minimum pressure
    -10.

    generate tables from unsaturated functions

    end ! functions
```

then the contents of the pressure-saturation output file would be:

```
    Title =  "test.mprops/porous medium"
    #  Van Genuchten function:
    #  Residual water saturation        0.160000
    #  Alpha                             2.25000
    #  Power index (beta)                1.89000
    #  Pore connectivity                0.500000
    #  Computed power index (gamma)     0.685218
    #  Minimum relative permeability               0.100000E-01
    #  Table minimum pressure                      -10.0000
    #  Table maximum s-k slope                      100.000
    #  Table smoothness factor                     0.100000E-01
    variables="Pressure","Saturation"
    zone t="Pressure - Saturation"
    #unsaturated tables
    #pressure-saturation
    -10.0000000000000        0.174869375404582
    -7.50000000000000        0.181552629607745
    -5.00000000000000        0.196301039876425
    -3.75000000000000        0.212424751627904
    -2.50000000000000        0.247430530192203
    -1.87500000000000        0.284639681867462
    -1.25000000000000        0.361026934163956
    -0.937500000000000        0.435114163471319
    -0.625000000000000        0.564528209032747
    0.000000000000000E+000    1.00000000000000
    #end ! pressure-saturation
```
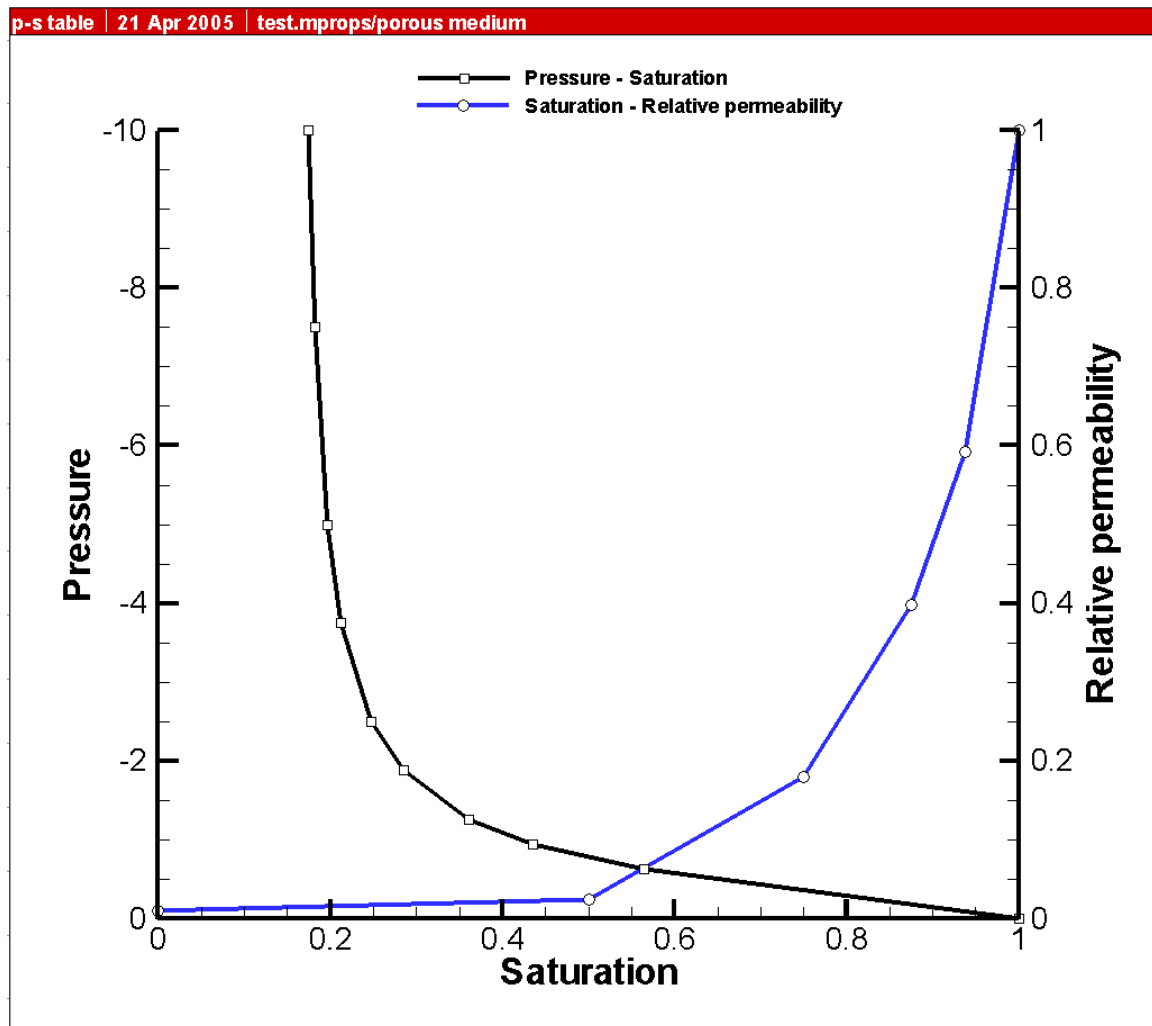
Figure 5.14: Example of Using Functional Parameters to Generate Tabular Constitutive Relationships.

The values used to define the table are included as comments, as are the instructions needed for incorporating the tabular data in the `.mprops` file.

Figure 5.14 is a plot of the resulting constitutive relationships. Note that the `.mprops` file and material names are used to form the Tecplot title and appear on the plot.

If desired, the `.mprops` file can be modified to use the tabular relationships. It is recommended that the Van Genuchten parameters that were used to generate the tabular data are retained in the file, either as comments or inside a skip on...skip off section. For our example, we could do the following:

```
skip on
unsaturated van genuchten functions
```

```
    ...etc...
    end ! functions
    skip off

    unsaturated tables
    pressure-saturation
    -10.0000000000000        0.174869375404582
    -7.50000000000000        0.181552629607745
    -5.00000000000000        0.196301039876425
    -3.75000000000000        0.212424751627904
    -2.50000000000000        0.247430530192203
    -1.87500000000000        0.284639681867462
    -1.25000000000000        0.361026934163956
    -0.937500000000000        0.435114163471319
    -0.625000000000000        0.564528209032747
    0.000000000000000E+000   1.00000000000000
    end ! pressure-saturation

    saturation-relative k
    0.000000000000000E+000   1.000000000000000E-002
    0.500000000000000        2.340977494655262E-002
    0.750000000000000        0.180180362276789
    0.875000000000000        0.398602012758378
    0.937500000000000        0.591618307040100
    1.00000000000000         1.00000000000000
    end ! saturation-relative k
    end ! unsaturated tables
```

We will now move on to discuss instructions which can be used to define tabular relationships.

5.8.3.5   Tabular Constitutive Relationships

The instructions described here can be used to modify the default variably-saturated properties for a porous medium, discrete fracture or dual continuum. Before issuing them it is necessary to choose which type of medium they should be applied to, as discussed in Section 5.8.1.

For each instruction we will again indicate its scope (i.e. .grok .mprops, .fprops, .dprops). Recall that if an instruction is used in the *prefix*.grok file, it will affect the current set of chosen zones, while in a properties (e.g. .mprops) file, it will only affect the named material of which it is a part.

Unsaturated tables

Scope: `.grok` `.mprops` `.fprops` `.dprops`
Causes **grok** to use tables to describe the constitutive relationships for the medium and to begin reading a group of instructions that define the tables until it encounters an End instruction.

If no further instructions are issued, the default tabular parameter values given in Tables 5.9, 5.11 and 5.14 will be used for porous media, discrete fractures and dual continuua respectively.

In the case of porous and dual media, this instruction overrides the pseudo-soil default so that relative permeability factors are applied to both horizontal and vertical flow.

● ● ●

The following instructions can be used to modify the default tables which define the constitutive relationships:

Pressure-saturation
Scope: `.grok` `.mprops` `.fprops` `.dprops`

1. **pressure(1), saturation(1)** First entry.
2. **pressure(2), saturation(2)** Second entry.
   ⋮ etc.
n. **pressure(n), saturation(n)** n^{th} entry.
n+1. **end** The string 'end'

Paired values of pressure ψ and saturation S should be entered from lowest pressure (i.e. most negative) to highest pressure, usually zero. The last line of the table must be an **end** card, and the number of entries in the list are counted automatically to determine the table size.

● ● ●

Saturation-relative k
Scope: `.grok` `.mprops` `.fprops` `.dprops`

1. **saturation(1), relative permeability(1)** First entry.
2. **saturation(2), relative permeability(2)** Second entry.
   ⋮ etc.
n. **saturation(n), relative permeability(n)** n^{th} entry.
n+1. **end** The string 'end'

Paired values of saturation S and relative permeability k_{rw} should be entered from lowest to highest saturation. The last line of the table must be an **end** card, and the number of entries in the list are counted automatically to determine the table size.

● ● ●

Parameter	Value	Unit
Name	Surface 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	
Coupling length	1×10^{-4}	m

Table 5.15: Default Properties for Surface Flow.

5.8.4 Surface Flow

Unless you modify the default values, all zones (and elements) in the the surface flow domain will be assigned the default properties listed in Table 5.15.

For each instruction we will again indicate its scope (i.e. `.grok`, `.oprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, while in a properties (e.g. `.oprops`) file, it will only affect the named material of which it is a part.

X friction
`Scope: .grok .oprops`

1. **val** x friction factor, n_x in Equation 2.45.

● ● ●

Y friction
`Scope: .grok .oprops`

1. **val** y friction factor, n_y in Equation 2.46.

● ● ●

Rill storage height
`Scope: .grok .oprops`

1. **val** Rill storage height [L], H_d in Section 2.2.2.2.

● ● ●

Obstruction storage height
`Scope: .grok .oprops`

1. **val** Obstruction storage height [L], H_o in Section 2.2.2.2.

Table 5.16: Default Properties for Evapotranspiration.

Parameter	Value	Unit
Name	Default ET (grass)	
Canopy storage parameter	0.04	m
Initial interception storage	0.04	m
Transpiration fitting parameters:		
C1	0.5	
C2	0.0	
C3	1.0	
Transpiration limiting saturations:		
Wilting point	0.05	
Field capacity	0.2	
Oxic limit	0.6	
Anoxic limit	0.8	
Evaporation limiting saturations:		
Minimum	0.5	
Maximum	0.1	
Leaf area index	1.0	
Root zone depth	0.2	m
Evaporation depth	0.2	m

• • •

Coupling length
Scope: `.grok` `.oprops`

1. **val** Coupling length [L], l_{exch} in Equation 2.67.

• • •

5.8.5   Evapotranspiration

NOTE: Moisture contents referred to below and in section 2.4.3 are actually input in **grok** as saturations, which are less prone to error as they always vary between zero and 1, while moisture content varies between zero and porosity.

Unless you modify the default values, all zones (and elements) in the ET domain will be assigned the default properties listed in Table 5.16. .

For each instruction we will again indicate its scope (i.e. `.grok`, `.etprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, while in a properties (e.g. `.etprops`) file, it will only affect the named material of which it is a part.

Canopy storage parameter
`Scope: .etprops`

1. **cint_et** Canopy storage parameter [L], c_{int} in Equation 2.69.

<div align="center">● ● ●</div>

Initial interception storage
`Scope: .etprops`

1. **init_sint_et** initial canopy interception storage value [L], S_{int}^0 in Equation 3.63.

<div align="center">● ● ●</div>

Transpiration fitting parameters
`Scope: .etprops`

1. **C_1** Coefficient C_1 [dimensionless] in Equation 2.71.

2. **C_2** Coefficient C_2 [dimensionless] in Equation 2.71.

3. **C_3** Coefficient C_3 [dimensionless] in Equation 2.73. By default, this coefficient is set to 1.0, which gives a linear ramping function whereas higher values would give higher order ramping functions.

<div align="center">● ● ●</div>

Transpiration limiting saturations
`Scope: .etprops`

1. **thwp_et** Saturation [dimensionless] at wilting point, θ_{wp} in Equation 2.73.

2. **thfc_et** Saturation [dimensionless] at field capacity, θ_{fc} in Equation 2.73.

3. **tho_et** Saturation [dimensionless] at oxic limit, θ_o in Equation 2.73.

4. **than_et** Saturation [dimensionless] at anoxic limit, θ_{an} in Equation 2.73.

<div align="center">● ● ●</div>

Evaporation limiting saturations
`Scope: .etprops`

1. **the2_et** Saturation [dimensionless] below which evaporation is zero, θ_{e2} in Equation 2.73.

2. **the1_et** Saturation [dimensionless] above which full evaporation can occur, θ_{e1} in Equation 2.73.

• • •

The following instruction can be used to modify the default value (1.0 for all time) of the leaf area index *LAI*:

Lai tables...End
`Scope: .etprops`

    1.   **time(1), lai(1)** First entry.
    2.   **time(2), lai(2)** Second entry.
          ⋮ etc.
    n.   **time(n), lai(n)** n^{th} entry.

Causes **grok** to begin reading a group of time vs leaf area index instructions until it encounters an End instruction.

Paired values of time t and leaf are index *LAI* should be entered from earliest to latest time. The number of entries in the list are counted automatically to determine the table size.

Observed values of leaf are index [*Scurlock et al.*, 2001] and maximum rooting depth [*Canadell et al.*, 1996] for various terrestrial biomes are shown in Table 5.17 .

• • •

Root depth
`Scope: .etprops`

    1. **root_depth_et** Maximum root depth [L].

A normalized root depth function is mapped onto porous media elements above this maximum depth. Currently, four root depth functions are available; constant, linear, quadratic and cubic, as shown in Figure 5.15. These functions are defined such that the area under the normalized function is 1.0.

• • •

By default, the linear form of the depth function is used. The following instructions are available for using the other forms:

        Rdf constant function
        Rdf quadratic decay function
        Rdf cubic decay function

Evaporation depth
`Scope: .etprops`

    1. **evap_depth_et** Evaporation depth [L].

Table 5.17: Observed Values of Leaf Area Index [*Scurlock et al.*, 2001] and Maximum Rooting Depth [*Canadell et al.*, 1996] for Various Terrestrial Biomes.

Vegetative cover	Leaf area index	Maximum rooting depth
Desert	1.31	9.5
Tundra, circumpolar and alpine	2.69	0.5
Wetlands, temperate and tropical	6.34	
Grasslands, temperate	2.50	2.6
Grasslands, tropical	2.50	15.0
Crops, temperate and tropical	4.22	2.1
Shrubland, heath or Mediterranean-type vegetation		5.2
Forest:		
boreal deciduous broadleaf	2.64	2.0
boreal evergreen needleleaf	3.50	2.0
boreal/temperate deciduous needleleaf	4.63	2.0
temperate deciduous broadleaf	5.12	2.9
temperate evergreen needleleaf	6.70	3.9
temperate evergreen broadleaf	5.82	
tropical deciduous broadleaf	3.92	3.7
tropical evergreen broadleaf	4.90	7.3
Plantations (managed forests)	8.72	
Overall mean	5.23	
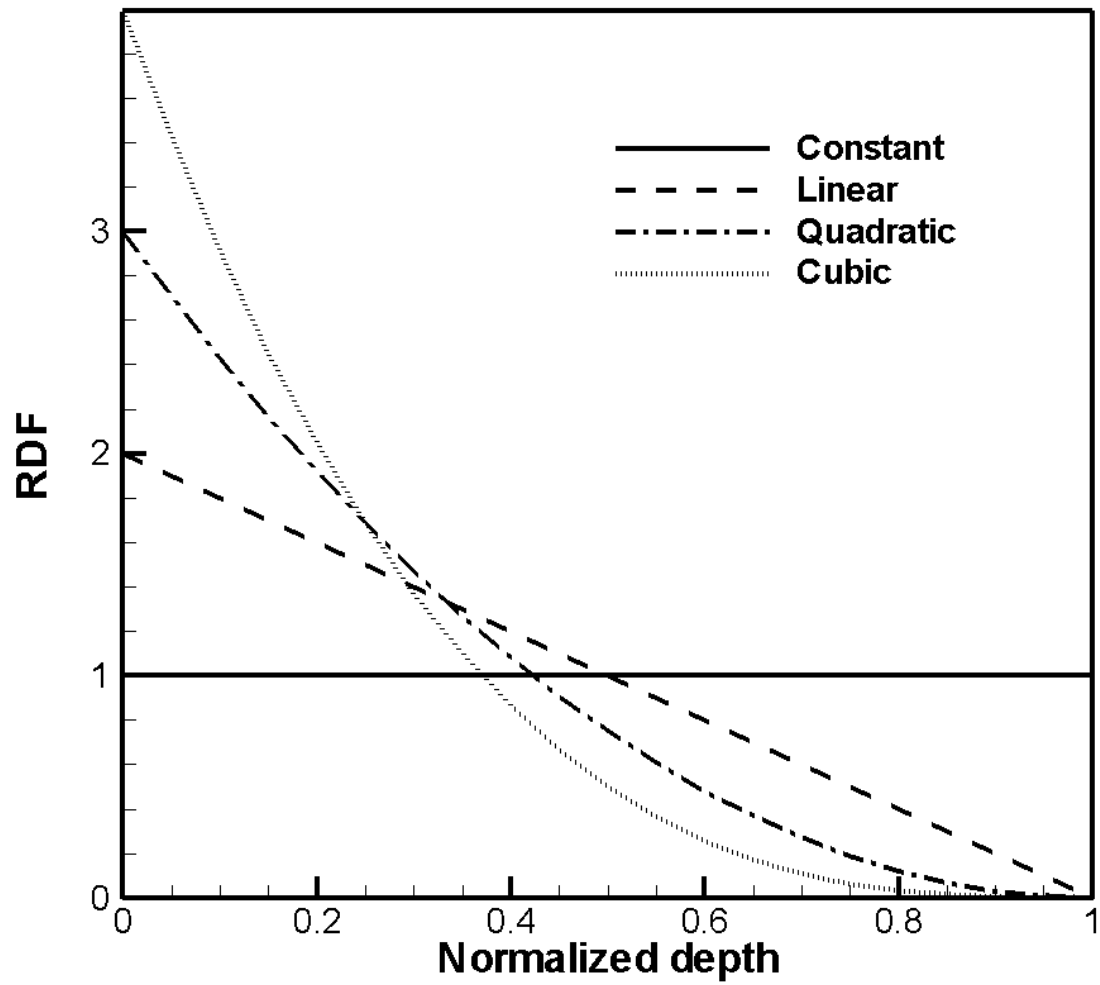
Figure 5.15: Normalized Root Depth Functions.

Evaporation as a function of depth is treated in a similar fashion as the root zone depth described above.

<div align="center">• • •</div>

By default, the linear form of the evaporation function is used. The following instructions are available for using the other forms:

> Edf constant function
> Edf quadratic decay function
> Edf cubic decay function

Echo et at point
Scope: .grok

1. **x1, y1** xy-coordinate.

This instruction finds the column of nodes that the given coordinate falls within and then writes the pertinent evapotranspiration information to the `o.eco` file.

<div align="center">• • •</div>

For example:

