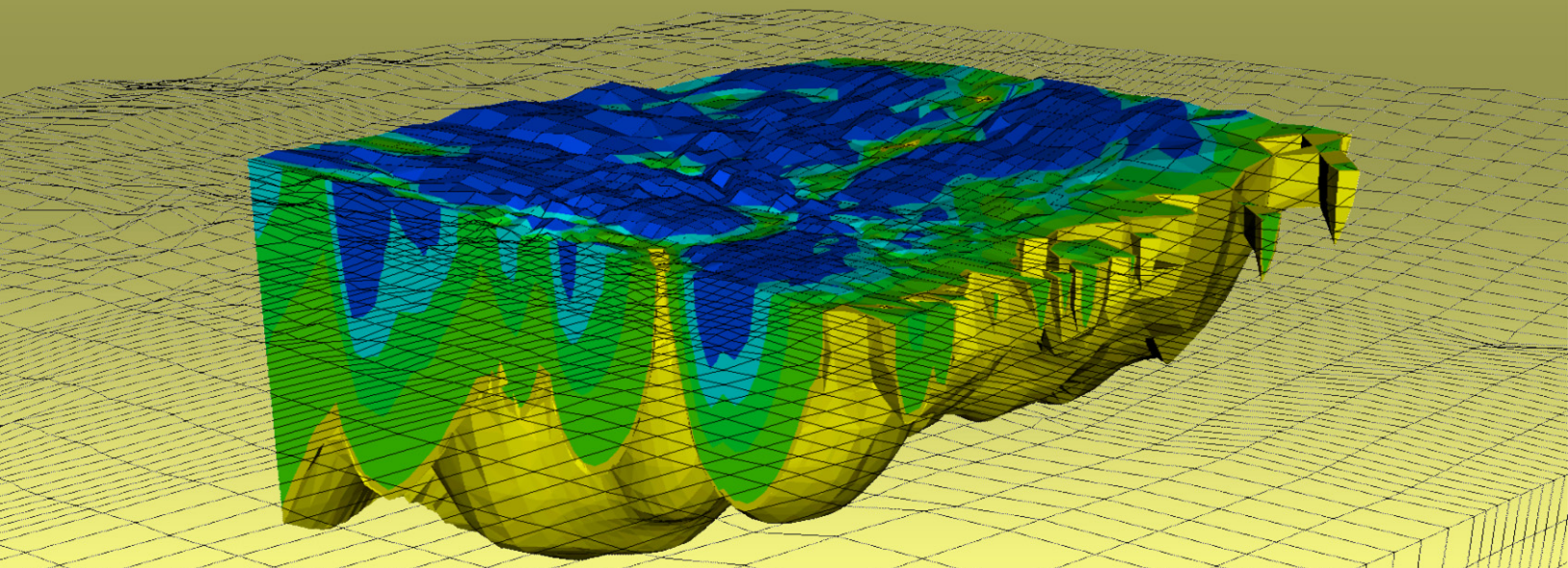
U.S. Department of the Interior
U.S. Geological Survey

SUTRA

A Model for Saturated-Unsaturated Variable-Density Ground-Water Flow with Solute or Energy Transport



Water-Resources Investigations Report 02-4231

Version of June 2, 2008 (SUTRA Version 2.1)
Latest version available at http://water.usgs.gov/nrp/gwsoftware

USGS
science for a changing world

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

SUTRA

A Model for Saturated-Unsaturated, Variable-Density Ground-Water Flow with Solute or Energy Transport

by Clifford I. Voss and Alden M. Provost

Water-Resources Investigations Report 02-4231

This report supersedes
U.S. Geological Survey Water-Resources Investigations Report 84-4369

Version of June 2, 2008 (SUTRA Version 2.1)
Latest version available at http://water.usgs.gov/nrp/gwsoftware

Reston, Virginia
2008

U.S. Department of the Interior
DIRK KEMPTHORNE, Secretary

U.S. Geological Survey
Mark D. Myers, Director

For additional information write to:

SUTRA Support
U.S. Geological Survey
431 National Center
Reston, Virginia 20192
USA

Copies of this report can be purchased from:

U.S. Geological Survey
Branch of Information Services
Box 25286
Denver, Colorado 80225
USA

This document and the SUTRA computer code may be downloaded without cost from a U.S. Geological Survey Internet site: *http://water.usgs.gov/nrp/gwsoftware*

PREFACE

This report describes a complex computer model, SUTRA, for analysis of fluid flow and solute or energy transport in subsurface systems. The original version of SUTRA was released in 1984 (Voss, 1984). The version described in this report (SUTRA 2.1) is an upgrade that adds to the previous version (SUTRA 2.0, originally designated as 2D3D.1) (Voss and Provost, 2002) the ability to use irregular 3D meshes, conveniently incorporate input data from separate files, define schedules to control time stepping and observation output, interpolate observations in time and space, and output observations in an alternative format. The user is cautioned that although the model will accurately reproduce the physics of flow and transport when used with proper discretization, it will give meaningful results only for well-posed problems based on sufficient supporting data.

The user is kindly requested to notify the originating office of any errors found in this report or in the computer program. Please report these by mail to:

SUTRA Support
U.S. Geological Survey
431 National Center
Reston, VA 20192
USA

Updates will occasionally be made to the report and the computer program to include corrections of errors, addition of processes that may be simulated, and changes in numerical algorithms. The version date of this report is given on the title page.