```
ET properties for element column

ZONE:  1
ET MATERIAL: dense deciduous forest
    Canopy storage parameter                        =   0.0
    Transpiration fitting parameter C1              =   0.3
    Transpiration fitting parameter C2              =   0.2
    Transpiration fitting parameter C3              =   3.0E-6
    Moisture content at wilting point               =   0.32
    Moisture content at field capacity              =   0.2
    Moisture content at oxic limit                  =   0.76
    Moisture content at anoxic limit                =   0.9
    Moisture content at at energy limiting maximum  =   0.32
    Moisture content at at energy limiting minimum  =   0.2
    Initial interception storage                    =   0.0

Tabular data:
     Time           LAI
     0.00000        2.08000
   0.100000E+42     2.08000

Maximum evaporative zone depth   =    2.0
     Depth           EDF
```

```
      0.00000         2.00000
      1.00000         0.00000


 Maximum root zone depth          =     4.0
        Depth           RDF
      0.00000         2.00000
      1.00000         0.00000


      ELEMENT    ET FLAG      DEPTH          EDF             RDF
       12889        T        0.00000      0.281045        0.146305
       11528        T        0.303244     0.234783        0.134740
       10167        T        0.606488     0.188522        0.123174
        8806        T        0.909732     0.142261        0.111609
        7445        T        1.21298      0.959994E-01    0.100044
        6084        T        1.51622      0.497381E-01    0.884783E-01
        4723        T        1.81946      0.765174E-02    0.769130E-01
        3362        T        2.1227       0.00000         0.653477E-01
        2001        T        2.4259       0.00000         0.537824E-01
         640        T        2.7292       0.00000         0.422171E-01
                                                     ------------    ------------
     Totals                                            1.00000        0.942610
```

In this case, the element column falls in ET zone 1, and the default linear root depth and evaporation depth functions have been used and mapped onto it. All of the evaporative potential has been distributed to the element column, as indicated by a total EDF value of 1.0. However, only a portion (0.942610 or 94%) of the total RDF of 1.0 has been mapped onto it, because the maximum root zone depth (4 m) exceeds the total depth of the element column at this point (2.7292 m).

5.8.6 Transport

5.8.6.1 Porous Medium

By default, all porous media zones (and elements) in the domain will be assigned default porous media transport properties which are listed in Table 5.18. Included here are parameters for modifying the porous medium so that it acts as a double-porosity medium for simulating transport, as described in Section 2.5.1.3 and also for isotopic fractionation, as described in Section 2.5.1.4.

The following instructions can be applied to porous media, as discussed in Section 5.8.1, to modify the default transport parameters. For each instruction we will indicate its scope (i.e. .grok .mprops). Recall that if an instruction is used in the *prefix*.grok file, it will affect the current set of chosen zones, while in a properties (e.g. .mprops) file, it will only affect the named material of which it is a part.

Table 5.18: Default Values for Porous Media Transport Properties.

Parameter	Value	Unit
Longitudinal dispersivity α_l	1.0	m
Horizontal component of transverse dispersivity α_t	0.1	m
Vertical component of transverse dispersivity α_t	0.1	m
Bulk density ρ	2031.25	kg m^{-3}
Tortuosity τ	0.1	
Immobile zone porosity θ_{Imm}	0.0	
Immobile zone mass transfer coefficient α	0.0	s^{-1}
Reverse fractionation rate k_r	0.0	s^{-1}
Fractionation factor α_r	0.0	
Mass ratio, solid to water phases x_r	0.0	
Thermal conductivity of the solids k_s	3.0	W m^{-1} K^{-1}
Specific heat capacity of the solids c_s	738.0	J kg^{-1} K^{-1}
Density of the solids ρ_s	2650.0	kg m^{-3}

Longitudinal dispersivity
Scope: .grok .mprops

   1. **val** Longitudinal dispersivity [L], α_l in Equation 2.81.

● ● ●

Transverse dispersivity
Scope: .grok .mprops

   1. **val** Horizontal component of the transverse dispersivity [L], α_t in Equation 2.81.

● ● ●

Vertical transverse dispersivity
Scope: .grok .mprops

   1. **val** Vertical component of the transverse dispersivity [L], α_t in Equation 2.81.

● ● ●

Tortuosity
Scope: .grok .mprops

   1. **val** Tortuosity , τ in Equation 2.81.

● ● ●

Anisotropic tortuosity ratio
Scope: .grok .mprops

1. **y_tortratio** Tortuosity ratio in the y-direction. Default value is 1.

2. **z_tortratio** Tortuosity ratio in the z-direction. Default value is 1.

By default, tortuosity is isotropic, since the ratio values are set to 1 in both the y and z directions. You may make tortuosity anisotropic by entering a value greater than 0 and less than 1. These values will be used to multiple the tortuosity , τ in the y and z directions respectively, to obtain the directional values.

$\bullet\ \bullet\ \bullet$

Bulk density
Scope: .grok .mprops

1. **val** Bulk density [M L^{-3}], ρ_b in Equation 2.80. If this instruction is used, the value of the density of solids previously saved for this material is overwritten by the density of solids computed from the bulk density, the density of water and the porosity [$\rho_s = (\rho_b - \theta_s\rho)/(1 - \theta_s)$].

$\bullet\ \bullet\ \bullet$

By default, the porous medium acts a single-porosity medium (i.e. the immobile zone is inactive) because the porosity and mass transfer coefficient are set to zero. In order to activate the double porosity feature, you can enter non-zero values for these parameters using the following two instructions:

Immobile zone porosity
Scope: .grok .mprops

1. **val** Immobile zone porosity, θ_{Imm} in Equations 2.85, 2.119 and 2.120.

$\bullet\ \bullet\ \bullet$

Immobile zone mass transfer coefficient
Scope: .grok .mprops

1. **val** Immobile zone mass transfer coefficient [T^{-1}], α_{Imm} in Equation 2.118.

$\bullet\ \bullet\ \bullet$

Isotope fractionation data...End
Scope: .mprops
Causes **grok** to begin reading a group of isotope fractionation instructions until it encounters an End instruction.

If no further instructions are issued, the default isotopic fractionation parameter values listed in Table 5.18 will be used.

$\bullet\ \bullet\ \bullet$

The following three instructions can be used to change these values:

Reverse rate
`Scope:   .mprops`

   1. **val** Reverse fractionation rate $[\mathrm{L}^{-1}]$, k_r in Equation 2.127.

<div align="center">● ● ●</div>

Fractionation factor
`Scope:   .mprops`

   1. **val** Fractionation factor, α_r in Equation 2.127.

<div align="center">● ● ●</div>

Rock-water mass ratio
`Scope:   .mprops`

   1. **val** Isotopic rock-water mass ratio, x_r in Equation 2.86.

<div align="center">● ● ●</div>

The next four instructions can be used to change the thermal properties of the porous medium:

Thermal conductivity of solid
`Scope: .grok .mprops`

   1. **val** Temperature invariant thermal conductivity of the solids $[\mathrm{W\ L^{-1}\ K^{-1}}]$. The bulk thermal conductivity is computed internally from the volume fractions of the solid and liquid phases.

<div align="center">● ● ●</div>

Temperature-dependent thermal conductivity of solid
`Scope: .grok .mprops`

   1. **k_s1** Thermal conductivity $[\mathrm{W\ L^{-1}\ K^{-1}}]$ at temperature t_{s1}.

   2. **t_s1** Temperature $[°\mathrm{C}]$ at which the thermal conductivity is equal to k_{s1}.

If that instruction is specified, the thermal conductivity of the solid phase is temperature dependent. The bulk thermal conductivity is also temperature dependent and is computed internally from the volume fractions of the solid and liquid phases. It is assumed here that the thermal conducutivity of the solids decreases at a constant rate of 1% per 10°C increase in temperature and the relationship between thermal conductivity and temperature

Table 5.19: Default Values for Discrete Fracture Transport Properties.

Parameter	Value	Unit
Longitudinal dispersivity α_l	1.0	m
Transverse dispersivity α_t	0.1	m

is defined with the pair of values (k_{s1}, t_{s1}).

● ● ●

Specific heat capacity of solid
`Scope: .grok .mprops`

1. **val** Specific heat capacity of the solid phase [J kg^{-1} K^{-1}]. The default value is 730.0 J kg^{-1} K^{-1}.

● ● ●

Note that the density of the solid phase of the porous medium is now computed automatically from the bulk density and porosity.

5.8.6.2   Discrete Fractures

By default, all fracture zones (and elements) in the domain will be assigned default transport properties which are listed in Table 5.19.

The following instructions can be applied to discrete fractures, as discussed in Section 5.8.1, to modify the default transport parameters. For each instruction we will indicate its scope (i.e. `.grok .fprops`). Recall that if an instruction is used in the *prefix*`.grok` file, it will affect the current set of chosen zones, while in a properties (e.g. `.fprops`) file, it will only affect the named material of which it is a part.

Longitudinal dispersivity
`Scope: .grok .fprops`

1. **val**

Longitudinal dispersivity [L], similar to α_l in Equation 2.81.

● ● ●

Transverse dispersivity
`Scope: .grok .fprops`

1. **val** Transverse dispersivity [L], similar to α_t in Equation 2.81.

● ● ●

Table 5.20: Default Values for Dual-continuua Transport Properties.

Parameter	Value	Unit
Longitudinal dispersivity α_{ld}	1.0	m
Horizontal component of transverse dispersivity α_{td}	0.1	m
Vertical component of transverse dispersivity α_{td}	0.1	m
Bulk density ρ_{bd}	2650.0	kg m^{-3}
Tortuosity τ_d	0.1	
First-order mass transfer coefficient α_s	0.0	s^{-1}

Coupling dispersivity
Scope: .grok .oprops

1. **val** Dispersivity value [L] to be applied during exchange between the overland flow and subsurface domains.

Default value is 0.0.

• • •

5.8.6.3   Dual continuum

By default, all dual continuua zones (and elements) in the domain will be assigned default transport properties which are listed in Table 5.20.

The following instructions can be applied to dual continuua, as discussed in Section 5.8.1, to modify the default transport parameters. For each instruction we will indicate its scope (i.e. .grok .dprops). Recall that if an instruction is used in the *prefix*.grok file, it will affect the current set of chosen zones, while in a properties (e.g. .dprops) file, it will only affect the named material of which it is a part.

Longitudinal dispersivity
Scope: .grok .dprops

1. **val** Longitudinal dispersivity [L], α_{ld} in Equation 2.89.

• • •

Transverse dispersivity
Scope: .grok .dprops

1. **val** Horizontal component of the transverse dispersivity [L], α_{td} in Equation 2.89.

• • •

Vertical transverse dispersivity
Scope: .grok .dprops

1. **val** Vertical component of the transverse dispersivity [L], α_{td} in Equation 2.89.

$$\bullet \; \bullet \; \bullet$$

Tortuosity
Scope: `.grok .dprops`

1. **val** Tortuosity , τ_d in Equation 2.89.

$$\bullet \; \bullet \; \bullet$$

Anisotropic tortuosity ratio
Scope: `.grok .mprops`

1. **y_tortratio** Tortuosity ratio in the y-direction. Default value is 1.

2. **z_tortratio** Tortuosity ratio in the z-direction. Default value is 1.

By default, tortuosity is isotropic, since the ratio values are set to 1 in both the y and z directions. You may make tortuosity anisotropic by entering a value greater than 0 and less than 1. These values will be used to multiple the tortuosity , τ_d in the y and z directions respectively, to obtain the directional values.

$$\bullet \; \bullet \; \bullet$$

Bulk density
Scope: `.grok .dprops`

1. **val** Bulk density [M L^{-3}], ρ_{bd} in Equation 2.88.

$$\bullet \; \bullet \; \bullet$$

First-order mass exchange
Scope: `   .dprops`

1. **val** First-order mass transfer coefficient [L^{-1}], α_s in Equations 2.122 and 2.123.

$$\bullet \; \bullet \; \bullet$$

5.8.6.4 Surface Runoff

By default, all surface flow zones (and elements) in the domain will be assigned default transport properties which are listed in Table 5.21.

The following instructions can be applied to the surface flow domain, as discussed in Section 5.8.1, to modify the default transport parameters. For each instruction we will indicate its scope (i.e. `.grok .oprops`). Recall that if an instruction is used in the *prefix*`.grok` file,

Table 5.21: Default Values for Surface Flow Transport Properties.

Parameter	Value	Unit
Longitudinal dispersivity α_{lo}	1.0	m
Transverse dispersivity α_{to}	1.0	m
Coupling dispersivity	0.0	m

it will affect the current set of chosen zones, while in a properties (e.g. `.oprops`) file, it will only affect the named material of which it is a part.

Longitudinal dispersivity
`Scope: .grok .oprops`

1. **val** Longitudinal dispersivity [L], α_{lo}. Analogous to α_l in Equation 2.81.

● ● ●

Transverse dispersivity
`Scope: .grok .oprops`

1. **val** Horizontal component of the transverse dispersivity [L], α_t. Analogous to α_t in Equation 2.81.

● ● ●

Coupling dispersivity
`Scope: .grok .oprops`
1. **val** Dispersivity value [L] to be applied during exchange between the overland flow and subsurface domains.

Default value is 0.0.

● ● ●

Dry albedo
`Scope: .grok .oprops`

1. **Value** Dry albedo for soil type, α_{dry} in Equation 2.107. Default value is 0.35.

● ● ●

Saturated albedo
`Scope: .grok .oprops`

1. **Value** Saturated albedo for soil type, α_{sat} in Equation 2.107. Default value is 0.18

● ● ●

Heat coupling length
`Scope: .grok .oprops`

1. **Value** The heat coupling length [L], is the depth of the surface/subsurface exchange zone used to compute α_o in Equation 2.126. Default value is 1.0×10^{-4} m.

• • •

5.9 Output

During execution, **grok** and **HydroGeoSphere** create many output files, for which a complete list with brief descriptions can be found in Appendix B. Here, we will discuss in more detail the output which is of most interest to the user.

The main mechanism for generating output from **HydroGeoSphere** is through the use of the Output times instruction. For each output time defined by the user, there will be a problem dependent set of output files which will be generated automatically. Here is a partial list of the files which were created by the verification problem described in Section 4.4.2:

```
f_cdo.head.001
f_cdo.velocity_darcy.001
f_cdo.velocity_darcy_fracture.001
f_cdo.velocity_linear.001
f_cdo.velocity_linear_fracture.001
f_cdo.fracture_aperture.001
f_cdo.concentration.Radium.001
f_cdo.concentration.Thorium.001
f_cdo.concentration.Uranium.001
f_cdo.concentration.Radium.002
f_cdo.concentration.Thorium.002
f_cdo.concentration.Uranium.002
```

As you can see, file names are made up of the problem prefix, in this case `f_cd`, a descriptor, for example `o.concentration.Radium.` and a 3 digit number such as 001, which relates the data contained in the file to an output time number. Note that for output time number 001, there are quite a few more output files than for number 002. This is because the flow solution in this case is steady-state, and so it is fully described by one set of output. For timestep 002, only variables which have changed are output, in this case the concentrations of the 3 species in the transport solution. Finally, the set of files which gets generated is problem dependent, and so certain types of files may or my not appear. For example, since this problem has discrete fractures, we are seeing some fracture velocity and aperture output.

In order to conserve disk space and increase the efficiency of file I/O, these files are all stored in binary format, so they can not be viewed with a standard text editor. However, through the use of the post-processing tool **HSPLOT**, which is described in Section E, it is possible

to create ascii versions of the data contained in these files which are also compatible with the third-party visualization tools TECPLOT and GMS.

By default, simulation time is written in listing and output files in a fixed format using the FORTRAN F format descriptor f17.5, which is not suitable for outputting times greater than 999,999 or less than 0.00001. If you prefer to use scientific notation, you can choose it using the instruction:

> Time output scientific format

The other option is to use the FORTRAN G format descriptor (i.e. general format) in which a mix of fixed and scientific format is used depending on the magnitude of the output. If you prefer this notation, you can choose it using the instruction:

> Time output general format

Sice these instructions can be used in the `debug.control` file (see Section C), it might sometimes be usefult to switch back to fixed format. To do so, use the instruction:

> Time output fixed format

Similarly, mass balance output can be controlled by the instructions:

> Mass balance output scientific format

and

> Mass balance output general format
> Mass balance output fixed format

5.9.1   Grid

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 *prefix*o.eco file.

<div align="center">● ● ●</div>

Echo incidences
 Causes element incidences to be written to the *prefix*o.eco file.

<div align="center">● ● ●</div>

Echo fracture incidences

Causes fracture element incidences to be written to the *prefix*o.eco file.

<center>● ● ●</center>

List surface flow nodes

Causes the list of surface flow nodes to be written to the *prefix*o.eco file.

<center>● ● ●</center>

Echo element area numbers

Causes element area numbers (as read in Section 5.3.5 for a 2-D slice generated by GRID BUILDER) to be written to the *prefix*o.eco file.

<center>● ● ●</center>

Write faces and segments

Whenever **HydroGeoSphere** generates a 3-D 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 instruction to write the information to a file which will receive the name *prefix*o.fac. These files can become quite large, so the default is not to save them. See also Section 5.3.8.

<center>● ● ●</center>

5.9.1.1 GMS

The following instructions can be used to export data generated by **grok** (e.g. 2-D and 3-D meshes, wells, fractures etc.) to GMS.

Mesh to gms

1. **gmsfile** Name of the file to which the 3-D mesh information (blocks or prisms) will be written in GMS-readable format.

<center>● ● ●</center>

2D mesh to gms

1. **gmsfile** Name of the file to which the 2-D mesh information (quadrilaterals or triangles) will be written in GMS-readable format.

<center>● ● ●</center>

Rectangles to triangles

1. **ofile** Name of the file to which the 2-D triangular mesh information will be written in GMS-readable 2-D mesh file format. Each rectangle in the 2-D mesh is split into two triangles. Nodal coordinates are written first followed by element node lists.

• • •

The following instructions should be placed near the end of the *prefix*.grok after any instructions which are used to generate wells or fractures.

Wells to gms

1. **gmsfile** Name of the file to which the 1-D line information (wells and/or tile drains) will be written in GMS-readable borehole file format.

• • •

Fractures to gms

1. **gmsfile** Name of the file to which the the 2-D fracture element information which will be written in GMS-readable 2-D mesh file format.

• • •

5.9.1.2   GRID BUILDER

The following instructions can be used to export data generated by **grok** to GRID BUILDER.

Gms mesh to gb

Changes the behaviour of the instruction read 2d gms slice, causing it to output the 2-D GMS mesh (coordinates, element node lists and element zone numbers) to a file `prefixo.imp`, which can be imported into GRID BUILDER.

• • •

5.9.1.3   TECPLOT

The following instructions can be used to export data generated by **grok** to TECPLOT.

Mesh to tecplot

1. **tecfile** The name of the file to which the 3-D mesh information (blocks or prisms) will be written in TECPLOT-readable format.

• • •

Wells to tecplot

1. **tec_wells_file** Name of the file to which the 1-D well element information will be written in TECPLOT LINE3D geometry format.

• • •

Tiles to tecplot

1. **tec_tiles_file** Name of the file to which the 1-D tile drain element information will be written in TECPLOT LINE3D geometry format.

• • •

K to tecplot

1. **tecfile** The name of the file to which the hydraulic conductivity information (blocks or prisms) will be written in TECPLOT-readable format.

They are written as cell-based (i.e. elemental) variables for each of the principal coordinate directions.

• • •

Porosity to tecplot

1. **tecfile** The name of the file to which the porosity information (blocks or prisms) will be written in TECPLOT-readable format.

• • •

Tortuosity to tecplot

1. **tecfile** The name of the file to which the tortuosity information (blocks or prisms) will be written in TECPLOT-readable format.

• • •

5.9.1.4   IBM Data Explorer (OPENDX)

The following instructions can be used to export data generated by **grok** to IBM Data Explorer (OPENDX).

Mesh to opendx

1. **odxfile** Name of the file to which the 3-D mesh information (blocks or prisms) will be written in OPENDX readable format.

• • •

Wells to opendx

1. **odx_wells_file** Name of the file to which the 1-D well element information will be written in OPENDX format.

2. **odxfile** Name of the file which contains the 3-D mesh information, as given in the Mesh to opendx instruction.

<div align="center">• • •</div>

Tiles to opendx

1. **odx_tiles_file** Name of the file to which the 1-D tile element information will be written in OPENDX format.

2. **odxfile** Name of the file which contains the 3-D mesh information, as given in the Mesh to opendx instruction.

<div align="center">• • •</div>

5.9.2   Flow Solution

The following instructions affect the I/O for the flow solution.

Echo to output

Causes heads, saturations,concentrations and velocities for the subsurface domain to be written to the .lst file as well as to the binary output files.

<div align="center">• • •</div>

Flux output nodes

1. **new_noutfc** 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.

Listed nodes are flagged as flux output nodes, at which detailed fluid flux $[L^3 \ T^{-1}]$ information for the subsurface domain is output at each timestep. Flux output is written to a file called *prefix*o.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 4.4.1.

<div align="center">• • •</div>

Flux output nodes from chosen

This instruction works in a similar way to Flux output nodes except that all currently chosen nodes are flagged as flux output nodes.

● ● ●

Binary flux output nodes

This instruction works in a similar way to Flux output nodes and requires identical input. However, it creates a binary file called *prefix*o.flu_b.

● ● ●

Binary flux output nodes from chosen

This instruction works in a similar way to Binary flux output nodes except that all currently chosen nodes are flagged as flux output nodes.

● ● ●

Output saltwater head

By default, when running density-dependent flow problems, the equivalent freshwater head is written to the head file (e.g. f_cdo.head.001). This instruction causes salt water heads to be written instead.

● ● ●

5.9.2.1 Observation Wells And Points

The following instructions are used to output hydraulic head, saturation (if variably-saturated) and nodal fluid flux at a single node or a group of nodes lying on a line.

Output is written to a TECPLOT-formatted file called *prefix*o.observation_well_flow.-*obs_well_name*.dat.

If the problem is formulated as a dual continuum then extra columns of output will be produced with the dual continuum head, saturation (if variably-saturated) and fluid flux. If the node is not a dual continuum node, no data values (currently -999.0) will be written instead.

If the problem is formulated with a surface flow domain then extra columns of output will be produced with surface flow head and fluid flux. If the node is not a surface flow node, a no data value (currently -999.0) will be written instead.

These instruction do not affect the flow solution.

Make observation point

1. **well_name** Descriptive name for the well up to 40 characters.

2. **x1, y1, z1** *xyz*-coordinates of the observation point.

The node closest to this point will be flagged as an observation point node.

Data for each timestep in the flow solution is written as one line in the output file. Each line contains the time and then the porous medium hydraulic head, saturation (if variably-saturated), fluid flux, node coordinates and node number.

● ● ●

Make node observation point

1. **well_name** Descriptive name for the well up to 40 characters.

2. **nnumber** The number of the node to be made an observation point.

This node will be flagged as an observation point node.

Data for each timestep in the flow solution is written as one line in the output file. Each line contains the time and then the porous medium hydraulic head, saturation (if variably-saturated), fluid flux, node coordinates and node number.

● ● ●

Make observation well

1. **well_name** Descriptive name for the well up to 40 characters.

2. **x1, y1, z1** *xyz*-coordinates of one end of the observation well.

3. **x2, y2, z2** *xyz*-coordinates of the other end of the observation well.

Element face edges (i.e. segments) which form the shortest path between the two nodes closest to the end points of the well (i.e. **x1, y1, z1** and **x2, y2, z2**) will be chosen to produce output.

Each timestep in the flow solution represents a Tecplot zone in the output file, and each zone contains one line per observation well node. Each line contains the porous medium hydraulic head, saturation (if variably-saturated), fluid flux, node coordinates and node number.

Note that fluid fluxes at internal nodes are currently reported as zero unless they are constant head or specified flux nodes.

● ● ●

5.9.2.2   Fluid Mass Balance

By default, fluid mass balance information is computed at each time step and written to the *prefix*o.lst file. Figure 5.16 show some sample output as it appears in the .lst file.

```
-----------------------------------------------------------------------
 FLUID BALANCE, TIME:   0.500000000000000
-----------------------------------------------------------------------
RATE OF FLUID EXCHANGE                 IN               OUT
TOTAL
   Fixed flux                     0.0059395897      0.0000000000
   Critical depth                                   0.0000000000
   NET1 EXCHANGE RATE (IN-OUT)                                  0.0059395897


RATE OF FLUID ACCUMULATION [L**3/T]
   Porous medium                  0.0003197172
   Overland                       0.0057208226
   NET2 ACCUMULATION RATE                                       0.0060405397


FLUID BALANCE ERROR
   Absolute: (NET1-NET2)                                       -0.0001009500
   Relative: (NET1-NET2)/(abs(NET1)+abs(NET2))/2.0              0.0168529029
   Percent:   abs(NET1-NET2)/NET1(+ve)*100.0d0        1.6996119929


FLUID EXCHANGE BETWEEN SURFACE AND SUBSURFACE DOMAIN
  Infiltration                    0.0003219343
  Exfiltration                    0.0000000000
```

Figure 5.16: Sample Fluid Balance Information for Example Abdul.

This detailed fluid balance information is also written to a TECPLOT formatted ascii output file called *prefix*o.water_balance.dat so that it can be easily visualized. For each timestep in the flow solution one line is written to the file. The number of columns in the file depends on the nature of the problem. For example, for problems with no overland flow component, no overland flow data will be written to the file.

The first section of the fluid balance information relates to the rate at which water is entering (IN) or leaving (OUT) the domain via the various types of sources and sinks. Column headings that could appear in this section are:

- **1st+ 1st-** The rate at which water is entering or leaving the domain via porous media first-type nodes.

- **1stdual+ 1stdual-** The rate at which water is entering or leaving the domain via dual continuum first-type nodes.

- **1stSurface+ 1stSurface-** The rate at which water is entering or leaving the domain via overland flow first-type nodes.

- **2nd+ 2nd-** The rate at which water is entering or leaving the domain via second-type nodes.

- **EvapoTrans-** The rate at which water is leaving the domain via evapotranspiration.

- **Well+ Well-** The rate at which water is being added to or removed from the domain via wells.

- **Tile+ Tile-** The rate at which water is being added to or removed from the domain via tile drains.

- **River+ River-** The rate at which water is entering or leaving the domain via river nodes, as defined in Section **??**.

- **Drain-** The rate at which water is entering or leaving the domain via drain nodes, as defined in Section **??**.

- **Seep-**The rate at which water is entering or leaving the domain via seepage faces.

- **ZeroDepth-** The rate at which water is leaving the domain via zero-depth gradient boundaries.

- **CritDepth-** The rate at which water is leaving the domain via critical depth boundaries.

- **FreeDr-** The rate at which water is leaving the domain via free drainage boundaries.

- **FixedFlow+ FixedFlow-** The rate at which water is added to or removed from the domain via specified flowrate boundaries, as defined in Section 5.7.2.6.

- **NET1 Sources/Sinks** The total rate of all water entering or leaving the domain through sources and sinks, $\pm Q$.

The next section of the fluid balance information relates to the changes in storage that occurred in the various media. Column headings that could appear in this section are:

- **PM** Rate of accumulation in the porous medium.

- **Overland** Rate of accumulation in the overland flow domain.

- **Dual** Rate of accumulation in the dual continuum.

- **Fracture** Rate of accumulation in discrete fractures.

- **Tiles** Rate of accumulation in tile drains.

- **Wells** Rate of accumulation in wells.

- **NET2 Accumulation** The rate of change of storage for the entire domain ΔS.

In a perfectly balanced system the rate at which water enters or leaves the domain $\pm Q$ and the rate of change in storage ΔS would be equal and the fluid balance error ε would be:

$$\varepsilon = \pm Q - \Delta S = 0 \tag{5.13}$$

The next section of the fluid balance information relates to the fluid balance error, which can be expressed in various ways:

- **ERROR (NET1-NET2)** The absolute error ε.

- **Error rel** The relative error ε_R:

$$\varepsilon_R = \frac{\varepsilon}{(|\pm Q| + |\Delta S|)/2} \tag{5.14}$$

- **Error percent** The percent error ε_P:

$$\varepsilon_P = \frac{|\varepsilon|}{Q^{positive}} 100 \tag{5.15}$$

The final section of the fluid balance output gives the amount of water that has moved between the overland flow domain and the subsurface domain:

- **Infilt** The amount of water that has infiltrated the subsurface (i.e moved from overland flow domain to subsurface domain).

- **Exfilt** The amount of water that has discharged from the subsurface (i.e moved from the subsurface domain to the overland flow domain).

5.9.3   Surface Flow Hydrographs

The following instructions affect the I/O for the surface flow solution.

Set hydrograph nodes

1. **label_hyd** Descriptive name for the hydrograph up to 80 characters.

Chosen nodes are flagged as hydrograph nodes, at which detailed total fluid flux $[L^3 \ T^{-1}]$ information is output at each timestep. Output is written to a TECPLOT-formatted file called *prefix*o.hydrograph.*hydrograph*.dat. For each timestep in the flow solution, one line will be written to the file. Each line contains the time, surface domain flux, porous media flux and total flux.

• • •

5.9.4   Transport

The following instructions affect the I/O for the transport solution.

Flux output nodes

1. **new_noutfc** Number of new output nodes desired. Read the following **new_noutfc** times:

   (a) **ioutfc** Flux output node number.

Listed nodes are flagged to generate detailed mass flux $[M \ T^{-1}]$ information to the file *prefix*o.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 Section 4.4.1.

• • •

Flux output nodes from chosen
This instruction works in a similar way to Flux output nodes except that all currently chosen nodes are flagged as flux output nodes.

• • •

Flux volume output nodes

1. **fvol_name** Descriptive name for the volume up to 40 characters.

Chosen nodes are used to define a volume for which detailed mass flux information will be written to the Tecplot-formatted file *prefix*o.mass_entering_volume.*volume.species*.dat, where the *prefix*, *volume* and *species* portions would be the user-defined prefix, volume name and species (i.e. solute) name respectively. For each timestep in the flow solution, one line will be written to the file. Each line contains the time, the average concentration and instantaneous mass flux for all media (i.e. porous, dual , fractured or overland) and the cumulative mass flux for the volume.

Active nodes (the surface of the volume) and contributing nodes (just outside the surface of the volume) are defined automatically by searching for 3-D elements that contain a mixture of chosen and unchosen nodes. For such elements, chosen nodes are flagged as active and unchosen nodes are flagged as contributing.

Mass fluxes reported for the volume are positive if mass is entering the volume and negative if mass is leaving the volume.

● ● ●

In some cases, it may be necessary to restrict the computation of mass flux to a slice of nodes. The following instructions can be used to do this:

Slice flux output nodes from chosen

1. **fvol_name** Descriptive name for the volume up to 40 characters.

This instruction works in a similar way to Flux volume output nodes except that all currently chosen nodes are flagged as active nodes for the mass flux calculation.

● ● ●

The preceding instruction must be used in conjunction with the following instruction:

Slice flux contributing nodes from chosen

This instruction flags all currently chosen nodes as contributing nodes for the mass flux calculation.

grok checks that both instructions are issued in the correct order, and that there is at least one element that shares an active and contributing node.

● ● ●

Plot concentration penetration depth

1. **threshold_conc** Causes **HydroGeoSphere** to write the distance from domain top to the elevation of the threshold concentration contour versus time to a file called *prefix*o.penetration_depth.

This is especially useful in conjunction with density-dependent flow simulations where the migration of a dense plume from the domain top into the aquifer versus time is of interest.

● ● ●

Plot maximum velocity

Causes **HydroGeoSphere** to write the maximum porous matrix velocity versus time to a file called *prefix*o.maxvel_pm. If fractures are present, the additional file *prefix*o.maxvel_fm is created, containing the maximum fracture velocity versus time data.

● ● ●

5.9.4.1   Observation Wells And Points

The following instructions are used to output concentration at a single node or a group of nodes lying on a line.

These instruction do not affect the transport solution.

Make observation point

1. **well_name** Descriptive name for the well up to 40 characters.

2. **x1, y1, z1** *xyz*-coordinates of the observation point.

The node closest to this point will be flagged as an observation point node.

Output is written to a TECPLOT-formatted file called *prefix*o.observation_well_conc.-*obs_well_name*.dat. Data for each timestep in the flow solution is written as one line in the output file. Each line contains the time and porous medium concentration (and immobile zone concentration if dual porosity).

If the problem is formulated as a dual continuum then an extra column of output will be written with the dual continuum concentration. If the node is not a dual continuum node, a no data value (currently -999.0) will be written instead.

If the problem is formulated with a surface flow domain then an extra column of output will be produced with the surface flow concentration. If the node is not a surface flow node, a no data value (currently -999.0) will be written instead.

The line ends with the node coordinates and number.

● ● ●

Make observation well

1. **well_name** Descriptive name for the well up to 40 characters.

2. **x1, y1, z1** *xyz*-coordinates of one end of the observation well.

3. **x2, y2, z2** *xyz*-coordinates of the other end of the observation well.

Element face edges (i.e. segments) which form the shortest path between the two nodes closest to the end points of the well (i.e. **x1, y1, z1** and **x2, y2, z2**) will be chosen to produce output.

Output is written to a TECPLOT-formatted file called *prefix*o.`observation_well_flow`.-*obs_well_name*.`dat`. Each timestep in the flow solution represents a Tecplot zone in the output file, and each zone contains one line per observation well node. Each line contains the porous medium concentration (and immobile zone concentration if dual-porosity).

If the problem is formulated as a dual continuum then an extra column of output will be written with the dual continuum concentration. If the node is not a dual continuum node, a no data value (currently -999.0) will be written instead.

If the problem is formulated with a surface flow domain then an extra column of output will be produced with the surface flow concentration. If the node is not a surface flow node, a no data value (currently -999.0) will be written instead.

The line ends with the node coordinates and number.

• • •

5.9.4.2   Solute Mass Balance

By default, solute mass balance information for each species is computed at each time step and written to the *prefix*o.`lst` file. Table 5.17 show some sample output as it appears in the `.lst` file.

No solute mass balance
This instruction prevents solute mass balance information from being written to the listing file.

• • •

This detailed mass balance information is also written to a TECPLOT formatted ascii output file called, for example, *prefix*o.`mass_balance_`*species_001* so that it can be easily visualized.

Summary mass balance information is written to a TECPLOT formatted ascii output file called, for example, *prefix*o.`mass_balance_summary` so that it can be easily visualized. For each timestep in the transport solution, one line per species is written to the file. Each line has 7 columns labelled as follows:

1. **time** Simulation time.

2. **Tmass** Total mass in the system.

3. **dBoundary** Change in mass due to sources and sinks.

4. **dStored** Change in mass stored.

```
********************************************************************************
MASS BALANCE SPECIES  1 - Uranium
********************************************************************************
 (Time =   1999.00999999978      Time-step =  9.999999776482582E-003 )
RATE OF MASS EXCHANGE                   IN                  OUT
TOTAL
   Fixed head nodes           100.0000006818       0.0000000000
   Fixed concentration nodes  150.0985515732       0.0000000000
   NET1 EXCHANGE RATE (IN-OUT)                                  250.0985522550

MASS ACCUMULATION                  COMPONENT         SUBTOTAL
   In storage:
      Porous medium            2.5001748233
      Porous medium (sorbed) 35749.9997988755
   ACCUMULATED                                  35752.4999736989
   Lost by decay:
      Porous medium            0.0000000708
      Porous medium (sorbed)   0.0010116896
   DECAYED                                          0.0010117604
   EXCHANGED (via chemical reactions)               0.0000000000
   INITIAL (stored at start of timestep)        35750.0000000000
   NET2 RATE OF ACCUMULATION                                    250.0985515100
      i.e. (ACCUMULATED -DECAYED +(-)EXCHANGED -INITIAL)/delta

   NET (since start of simulation)              250.0985515100

MASS BALANCE ERROR
   Absolute: (NET1-NET2)                                        0.0000007450
   Relative: (NET1-NET2)/(abs(NET1)+abs(NET2))/2.0              0.0000000015
********************************************************************************
```

Figure 5.17: Sample Mass Balance Information for Example PM_CD.

5. **dBoundary - dStored** Error.

6. **(dBoundary - dStored)/Tmass*100)** Normalized error.

7. **Species** Species number.

Besides being plotted with TECPLOT, such output can be imported into an editor (e.g. Microsoft Excel) and sorted to facilitate, for example, the creation of a plot of mass balance error versus time for a given species.

Cumulative mass balance information is written to a TECPLOT formatted ascii output file called, for example, *prefix*o.mass_balance_cumulative so that it can be easily visualized. For each timestep in the transport solution, one line per species is written to the file. Each line has 7 columns labelled as follows:

1. **time** Simulation time.

2. **In (cumulative)** Mass in since start of simulation.

3. **Out (cumulative)** Mass out since start of simulation.

4. **Stored** Change in mass stored since start of simulation.

5. **In+Out-Stored** Error.

6. **Species** Species number.

Besides being plotted with TECPLOT, such output can be imported into an editor (e.g. Microsoft Excel) and sorted to facilitate, for example, the creation of a plot of mass balance error versus time for a given species.

5.9.4.3 Flux-averaged Concentration at a Well

Using either of the instructions Make well or Make infilled well causes **HydroGeoSphere** to create a file called *prefix*o.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.

5.9.4.4 Travel Time Probability

Output travel time statistics
HydroGeoSphere will perform descriptive statistics, following Eqs. (2.138) and (2.139): mean travel time, mode and standard deviation will be calculated at each node/element.

$\bullet\bullet\bullet$

Integrate production zone

1. **fname** Name of the file which contains the list of elements that contain a mass source function and the tabulated functions. It is formatted as follows:

   (a) **nprodel** Number of production elements. Read the following **nprodel** times:
      i. **nel**, **ifunc** The element number and the ID (number) of the associated tabulated function.

   (b) **maxdatprod, delta_conv** Size of the largest set of tabulated data which follows and the timestep size **delta_conv** for evaluating the convolution integral in equation 5.16. Read the following for each of the **ifunc** time-series:
      i. **ndata** Number of data to read for the current time-series.
      ii. **time**, **val** Time and corresponding source value.

If **ifunc** $= 0$, then m^* corresponds to a unit and instantaneous mass input function, and thus no time-series are required in the input file.

The element **nel** with the maximum value of **ifunc** determines how many sets of time-series data must be supplied.

This option is meant to simulate a forward transport solution by means of a backward solution. It requires the problem to be backward-in-time. Integration of the backward travel time PDF's will be performed over a series of element numbers, which are input in the `.np` file. In a forward transport run, these elements would contain a mass source m^*. The following equation is solved at each time-step in order to simulate the output mass flux $J_O(t)$ resulting from a forward transport (see [*Cornaton*,2003]):

$$J_O(t) = \int_\Delta \int_0^t g_t(t - u, \boldsymbol{x}) m^*(\boldsymbol{x}, u)\, du\, d\Omega \qquad (5.16)$$

if m^* is an arbitrary mass source function $[\mathrm{ML}^{-3}\mathrm{T}^{-1}]$, or

$$J_O(t) = \int_\Delta g_t(t, \boldsymbol{x}) \delta(\boldsymbol{x} - \boldsymbol{x}_i)\, d\Omega \qquad (5.17)$$

if m^* is a unit and instantaneous mass input function $[\mathrm{L}^{-3}\mathrm{T}^{-1}]$. Δ denotes the domain of elements where $m^*(\boldsymbol{x}, t) \neq 0$.

$\bullet\bullet\bullet$

Chapter 6

Illustrative Examples

6.1 Travel Time Probability

6.1.1 Capture zone probability of a pumping-well

In this section, an illustration of the capture zone probability definition by using the boundary value problem (2.148) is given. The theoretical system corresponds to a uniformly recharged aquifer (rainfall infiltration of 0.432 m/year), containing a single extraction well (extraction rate of 300 m^3/year), and with a natural outlet to which a uniform hydraulic head of 32 m is prescribed (see Fig. 6.1). The aquifer is homogeneous with respect to hydraulic conductivity, porosity and dispersivity. Flow is at steady-state. Figure 6.1 shows the well capture zone probability at time $t = 1, 5$ and 50 days, the date 50 days providing a probability field close to steady-state (absolute capture zone).

Figure 6.2 shows the solution of Eqs. (2.140) and (2.142). Figure 6.2a provides the average time a water particle will take prior to exiting at the pumping-well, exclusively. To do so, Eq. (2.142) has been solved by assigning $\langle E \rangle = 0$ at the well and a zero flux at the natural outlet. This solution can be combined with the capture zone probability distribution, for well protection zone definition purposes.

Figure 6.2b shows the mean age distribution at aquifer scale, figure 6.2c the mean life expectancy distribution, and figure 6.2d gives the mean total transit time (from inlet to outlets) distribution ($\langle T \rangle = \langle A \rangle + \langle E \rangle$).

6.2 Simulating Tidal Fluctuation

This example shows the use of historical tide data as a boundary condition on a simple sloping surface water domain.

This data was obtained from the Center for Operational Oceanographic Products and Ser-
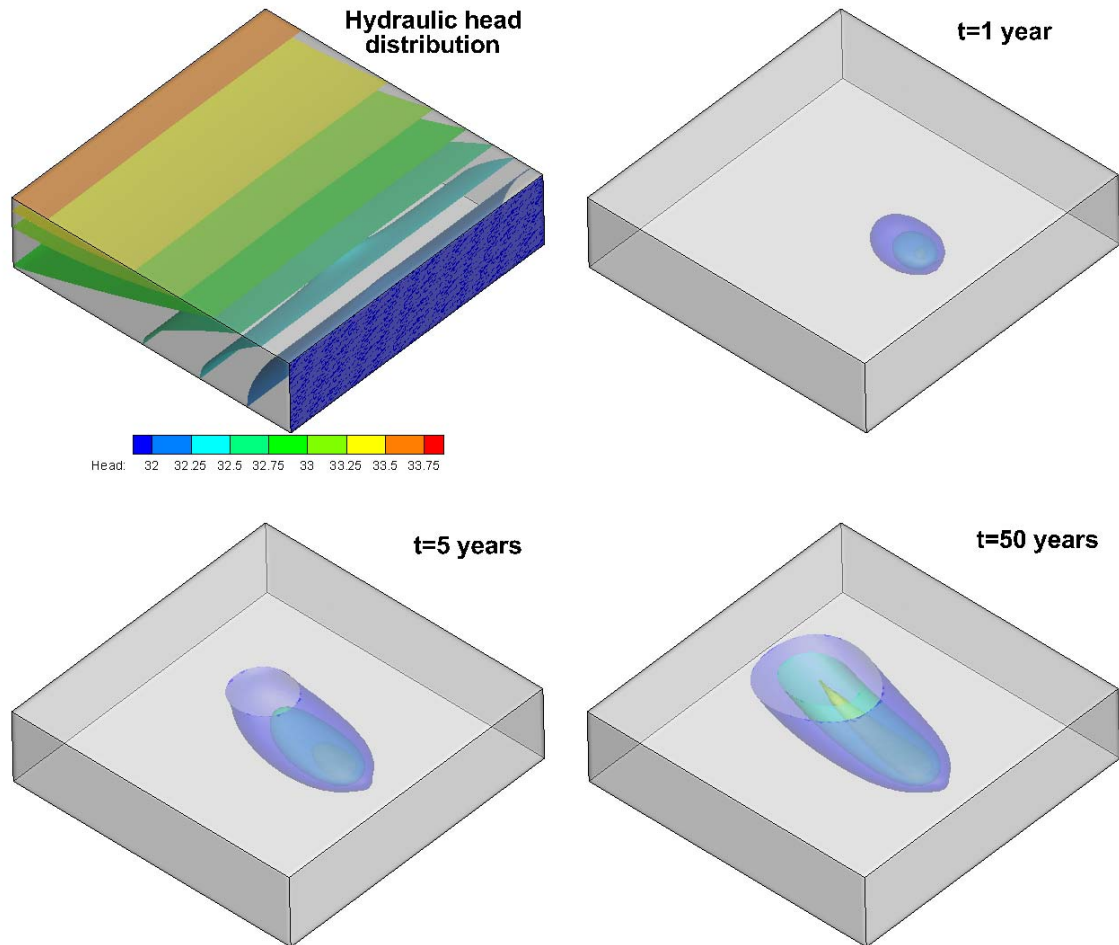
Figure 6.1: Pumping-well temporal capture zone probability. The aquifer size is $128 \times 128 \times 32$ m. Iso-probability surfaces 0.1-0.5-0.9.
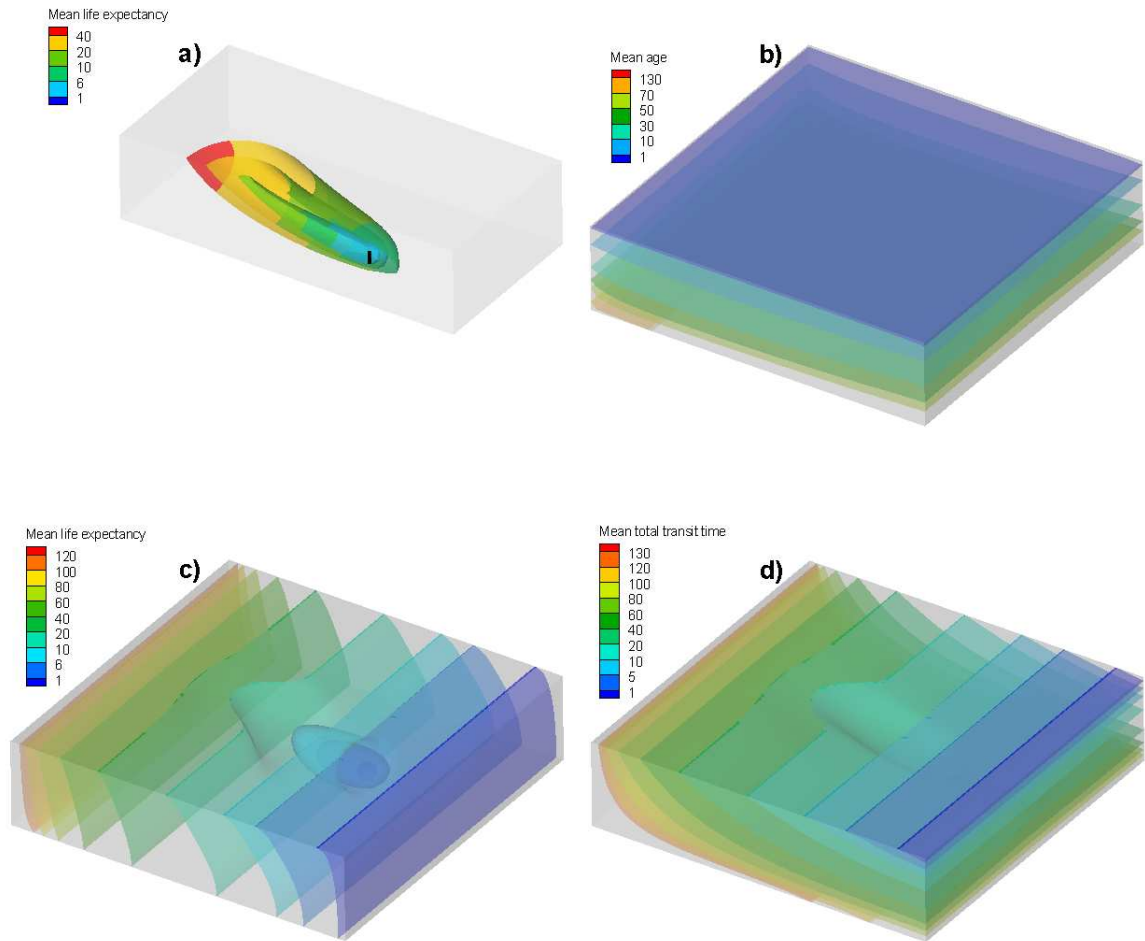
Figure 6.2: Temporal moment solutions in days: (a) Mean life-expectancy-to-well distribution; (b) Mean age; (c) Mean life expectancy; (d) Mean total transit time.

vices (http://tidesandcurrents.noaa.gov/index.shtml) for the Station at Redwood City, California. A portion of the raw data file (`redwood_hl_dec12006.orig`) is shown here:

```
Station ID: 9414523 Page Help

Historic Tide Data
Station Date      Time   Vrfy 6
DCP#:
Units:                   Meters
Data%:  MSL        LST    98.92
Maximum:                  1.545
Minimum:                 -1.903
------- -------- ----- -------
9414523 20061201 00:00
9414523 20061201 00:06
...
9414523 20061201 02:18
9414523 20061201 02:24  -1.033
9414523 20061201 02:30
...
```

The header contains important information about the data, such as the units (metres) and the reference elevation (MSL or mean sea level). This data file contains daily minimum and maximum tide elevations at 6-minute intervals so many of the records do not contain an elevation value. These records are ignored by **grok**.

This data can be easily modified for use with **grok** by stripping out the header and footer information and leaving only the data records. This has been done in file `redwood_hl_dec12006.tide_data`. The beginning of the file is shown here:

```
9414523 20061201 00:00    0.0
9414523 20061201 00:06
...
9414523 20061201 02:18
9414523 20061201 02:24  -1.033
9414523 20061201 02:30
...
```

Note that we set the initial tide elevation value to zero in the first record of the data.

Figure 6.3 shows the surface water domain and finite-element mesh. It is a tilted strip 100 m long and 1 m wide, which slopes from $z = -2$ m at $x = 0$ to $z = 2$ m at $x = 100$m.

Here is a portion of the main input file, `tide.grok`, where we assign the tidal boundary condition:

```
clear chosen nodes
```
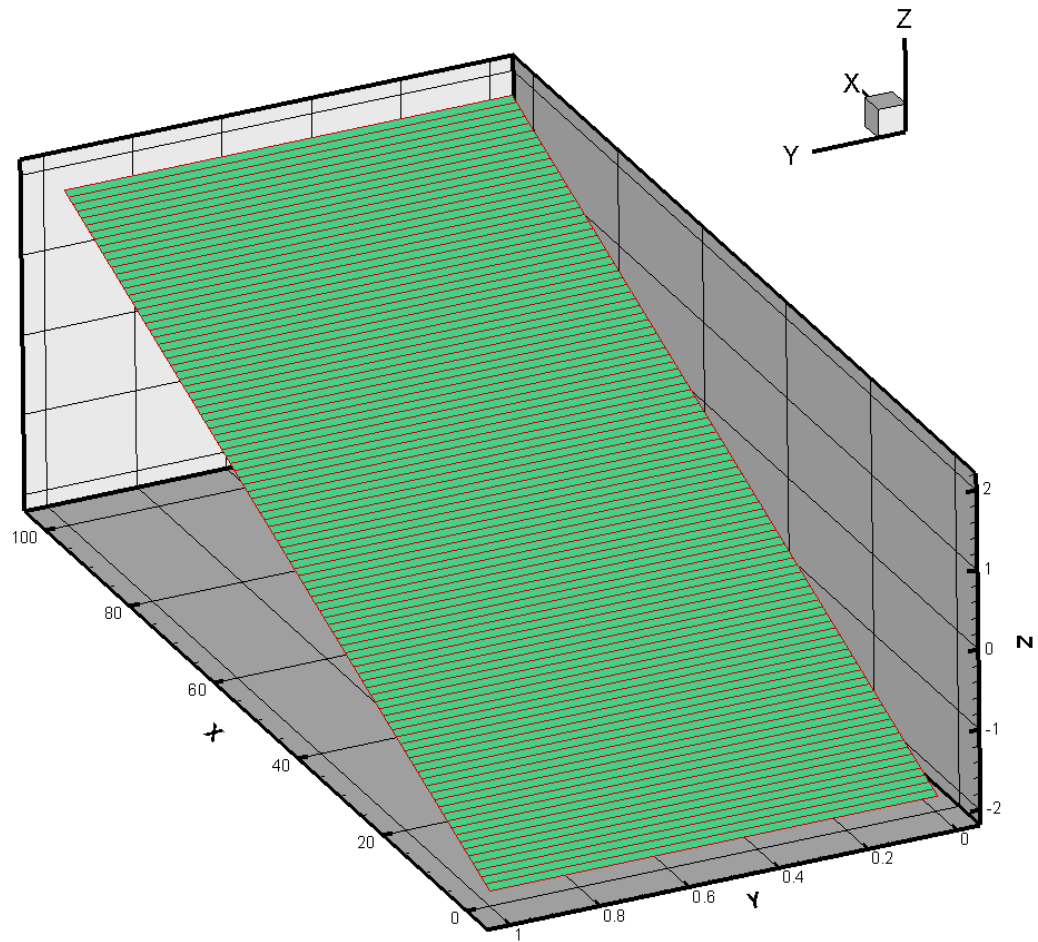
Figure 6.3: Surface domain and mesh.

```
choose nodes block
0.0 0.0
0.0 1.0
-2.0 -2.0

interpolate specified head
.true.

specified head from tidal data
redwood_hl_dec12006.tide_data
20061201 00:00

echo flow boundary conditions

output times
4.7
end
```

Note that we have chosen to index the time to zero at `20061201 00:00`, the first record.

We have chosen to end the simulation at time 4.7 days, as defined by the final (and only) entry to the output times instruction.

In this problem, time units are in days because we included the instruction units: kilogram-metre-day.

The resulting head function can be examined by inserting the instruction echo flow boundary conditions, which gives the following output:

```
INSTRUCTION: echo flow boundary conditions
 Number of prescribed head nodes            2
   Node       Head          Time on        Time off
    203     0.0000         0.0000        0.10001
           -1.0330         0.10001       0.36667
            1.2350         0.36667       0.66251

            ...
           -1.8740         4.8000        0.10000E+21
    304     0.0000         0.0000        0.10001
           -1.0330         0.10001       0.36667
            1.2350         0.36667       0.66251

            ...
           -1.5220         0.66251       0.92084
           -1.8740         4.8000        0.10000E+21
 No prescribed flux nodes
 No prescribed time-varying flux faces
```

Figure 6.4 shows the initial water surface (translucent blue) superimposed on the ground
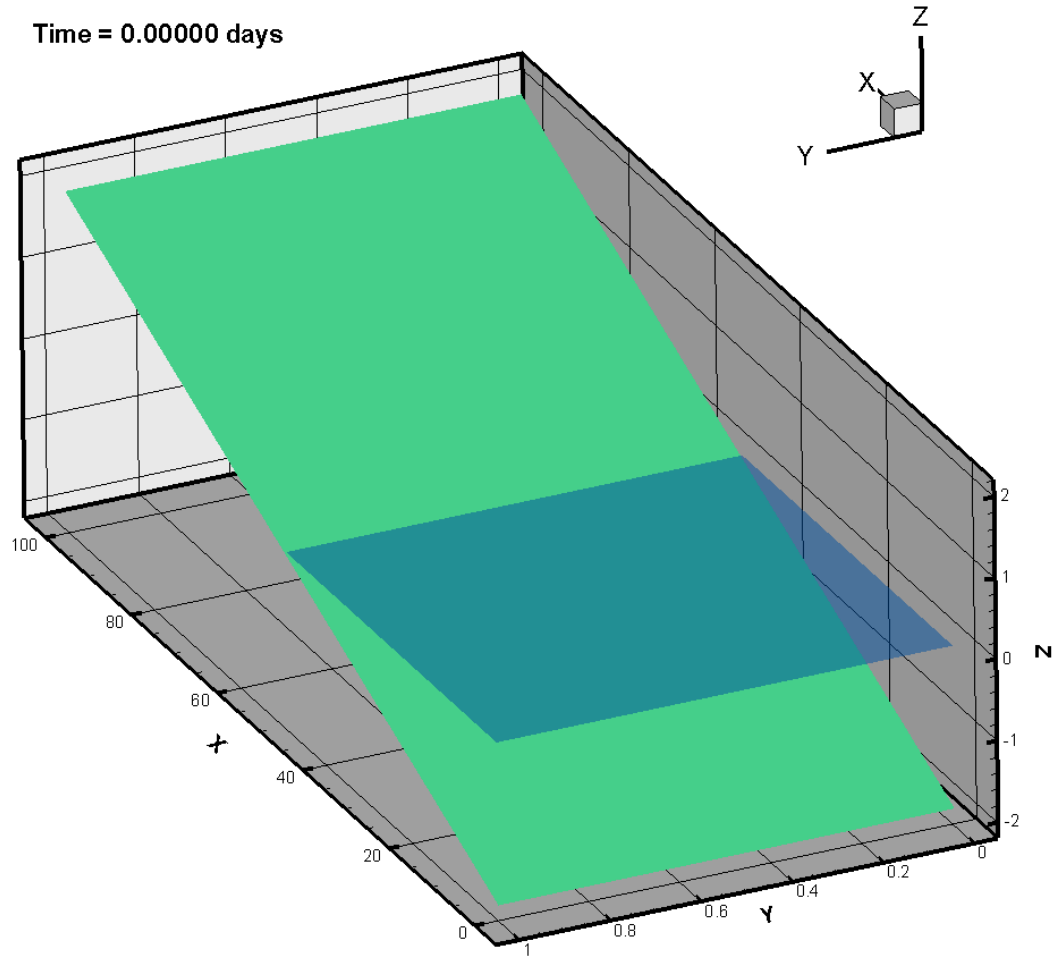
Figure 6.4: Tide level.

surface (green). This figure was produced by Tecplot using the layout file `tide.lay`. It consists of two identical frames which are superimposed. The background frame contains the ground surface, which is just the finite element mesh shaded green. The foremost frame, representing the water surface, has no background (so it doesn't obscure the ground surface) and was created by assigning the surface domain head as the z-coordinate and value blanking for surface water depths less than 1 cm.

An animated movie of the tidal fluctuation can be found in the file `tide.avi`. Rather than produce multiple output files by adding output times, we modified the `debug.control` file so that an output file was produced for each timestep.

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.

Biot, M.A., 1941. General Theory of Three-Dimensional Consolidation. J. Applied Physics, 12(2):155-164.

Brooks, R.J., and A. T. Corey, 1964. Hydraulic properties of porous media. Hydrology paper 3, Colorado State university, Fort Collins, CO.

Canadell, J., R.B. Jackson, J.R. Ehrlinger, H.A. Mooney, O.E. Sala and E.D. Schulze, 1996. Maximum rooting depth of vegetation types at the global scale, Oecologia, 108:583-595.

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.

Dingman, S.L., 1994. Physical Hydrology, 575 pp., Maxmillian, New York.

Feddes, R.A., P.J. Kowalik and H. Zaradny, 1978. Simulation of field water use and crop yield. New York: John Wiley and Sons.

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.

Forsyth, P.A., Wu, Y.S. and K. Pruess, 1995. Robust numerical methods for saturated-unsaturated flow with dry initial conditions in heterogeneous media, Adv. Water Res., 18(1), 25–38.

Forsyth, P.A., and M.C. Kropinski, 1997. Monotonicity considerations for saturated-unsaturated subsurface flow, SIAM J. Sci. Comp., 18, 1328–1354.

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., S.E. Jones 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., S.E. Jones and M.L. Kavvas, 1988b. On the Diffusion Wave Model for Overland Flow 2. Steady State Analysis, Water Resour. Res., 24(5), 745–754.

Guvanasen, V., 2007. FRAC3DVS Enhancements: Subgridding, Hydromechanical Deformation, and Anisotropic Molecular Diffusion. Report No: In preparation. Ontario Power Generation, Nuclear Waste Management Division, Toronto, Ontario, Canada, M5G 1X6.

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.

Kristensen, K.J. and S.E. Jensen. 1975. A model for estimating actual evapotranspiration from potential evapotranspiration. Nordic Hydrol., 6:170-88.

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.

Letniowski, F.W. and P.A. Forsyth,1991. A control volume finite element method for three-dimensional NAPL groundwater contamination, Int. J. Num. Meth. Fluids, 13, 955.

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.

Monteith, J.L., 1981. Evaporation and surface temperature. Q. J. R. Meteorol. Soc., 107:1-27.

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.

Neuzil, C.E., 2003. Hydromechanical coupling in geological processes, Hydrogeology Journal, 11:41–83.

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.

Ogata, A., and R.B. Banks, 1961. A solution of the differential equation of longitudinal dispersion in porous media. U.S. Geol. Surv. Prof. Papper 411-A.

Panday S. and P.S. Huyakorn, 2004. A fully coupled physically-based spatially-distributed model for evaluating surface/subsurface flow. Advances in Water Resources, 27:361-382.

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.

Provost, A.M., Voss, C.I. and Neuzil, C.E, 1998. Glaciation and regional ground-water flow in the Fennoscandian Shield; Site 94, Swedish Nuclear Power Inspectorate, SKI Report 96:11, Stockholm, Sweden.

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.

Scurlock, J.M.O., G.P. Asner, and S.T. Gower, 2001. Worldwide Historical Estimates of Leaf Area Index, 1932–2000, prepared for the Oak Ridge National Laboratory, Oak Ridge, Tennessee. ORNL/TM-2001/268.

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., 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.

Wigmosta M.S., L.W. Vail and D.P. Lettenmaier, 1994. A distributed hydrology-vegetation model for complex terrain. Water Resour. Res., 30(6):1665–1679.

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.

Yang, J., R.N. Edwards, 2000. Predicted groundwater circulation in fractured and unfractured anisotropic porous media driven by nuclear fuel waste heat generation. Canadian Journal of Earth Sciences, 37: 1301–1308.

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

Mathematical Notation

The following general rules can be applied throughout this appendix:

- 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.

- Symbolˆis 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	Subsurface-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].
B_f	Uniform spacing for a set of parallel fractures [L].
C	Subsurface, solute concentration [M L^{-3}].

Variants:

C_d	Dual continuum.
C_{Imm}	Double-porosity immobile region.
C_f	Fracture.
C_o	Surface(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	Channel.

$C_{c_{\mathrm{Inj}}}$ Channel, injected water.

C_h	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}].
$\mathbf{D}$	Subsurface, hydrodynamic dispersion tensor [L^2 T^{-1}].
	Variants:

         $\mathbf{D_d}$ Dual continuum.
         $\mathbf{D_f}$ Fracture.
         $\mathbf{D_o}$ Surface(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	Surface(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	Subsurface, hydraulic head [L].
	Variants:

         h_d Dual continuum.
         h_f Fracture.
         h_o Surface(overland) flow, water surface elevation.
         h_t Tile drain.
         h_w Well.

H_d	Surface(overland) flow, depression storage height [L].
H_o	Surface(overland) flow, obstruction storage height [L].
H_s	Surface(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.
$\mathbf{k}$	Subsurface, permeability tensor [L^2].
$\mathbf{K}$	Subsurface, saturated hydraulic conductivity tensor [L T^{-1}].
K^*	Component of hydraulic conductivity tensor normal to a seepage face.
K'	Subsurface, equilibrium distribution coefficient [L^{-3} M].
	Variants:

	K'_d	Dual continuum.
	K'_f	Fracture.
K_a	Subsurface-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}].	
$\mathbf{K_o}$	Surface(overland) flow, conductance tensor[L T^{-1}].	
k_r	Subsurface, relative permeability [dimensionless].	

Variants:

	k_{rd}	Dual continuum.
	k_{rf}	Fracture.
	k_{ro}	Surface(overland) flow.
	k_{rt}	Tile drain.
	k_{rw}	Well.

k_{ra}	Subsurface-macropore coupling, interface relative permeability [dimensionless].
k_{rso}	Subsurface-surface(overland) flow coupling, rill effect term [dimensionless].
K	Bulk modulus of porous media [M T^{-2} L^{-1}].
K_f	Bulk modulus of fluid [M T^{-2} L^{-1}].
K_{ss}	Bulk modulus of solids [M T^{-2} L^{-1}].
K_s	Surface flow, conductance term reduction factor [dimensionless].
K_{so}	Subsurface-surface (overland) flow coupling, leakance term [T^{-1}].
K_t	Tile drain, hydraulic conductivity [L T^{-1}].
K_w	Well, saturated hydraulic conductivity [L T^{-1}].
K_{ox}	Surface(overland) flow, conductance in $x-$ direction [L T^{-1}].
K_{oy}	Surface(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_p	Pore-connectivity parameter for unsaturated functions [dimensionless].
L_s	Well, screen length [L].
N	Finite element basis function [dimensionless].
n	Manning roughness coefficient [T L$^{-1/3}$].
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}].
$\mathbf{q}$	Subsurface, fluid flux [L T^{-1}].

Variants:

	$\mathbf{q_d}$	Dual continuum.
	$\mathbf{q_f}$	Fracture.
	$\mathbf{q_o}$	Surface(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_w	Well, wetted perimeter [L].
P_t	Tile drain wetted perimeter [L].
Q	Subsurface, fluid source or sink [T^{-1}].
Q_c	Channel, fluid source or sink [L^3 T^{-1}].
Q_{cd}	Dual continuum, solute source or sink [T^{-1}].
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	Surface(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	Subsurface, 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	Surface(overland) flow, coordinate along direction of maximum ground surface slope [L].
S_e	Effective saturation [dimensionless].
S_{fx}	Surface(overland) flow, friction slope in the $x-$direction [dimensionless].
S_{fy}	Surface(overland) flow, friction slope in the $y-$direction [dimensionless].
S_{ox}	Surface(overland) flow, bed slope in the $x-$direction [dimensionless].
S_{oy}	Surface(overland) flow, bed slope in the $y-$direction [dimensionless].
S_s	Subsurface, specific storage [L^{-1}].
S_{sf}	Fracture, specific storage [L^{-1}].
S_w	Subsurface, 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].
$\overline{v}_{io}$	Surface(overland) flow, vertically averaged flow velocity in the coordinate direction i [L T^{-1}].
$\overline{v}_{xo}$	Surface(overland) flow, vertically averaged flow velocity in the $x-$direction [L T^{-1}].
$\overline{v}_{yo}$	Surface(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	Subsurface, 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].
z	Subsurface, elevation head [L].
	Variants:
	z_d — Dual continuum.
	z_f — Fracture.
	z_t — Tile drain.
	z_w — Well.
z'	Depth coordinate from the soil surface [L].
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 [T^{-1}].
α_l	Subsurface, longitudinal dispersivity [L].
	Variants:
	α_{ld} — Dual continuum.
α_s	Subsurface-macropore coupling, first-order mass transfer [T^{-1}].
α_t	Subsurface, transverse dispersivity [L].
	Variants:
	α_{td} — Dual continuum.
α_w	Water, compressibility [L T^2 M^{-1}].
α_m	Matrix compressibility [L T^2 M^{-1}].
α_{ss}	Solids compressibility [L T^2 M^{-1}].
α_{wd}	Subsurface-macropore coupling, first-order fluid exchange coefficient [L^{-1} T^{-1}].
α_{wd}^*	Subsurface-macropore coupling, geometric factor [L^{-2}].
α'	$= 1 - K/K_s$ [dimensionless]
β	Van Genuchten parameter [dimensionless].
β_d	Subsurface-macropore coupling, geometrical factor [dimensionless].

β_{ske}	Skempton's coefficient [dimensionless]
γ	Water, kinematic viscosity [L^2 T^{-1}].
γ_{ij}	Term describing fluid flow between nodes i and j. [L^2 T^{-1}]
Γ_{ex}	Subsurface, 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	Surface(overland) flow.
	Γ_t	Tile drain.
	Γ_w	Well.

γ_w	Subsurface-macropore coupling, empirical constant [dimensionless].
δ	Dirac delta function.
ΔS_G	Subsurface, change in water storage [L^3].
ΔS_S	Surface 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	Subsurface, a set of nodes connected to node i.

Variants:

	η_{di}	Dual continuum.
	$\eta_{f\,i}$	Fracture.
	$\eta_{w\,i}$	Well.
	$\eta_{t\,i}$	Tile drain.

θ	Subsurface, water content [dimensionless].
θ_{Imm}	Double-porosity, porosity of the immobile region [dimensionless].
θ_s	Subsurface, saturated water content [dimensionless].

Variants:

	θ_{sd}	Dual continuum.

λ	Subsurface, solute first-order decay constant [L^{-1}].

Variants:

	λ_d	Dual continuum.
	λ_f	Fracture.
	λ_o	Surface(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].
ν^*	Poisson's ratio [dimensionless].
π	The constant $3.1415\cdots$. Would you prefer apple or cherry? Ice cream with that?
ψ	Subsurface, 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	An assigned pressure head [L].	
ρ	Water, density [M L^{-3}].	
ρ_b	Subsurface, bulk density [M L^{-3}].	
	Variants:	
	ρ_{bd}	Dual continuum.
$\sigma(r)$	Van Leer flux limiter [dimensionless] with smoothness sensor r.	
σ_{zz}	Surface vertical stress [M T^{-2} L^{-1}]	
τ	Subsurface, matrix tortuosity [dimensionless].	
	Variants:	
	τ_d	Dual continuum.
ϕ_o	Surface(overland) flow, surface porosity [dimensionless].	
χ	Water, surface tension [M T^{-2}].	
Ω_{ex}	Subsurface, 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	Surface(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.	
ζ	Loading efficiency [dimensionless].	

Appendix B

Output files

Unless otherwise stated, these files are ascii formatted.

These files of general use are created by **grok** during execution:

`array_sizes.default`
> See Section 5.1.3.1.

`grok.dbg`
> General purpose output file for debugging information.

These scratch files are usually created when **grok** scans input files and removes comments and blank lines:

`scratch4`
> Cleaned up copy of *prefix*`.grok` file.

`scratch_dprops`
> Cleaned up copy of `.dprops` file.

`scratch_fprops`
> Cleaned up copy of `.fprops` file.

`scratch_mprops`
> Cleaned up copy of `.mprops` file.

`scratch_oprops`
> Cleaned up copy of `.oprops` file.

`scratch_arc`
    Temporary file created if Zones from arcview is used.

`fractran2f3d.lst`
    Listing file created if Read fractran 2d grid is used.

These files are created by **grok** as data for a specific problem is processed and most are binary files which are read later by **HydroGeoSphere**:

prefix`o.eco`
    grok listing file.

prefix`o.gen`
    General input data (binary).

prefix`o.coordinates`
    3-D subsurface mesh node coordinates (binary).

prefix`o.coordinates_overland`
    2-D surface flow mesh node coordinates if dual approach used (binary).

prefix`o.elements`
    3-D subsurface mesh porous media element node lists (binary).

prefix`o.elements_dual`
    3-D subsurface mesh dual continuua element node lists (binary).

prefix`o.elements_overland`
    2-D surface flow mesh element node lists if dual approach used (binary).

prefix`o.elements_fracture`
    2-D discrete fracture element node lists (binary).

prefix`o.fac`
    3-D subsurface mesh face node lists (binary).

prefix`o.seg`
    3-D subsurface mesh segment node lists (binary).

prefix`o.species`
    Species (i.e. solute) data (binary).

prefix`o.hi`

Initial nodal heads (binary).

*prefix*o.ci
  Initial nodal concentrations (binary).

*prefix*o.siz
  Array size data determined by **grok** and used by **HydroGeoSphere** for run-time array allocation.

These files are created by **grok** if the 2-D Random Fracture Generator, as described in Section 5.3.4 is used:

*prefix*o.rfrac.fractures
  Listing file with fracture zone information.

*prefix*o.rfrac.apertures
  Listing file with fracture aperture information.

*prefix*o.rfrac.lengths
  Listing file with fracture length information.

*prefix*o.rfrac.orientations
  Listing file with fracture orientation information.

This file of general use is created by **HydroGeoSphere** during execution:

hs.dbg
  General purpose output file for debugging information.

These files are created by **HydroGeoSphere** for use with the Run-time Debug Utility described in Appendix C:

debug.control
  You can edit this file to change the run-time debug behaviour.

scratch_debug
  Cleaned up copy of debug.control file.

progress.dat

TECPLOT formatted file of CPU time vs system time.

These files are created by **HydroGeoSphere** for a specific problem as the run progresses:

*prefix*o.lst
HydroGeoSphere listing file.

*prefix*o.sat
Elemental saturation (binary).

*prefix*o.hen
Final heads for restart (binary).

*prefix*o.cen
Final concentrations for restart (binary).

*prefix*o.bal
Mass balance information (binary).

*prefix*o.vel
Porous medium elemental velocity (binary).

*prefix*o.vel_gb
Elemental vx, vy only for Grid Builder plot and unit-thickness problems.

*prefix*o.water_balance.dat
Fluid mass balance summary for each timestep.

*prefix*o.mass_balance_cumulative.*species*.dat
Cumulative solute mass balance for each timestep.

*prefix*o.mass_balance_summary.*species*.dat
Solute mass balance summary for each timestep.

*prefix*o.mass_crossing_slice
Mass crossing a slice if slice flux output used.

*prefix*o.mass_balance_raw*species*.dat
Mass balance information for each timestep for a species.

*prefix*o.observation_well_flow.*well*.dat
Observation well/point flow solution output.

*prefix*o.flu

Fluid flux if Flux output nodes used.

*prefix*o.`flu_b`
   Fluid flux if Binary flux output nodes used (binary).

*prefix*o.`river_flux`
   Fluid flux if Make river nodes used.

*prefix*o.`drain_flux`
   Fluid flux if Make drain nodes used.

*prefix*o.`hydrograph.`*hydrograph*.`dat`
   Total flux for each timestep for a set of hydrograph nodes.

*prefix*o.`wvo`
   Elemental velocity in wells.

*prefix*o.`well_averaged_transport.`*well.species*.`dat`
   Flux-averaged concentration in wells.

*prefix*o.`observation_well_conc.`*well.species*.`dat`
   Concentration at observation wells.

*prefix*o.`flm`
   Mass flux if Flux output nodes used.

*prefix*o.`flm_b`
   Mass flux if Binary flux output nodes used (binary).

*prefix*o.`mf_interp`
   Interpolated mass flux if used.

These files may be created by **HydroGeoSphere** for a specific problem for each output time, in this case number 001, as the run progresses:

*prefix*o.`head.001`
   Porous medium head.

*prefix*o.`saturation.001`
   Porous medium saturation.

*prefix*o.`concentration.`*species*.`001`
   Porous medium concentration.

*prefix*o.`immobile_concentration.`*species*.`001`

Porous medium immobile zone concentration.

*prefix*o.`velocity_linear.001`
  Porous medium average linear groundwater velocity.

*prefix*o.`velocity_darcy.001`
  Porous medium Darcy velocity.

*prefix*o.`head_dual.001`
  Dual continuum head.

*prefix*o.`saturation_dual.001`
  Dual continuum saturation.

*prefix*o.`concentration_dual.`*species*`.001`
  Dual continuum concentration.

*prefix*o.`velocity_linear_dual.001`
  Dual continuum average linear groundwater velocity.

*prefix*o.`velocity_darcy_dual.001`
  Dual continuum Darcy velocity.

*prefix*o.`head_overland.001`
  Surface flow head.

*prefix*o.`velocity_overland.001`
  Surface flow velocity.

*prefix*o.`exchange_flux.001`
  Surface/subsurface exchange flux.

*prefix*o.`et_pm_evap.001`
  Subsurface evaporation flux.

*prefix*o.`et_pm_trans.001`
  Subsurface transpiration flux.

*prefix*o.`et_surf_evap.001`
  Surface evaporation flux.

*prefix*o.`et_tot.001`
  Total evapotranspiration flux.

*prefix*o.`fracture_aperture.001`

Discrete fracture aperture.

prefix`o.velocity_linear_fracture.001`
Discrete fracture average linear groundwater velocity.

prefix`o.velocity_darcy_fracture.001`
Discrete fracture Darcy velocity.

Appendix C

Run-time Debug Utility

When using **HydroGeoSphere** to solve complex problems, especially non-linear ones, it is often the case that the end user would like to, for example, view intermediate results, modify input parameters or print out the contents of various matrices. Normally, the program developer would carry out such tasks with the aid of a debugger, which is normally supplied as part of a program development package. Unfortunately, most end users do not have access to such a package, and even if they did, running a code which has been compiled in debug mode is generally much slower than when it has been optimized for release to the public.

In order to address some of these problems, we have developed a run-time debug utility which operates on the optimized **HydroGeoSphere** executable and is able to be activated and deactivated interactively without halting the execution of the program.

At various points during program execution, **HydroGeoSphere** checks for the presence of a file called `debug.control` in the same directory as the *prefix*`.grok` file and, if found, performs certain actions as indicated in the file. If it is not found, a file will be created which contains instructions for performing run-time debug actions.

If you want to prevent **HydroGeoSphere** from performing run-time debugging (including checking the debug control file), you can do so using the instruction No runtime debug.

The head and tail of the `debug.control` file are shown in Table C.1. Lines beginning with the comment character (!) are ignored. When first generated, the only uncommented line is the first one, `debug off`, which causes **HydroGeoSphere** to run normally, and not to take any run-time debug actions.

The contents of the file depend on the nature of the problem which is being simulated. For example, if it has no transport component there will not be any transport-related information written to the file.

To activate the debug utility, just comment out the `debug off` instruction, and uncomment one or more of the remaining instructions as desired.

```
debug off
! --------------------------------------------- Pause execution
! pause timestep
! pause at time
!        10000.00000
! pause flow convergence loop
! --------------------------------------------- Produce output
! write output files
! write saturated flow matrices

...etc...

! --------------------------------------------- Adaptive timestep targets
! concentration change target
!           0.05000
! mass change target
!        0.10000E+21
! mass error target
!          100.00000
! minimum timestep multiplier
!            0.50000
! maximum timestep multiplier
!            2.00000
```

Table C.1: Sample Contents of File `debug.control`

```
!debug off
! ------------------------------------------------ Pause execution
 pause timestep
...etc...
```

Table C.2: Modified File `debug.control`

For example, if you modified and saved `debug.control` so it appeared as shown in Table C.2 then **HydroGeoSphere** would pause at the end of the next timestep.

Note that if the `debug control` file becomes corrupted, or if you modify the problem in some way (e.g. activate transport) you can automatically generate a fresh copy by deleting or renaming it before or during **HydroGeoSphere** execution.

The effect caused by uncommenting the various instructions in the `debug.control` file will now be described, and input data requirements will be discussed as required.

Debug off
Ignore the rest of the contents of the `debug control` file, whether or not they are uncommented. To activate the run-time debug feature, this line should be commented out.

• • •

Pause timestep
Pause at the beginning of the next timestep and issue the following message on the screen:
`DEBUG CONTROL: pause timestep, press a key to continue`
This is usually used to prevent other instructions, such as write output files, from producing too much output.

• • •

Pause at time

1. **time** Simulation time.

Proceed until the given simulation time is reached and then pause.

• • •

Pause newton loop
Pause at the beginning of the next Newton iteration.

• • •

Pause flow convergence loop

Pause at the beginning of the next flow solver iteration.

• • •

Write output files

Treat each timestep as if it was defined in the *prefix*.grok file as an ouput time (i.e. using the instruction Output times). Head, concentration, saturation etc. output files will be written at each timestep.

This instruction is usually used in conjunction with Pause timestep.

• • •

Watch node list...end

1. **first node number**

2. **second node number**

3. ...etc

A list of node numbers for which you want detailed output.

• • •

For each watch node, the following instructions write detailed output to the `hs.dbg` file:

    Write flow matrices
    Write transport matrices
    Write overland flow
    Write fracture (dual node) flow
    Write evapotranspiration

Detailed output usually consists of the coefficient matrix and right-hand side vector for the watch node.

In the case of the Write evapotranspiration instruction, the quantities of fluid that are removed by the various process (e.g. canopy evaporation, evaporation, transpiration etc.) and the remaining potential evapotranspiration are printed out at each timestep.

When using a feature such as `Write flow matrices` for example, be aware that **Hydro-GeoSphere** may write a lot of information to disk, and so they should be activated and deactivated for only a few timesteps at a time.

Write delval node info

Detailed information, including xyz-coordinate, head, saturation and relative permeability, for the node with the largest delval at each timestep is written to the .lst file.

• • •

Write resval node info

Detailed information, including xyz-coordinate, head, saturation and relative permeability, for the node with the largest resval at each timestep is written to the `.lst` file.

• • •

Write seepage face output to .lst file

Details of the seepage node calculations, including which nodes are currently acting as seepage nodes and the fluid flux exiting the domain at each active seepage node are written to the `.lst` file.

• • •

Time progress

Causes FRAC3DVS to write a Tecplot file which contains values of simulation time versus real time. The slope of this line is a measure of how efficient an **HydroGeoSphere** is, and can be used to gauge how effective changes in parameter values are in speeding up the simulation.

The first time this instruction is executed, the file `progress.dat` is created, and data is written to it as the run progresses. The results for a given simulation are tagged with a date and timestamp.

Results from subsequent are appended to the file so they can be compared from run to run.

• • •

Write krw file

1. **fname** The name of the file to write the hydraulic conductivity values to.

For each principal direction and for each element, outputs the product of the relative permeability and the saturated hydraulic conductivity to the file **fname**. This file can then be read in a subsequent simulation using the instruction Read elemental k from file.

• • •

Time format

1. **type** An integer value indicating the type of format to use when writing time values to output files. Acceptable values are:

1	Fixed.
2	Scientific.
3	general.

<div align="center">• • •</div>

Mass balance format
As above except for mass balance output

<div align="center">• • •</div>

Nodal flow check

1. **unproject_file** If .true, this logical switch turns on the nodal flow check feature or if .false., turns it off.

<div align="center">• • •</div>

Force timestep

1. **delta_t** Subsequent timesteps will be set to this value.

<div align="center">• • •</div>

The following commands are identical to those used in the `.grok` input file and will just be listed here. They are used to modify parameter values as **HydroGeoSphere** is running. Note that if you change a parameter here, **HydroGeoSphere** will continue to use the new value even if you disable run-time debugging (i.e. uncomment Debug off) or comment out the instruction that was used to change the value.

> Flow solver convergence criteria
> Flow solver detail
> Flow maximum iterations
> Transport solver convergence criteria
> Transport solver detail
> Transport solver maximum iterations
> Newton maximum iterations
> Newton absolute convergence criteria
> Newton residual convergence criteria
> Compute underrelaxation factor
> Nodal flow check tolerance

Newton maximum update for head
Newton maximum update for depth
Newton maximum residual increase
Maximum timestep
Minimum timestep multiplier
Maximum timestep multiplier

The following commands can be used to set the values of the various targets used in the adaptive timestepping procedure.

Head change target
Saturation change target
Water depth change target
Newton target
Concentration change target
Mass change target
Mass error target

Appendix D

HSBATCH: Windows Batch Run Utility

A common requirement of a computer program is that it be able to run several different problems in sequence, hopefully without user intervention. For example, a Monte Carlo approach could require the user to do several hundred runs with slightly varying data sets, or a developer might want to execute a suite of verification runs with each new release of a code.

With **HydroGeoSphere** you have the capability to carry out multiple batch runs by using the program **HSBATCH**. To meet our own requirements, we have set up the verification examples for **HydroGeoSphere** to be run in batch mode, and the end user can easily adapt the approach to meet their own requirements.

The first step in setting up batch runs is to create directories which will contain all of the data files which are necessary to run the problems. In the case of the **HydroGeoSphere** verification problems, the directory C:**Program Files**\HydroGeoSphere\verification contains one or more subdirectories for each verification problem. A partial list of its contents are shown here:

```
Directory of C:\program_files\HydroGeoSphere\verification

16/05/2006  11:47 AM    <DIR>          .
16/05/2006  11:47 AM    <DIR>          ..
16/05/2006  11:33 AM    <DIR>          abdul
16/05/2006  11:01 AM    <DIR>          digiammarco
16/05/2006  11:08 AM    <DIR>          dual
...etc...
16/05/2006  11:05 AM    <DIR>          ward
16/05/2006  11:05 AM    <DIR>          yang
16/05/2006  10:50 AM             1,025 dev_path.hsbatch
16/05/2006  10:50 AM               403 install.hsbatch
```

First, a file with the extension `.hsbatch` is set up to run the desired suite of batch problems. In this case we called it `install.hsbatch`, and here is a partial listing of its contents:

```
! HydroGeoSphere batch processor input file

! verification problems
batch directory list
digiammarco
dual
f_cd

...etc...

smith_woolhiser
panday
saltpool1
!elder
end
```

The file can contain comments, which begin with an exclamation point (!), blank lines, and include instructions.

The instruction Batch directory list is followed by a list of subdirectories, one per line, that contain the problem data to be run in batch mode, followed by an End card. Each subdirectory should contain one set of problem data, and in order for the executables to run without prompting the user for the problem prefix, it is necessary to create a file called `batch.pfx`, which consists of a single line containing the problem prefix, in each problem directory.

This example shows the simplest configuration that an `.hsbatch` file can have, in which it uses default locations for both the program executables and the problem data. Executables are found by searching the Windows default search path, which should contain C:\Program Files\HydroGeoSphere following a typical program installation. Problem data are to be found in subdirectories of the directory where the `.hsbatch` file is located, in this case C:\Program Files\HydroGeoSphere\verification.

To run **HSBATCH** for this example, you should first open a Command Prompt, change directories to C:\Program Files\HydroGeoSphere\verification, type:

```
hsbatch
```

and you will see the program banner. Next supply the `.hsbatch` file prefix, in this case:

```
install
```

As the program runs, output from **grok**, **HydroGeoSphere** and **HSPLOT** are echoed to the screen. A log of the **HSBATCH** process, including program run times, is written to the file *prefix*o.eco_hsbatch

A more complex situation could be one in which a developer wants to use a set of executables which are not in the default search path, or problem data from more than one directory. The file dev_path.hsbatch illustrates how to handle these types of situations:

```
! HydroGeoSphere batch processor input file
!
!  If you use this file without hardwiring pathes to the executables,
!   it will use the ones it finds in the default search path.

!  NOTE: always use an absolute path e.g. c:\program_files\hydrogeosphere\
!   and not relative e.g. ..\grok\release\

grok path
c:\program_files\hydrogeosphere\

hydrogeosphere path
c:\program_files\hydrogeosphere\

hsplot path
c:\program_files\hydrogeosphere\


!  If you use this file without hardwiring the path to the data parent
!   directory, it will use the .hsbatch file directory as the parent.

! First run a verification problem
change directory
c:\program_files\hydrogeosphere\verification

batch directory list
pm_cd
end

! Now run an illustrative problem
change directory
c:\program_files\hydrogeosphere\illustration

batch directory list
reactive_iron_barrier_in_fractures
```

```
   end
```

In this case, the instructions grok path, hydrogeosphere path and hsplot path point to the locations where the program executables can be found.

Change directory sets the parent directory to apply to all subdirectories listed in the subsequent Batch directory list. As you can see, the file can contain multiple sets of Change directory and Batch directory list instructions. In this case, the location of the `.hsbatch` file is not critical, since absolute pathes to the problem data are supplied.

Appendix E

HSPLOT: Visualization Post-processor

The utility program **HSPLOT** can be used to convert raw **HydroGeoSphere** output into files that are compatible with third-party visualization packages.

Like **grok** and **HydroGeoSphere**, when **HSPLOT** starts executing, it requires a problem prefix which can be entered interactively from the keyboard or supplied in the `batch.pfx` file.

The first time your run **HSPLOT** for a specific problem, it creates the file *prefix*`.plot.control` (referred to as the plot control file below), which will contain a list of instructions that affect the output file format and contents. Note that if the plot control file becomes corrupted you can automatically generate a fresh copy by deleting or renaming it before executing **HSPLOT**.

We will now discuss the various sections of the plot control file and how they can be modified to produce the desired outputs.

E.1   Output format modes

Currently, **HSPLOT** support two output formats: Tecplot and GMS. The first two lines of the plot control file are:

```
 tecplot mode
! gms mode
```

Lines beginning with the comment character (!) are ignored, and so by default, the file is set up to generate TECPLOT formatted output (i.e. `Tecplot mode` instruction is uncommented).

Uncommenting an instruction has the following effect:

Tecplot mode
Causes **HSPLOT** to generate Tecplot formatted output.

• • •

Gms mode
Causes **HSPLOT** to generate GMS formatted output.

NOTE: At this time, GMS mode produces output for the porous medium and dual continuum domains, but not for the discrete fractures or surface flow domains. The following instructions are also not available in GMS mode:

> Isolate node
> Truncate 3d domain
> Truncate time domain
> Compare heads 3d domain

• • •

E.2   Porous media output

The next section of the plot control file controls the output of porous media (i.e. 3-D domain) data:

```
 write 3d domain file
! no 3d heads
! no 3d linear velocities (vx, vy, vz)
! no 3d darcy velocities (vx, vy, vz)
! no elemental hydraulic conductivity
! no elemental porosity
! isolate node
!  1      ! node number
!  0      ! extension factor
! truncate 3d domain
! -0.10000000E+21 0.10000000E+21
! -0.10000000E+21 0.10000000E+21
! -0.10000000E+21 0.10000000E+21
! truncate time domain
! -0.10000000E+21 0.10000000E+21
! ---------------------------
```

This section begins with a general instruction that will control wether or not porous media output is written, followed by a group of instructions that can be used to tailor the porous media output.

Write 3d domain file
Causes **HSPLOT** to write the output files for the porous media domain.

The default Tecplot formatted output file would contain all available data (i.e. heads, saturations, concentrations, velocities etc.) at each output time and for the entire 3-D domain and would be named:

   *prefix*o.dat   All data

GMS formatted output files would be named:

   *prefix*o.gms.gpr   Project file
   *prefix*o.gms.3dm   Mesh file
   *prefix*o.gms_scalars.dat   Scalars (e.g. heads)
   *prefix*o.gms_vectors.dat   Vectors (e.g. velocities)

• • •

The following instructions prevent **HSPLOT** from writing the named variables to the output file.:

      No 3d heads
      No 3d saturations
      No 3d compaction
      No 3d linear velocities (vx, vy, vz)
      No 3d darcy velocities (qx, qy, qz)
      No 3d concentrations
      No 3d immobile zone concentrations
      No 3d fracture apertures
      No permafrost
      No elemental hydraulic conductivity
      No elemental hydraulic conductivity dual
      No elemental porosity
      No elemental specific storage
      No elemental tortuosity
      No elemental peclet numbers

NOTE: The presence of the instructions listed above can be dependent on the nature of the simulation. For example, saturations would not have been written unless the system is

variably-saturated, and the no 3d saturations instruction would not then appear in the plot control file.

Truncate 3d domain

1. **x1, x2** x-range of the domain.

2. **y1, y2** y-range of the domain.

3. **z1, z2** z-range of the domain.

Causes **HSPLOT** to restrict the output domain size to within the given xyz-ranges. This is used to zero in on an interesting subregion of the 3-D domain or to reduce the size of the output file and speed up file I/O.

● ● ●

For example, the following instructions:

```
truncate 3d domain
100.0    200.0
0.0   1000.0
-0.10000000E+21 0.10000000E+21
```

would reduce the domain size to the x-range from 100 to 200, the y-range from 0 to 1000 but would leave the z-range at its full extent.

Isolate node

1. **NodeNumber** Number of node to be isolated.

2. **ExtensionFactor** An integer with a value of zero or greater.

Causes **HSPLOT** to restrict the output domain so that only elements that contain **Node-Number** are plotted (**ExtensionFactor** = 0). Increasing **ExtensionFactor** causes neighbour elements to be plotted also. For example, **ExtensionFactor**=1 would be used to include elements that share a node with an element that contains **NodeNumber**.

● ● ●

Truncate time domain

1. **t1, t2** Time range of the domain.

Causes **HSPLOT** to restrict the time range of the output so that only output times that are between times **t1** and **t2** (inclusive) are included.

NOTE: The first output time is always included.

● ● ●

Compare heads 3d domain

1. **x1, x2** x-range of the domain.

2. **y1, y2** y-range of the domain.

3. **z1, z2** z-range of the domain.

Causes **HSPLOT** to write a Tecplot-formatted output file called *prefix*o.`compare_heads.dat` that contains simulated versus observed head data.

Only head data that falls within the restricted region defined by the xyz-ranges is included.

NOTE: This instruction will only be present in the plot control file if an observed head data file called *prefix*.`observed_heads` is present in the directory with the *prefix*.`grok` file.

● ● ●

The file *prefix*.`observed_heads` should be of the following form:

```
1255
8.29842E+05 2.41079E+06 2.85000E+02          2.85000E+02
8.34476E+05 2.39450E+06 2.86900E+02          2.95100E+02
...etc.
```

where the first line is the number of observed head points in the file, followed by one line for each point that contains the xyz coordinates and observed head value at each point.

E.3 Dual continuum output

The following instructions affect how **HSPLOT** handles the output for the dual continuum domain:

Write dual-permeability results to 3d domain file
Causes **HSPLOT** to write the dual-permeability results to the 3D domain output file.

Although it is not necessary to define all porous medium elements as dual continuum elements, **HydroGeoSphere** treats the dual continuum output as being the same size as the porous medium, and uses missing values (heads, saturations and concentrations $= -9999$, zone number $= 0$) to designate regions where the dual continuum is absent. These values can be used to value blank dual continuum output in Tecplot.

• • •

The following instructions prevent **HSPLOT** from writing the named variables to the output file:

> No 3d dual-permeability heads
> No 3d dual-permeability saturations
> No 3d dual-permeability linear velocities (vx dual, vy dual, vz dual)
> No 3d dual-permeability darcy velocities (qx dual, qy dual, qz dual
> No 3d dual-permeability concentrations

NOTE: The presence of the instructions listed above can be dependent on the nature of the simulation.

E.4   Discrete fracture output

The following instructions affect how **HSPLOT** handles the output for the discrete fracture domain:

Write fracture domain file
Causes **HSPLOT** to write the output files for the discrete fracture domain.

The default Tecplot formatted output file would contain all available data (i.e. heads, saturations, concentrations, velocities etc.) at each output time and for the entire fracture domain and would be named:

> *prefix*o.`fracture.dat`   All data

When using the common node approach, the full set of nodal output is written to the output file along with the 2-D fracture element data. In the dual-node approach, only the fracture node data and elements are written to the file.

GMS formatted output for discrete fractures is not currently supported.

• • •

These instructions have similar effects as those described above for porous media output:

> Truncate fracture domain
> Truncate fracture time domain

The following instructions prevent **HSPLOT** from writing the named variables to the output file:

No fracture heads
No fracture linear velocities (vx, vy, vz)
No fracture darcy velocities (vx, vy, vz)
No fracture concentrations
No fracture/subsurface solute exchange flux
No fracture immobile zone concentrations
No fracture apertures

NOTE: The presence of the instructions listed above can be dependent on the nature of the simulation.

The following instruction is written in the plot control file at the end of the porous medium section if the common-node approach is being used:

Write fracture elements to 3d domain file
Causes **HSPLOT** to append the fracture element data to the 3D domain output file *prefix*o.dat.

This causes fractures to appear as shown in Figure E.1.

● ● ●

E.5 Surface flow output

The following instructions affect how **HSPLOT** handles the output for the surface flow domain:

Write surface flow domain file
Causes **HSPLOT** to write the output files for the surface flow domain.

The default Tecplot formatted output file would contain all available data (i.e. heads, depth, concentrations, velocities etc.) at each output time and for the entire surface flow domain and would be named:

   *prefix*o.overland.dat   All data

GMS formatted output for surface flow is not currently supported.

● ● ●

These instructions have similar effects as those described above for porous media output:

Truncate surface flow domain
Truncate surface time domain
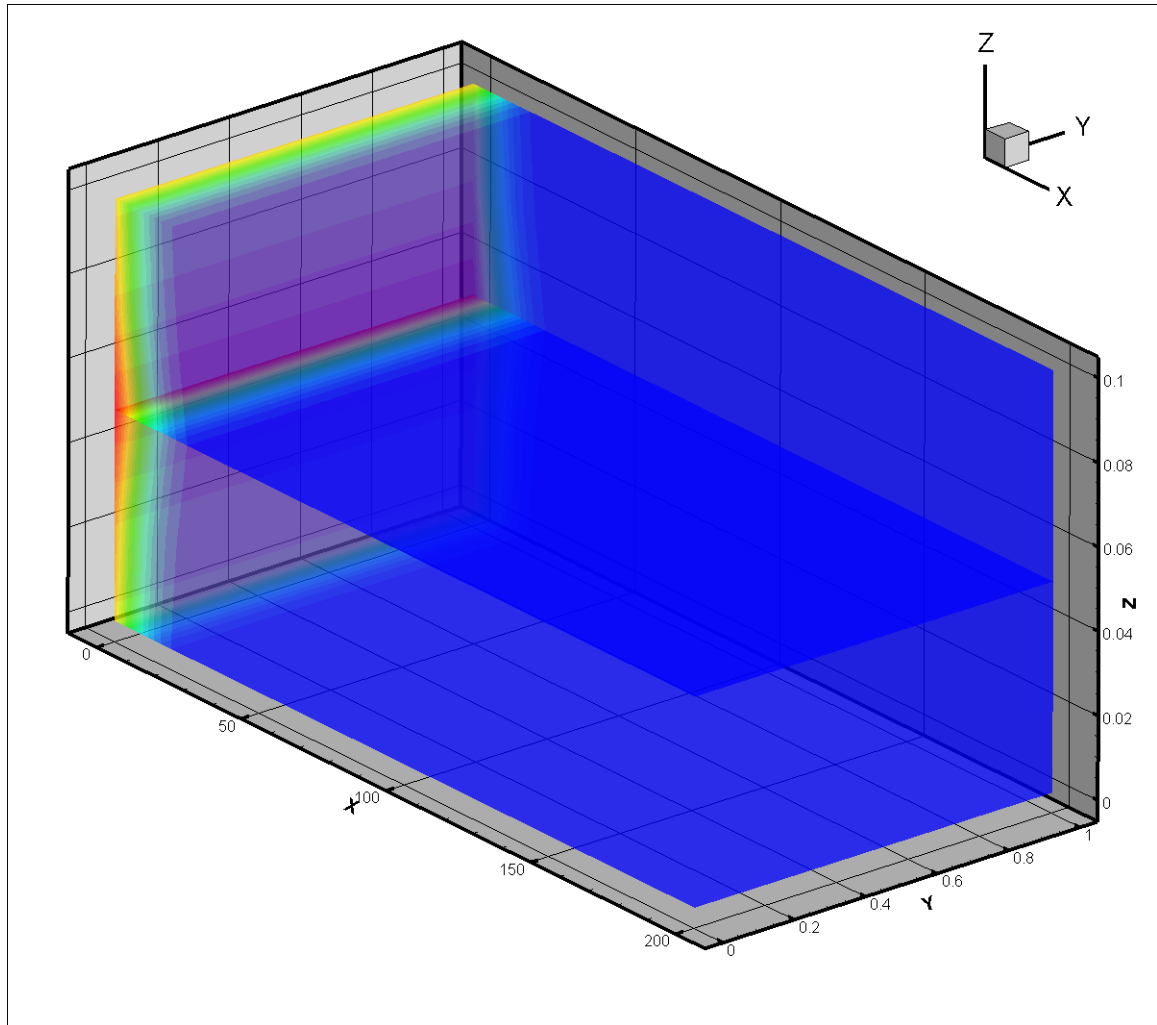
Figure E.1: Example output from the write fracture elements to 3d domain file instruction

The following instructions prevent **HSPLOT** from writing the named variables to the output file:

> No surface flow heads
> No surface log10(depth)
> No surface flow linear velocities (vx, vy, vz)
> No et
> No surface flow concentrations
> No surface/subsurface exchange flux
> No surface/subsurface solute exchange

NOTE: The presence of the instructions listed above can be dependent on the nature of the simulation.

Appendix F

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.

F.1   2-D meshes (i.e. slices)

The instructions Read slice and 2D mesh to gms read and write 2-D mesh data in GMS format respectively. The portion of the 2-D 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 2-D 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.

F.2   Ascii scalar data set files

The instruction Generate layers from slice can read a nodal data set in order to define a variable surface (usually an elevation for the z-coordinate) for the base of the 3-D grid or the top of a layer. This file should be written in GMS ascii format.

The ascii file format recognized by **grok** is as follows:

```
    DATASET                           ! File type identifier
    ...etc...
    ND n                              ! Number of data values
    ...etc...
    TS time                           ! Time step of the following data
```

```
    val_1                                ! Scalar data values
    val_2

    ...etc...

    val_n
```

In this case, the value n should correspond to the number of nodes. Currently, the first line of the file must contain the string `DATASET`, followed at some point by a `ND` card and then a `TS` card. Other other cards may be present in the file (e.g. `STAT`) but they will be ignored. 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 type formats are as follows:

Card Type	DATASET
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 2-D slice being used to generate the 3-D mesh.		
Required	YES		
Format	ND n		
Sample	ND 4		
Field	*Variable*	*Value*	*Description*
1	n	+	The number of nodes per 2-D slice.

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	$\pm$	The time step value. Not used by **grok**.
2–(n+1)	val	$\pm$	The scalar values for each item.

Appendix G

Grid Builder file formats

G.1 2-D meshes (i.e. slices)

The instruction Read gb 2D grid reads 2-D triangular-element mesh data in Grid builder format.

The file which contains the node coordinates for the 2-D 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 2-D triangular grid and x2d and y2dare the x-, y-coordinates of the nodes.

The file which contains the element incidences for the 2-D 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 2-D 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 2-D slices made up of rectangular elements in a future release.

The file which contains the element incidences for the 2-D 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 2-D 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.

G.2   Scalar data set files

The file which contains the scalar values for the nodes in a 2-D triangular mesh has a file name of the form **gb_prefix.nprop.***descriptor* assigned by GRID BUILDER. The following fortran code segment shows how this data is read:

```
integer*4 nn2d
real*4 nprop(maxnn2d)
character*80 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.

Appendix H

Raster file formats

Raster data consists of a set of values that are laid out on a rectangular grid with uniform spacing in x and y. The raster coordinates need not coincide with **grok** mesh nodes and can even fall outside the model domain.

The raster must consist of header information containing a set of keywords, followed by cell values in row-major order. The file format is based on the ArcGIS ascii file conventions and must consist of the following:

```
NCOLS xxx
NROWS xxx
XLLCORNER xxx
YLLCORNER xxx
CELLSIZE xxx
NODATA_VALUE xxx
row 1
row 2
.
.
.
row n
```

where `NCOLS` and `NROWS` are the number of rows and columns in the raster respectively, `XLLCORNER` and `YLLCORNER` are the x and y coordinates of the lower left corner of the raster, `CELLSIZE` is the size of each raster cell, `NODATA_VALUE` is the value in the ASCII file representing cells whose true value is unknown (i.e. missing) and `xxx` is a number. Row 1 of the data is at the top of the grid, row 2 is just under row 1, and so on.

For example:

```
ncols 480
```

```
nrows 450
xllcorner 378923
yllcorner 4072345
cellsize 30
nodata_value -32768
43 2 45 7 3 56 2 5 23 65 34 6 32 54 57 34 2 2 54 6
35 45 65 34 2 6 78 4 2 6 89 3 2 7 45 23 5 8 4 1 62  ...
```

Cell values should be delimited by spaces. No carriage returns are necessary at the end of each row in the grid. The number of columns in the header is used to determine when a new row begins. The number of cell values must be equal to the number of rows times the number of columns.

Index