Copies of the computer program and manual for SUTRA and associated utility codes for preprocessing and postprocessing are available free of charge from a U.S. Geological Survey Web site:

http://water.usgs.gov/nrp/gwsoftware/sutra.html

S U T R A

CONTENTS

Page

Abstract 1

INTRODUCTION

Chapter 1: Introduction 5
1.1 Purpose and Scope 5
1.2 The Model 6
1.3 SUTRA Processes 6
1.4 Some SUTRA Applications 7
1.5 SUTRA Numerical Methods 7
1.6 SUTRA as an Analytical Tool 8

SUTRA FUNDAMENTALS

Chapter 2: Physical-Mathematical Basis of SUTRA Simulation 13
2.1 Physical Properties of Solid Matrix and Fluid 14
2.2 Saturated-Unsaturated Ground-Water Flow 19
2.3 Energy Transport in Ground Water 26
2.4 Solute Transport in Ground Water 29
2.5 Dispersion 34
2.6 Unified Description of Energy and Solute Transport 46

Chapter 3: Fundamentals of Numerical Algorithms 51
3.1 Spatial Discretization by Finite Elements 52
3.2 Representation of Coefficients in Space 56
3.3 Integration of Governing Equation in Space 59
3.4 Time Discretization of Governing Equation 65
3.5 Boundary Conditions and Solution of Discretized Equation 67

DETAILS OF SUTRA METHODOLOGY

Chapter 4: Numerical Methods 73
4.1 Basis and Weighting Functions 73
4.2 Coordinate Transformations 81
4.3 Gaussian Integration 83
4.4 Numerical Approximation of SUTRA Fluid Mass Balance 85
4.5 Numerical Approximation of SUTRA Unified Solute Mass and Energy Balance 92

4.6 Consistent Evaluation of Fluid Velocity 99
4.7 Temporal Evaluation of Adsorbate Mass Balance 102

Chapter 5: Other Methods and Algorithms 105
5.1 Rotation of Permeability Tensor 105
5.2 Radial Coordinates 106
5.3 User-defined Schedules 108
5.4 Control of Time Stepping 108
5.5 Solution Sequencing 108
5.6 Observation Output 110
5.7 Velocity Calculation for Output 110
5.8 Budget Calculations 111
5.9 Program Structure and Program Unit Descriptions 116
5.10 Iterative Solver Package 147

SUTRA SIMULATION EXAMPLES

Chapter 6: Simulation Examples 151
6.1 Pressure Solution for Radial Flow to a Well (Theis Analytical Solution) 151
6.2 Radial Flow with Solute Transport (Analytical Solutions) 153
6.3 Radial Flow with Energy Transport (Analytical Solution) 156
6.4 Areal Constant-Density Solute Transport (Example at Rocky Mountain Arsenal) .. 159
6.5 Density-Dependent Flow and Solute Transport (Henry (1964) Solution for Seawater Intrusion) 163
6.6 Density-Dependent Radial Flow and Energy Transport (Aquifer Thermal Energy Storage Example) 167
6.7 Constant-Density Unsaturated Flow and Solute Transport (Example from Warrick, Biggar and Nielsen (1971)) 171
6.8 Variable-Density Saturated-Unsaturated Flow and Solute Transport (Comparison of 2D-Radial and Fully 3D SUTRA Solutions) 174

SUTRA SIMULATION SETUP

Chapter 7: Simulation Setup 183
7.1 SUTRA Data Requirements 183
7.2 Discretization Rules of Thumb 194
7.3 Program Dimensions 198
7.4 Input and Output Files 200
7.5 User-Supplied Programming 202
7.6 Modes and Options 204

References 207

APPENDICES

Appendix A: List of Symbols 211
Appendix B: SUTRA Input Data List 227

LIST OF FIGURES

Page

Figure 2.1. Schematic plot of saturation-capillary pressure relation. 17
Figure 2.2. Diagram showing the definition of anisotropic permeability and effective permeability 2D and 3D. 22
Figure 2.3. Schematic plot of relative permeability-saturation relation. 25
Figure 2.4a. Diagram showing the definition of flow-direction-dependent longitudinal dispersivity in 2D. 38
Figure 2.4b,c. Diagram showing the definition of flow-direction-dependent longitudinal and transverse dispersivities in 3D. 39
Figures 3.1a,b. Illustration of finite-element meshes and elements in 2D and 3D. 53
Figures 3.1c,d. Illustration of a regular, non-aligned finite-element mesh, a non-aligned generalized hexahedral element, and an irregular, vertically aligned mesh in 3D. 54
Figure 3.2. Illustration of elementwise discretization of a spatially varying coefficient. 56
Figure 3.3. Illustration of nodewise discretization of a spatially varying coefficient. 57
Figure 3.4. Diagram showing cells, elements and nodes for a two-dimensional finite-element mesh composed of quadrilateral elements. 57
Figure 3.5. Schematic representation of a specified head (or pressure) boundary condition. 69
Figure 4.1. Illustration of 2D and 3D finite elements in the local coordinate system. 73
Figure 4.2. Plot showing perspectives of the 2D basis function. 75
Figure 4.3. Illustration of a 2D finite element in the local coordinate system with Gauss points. 84
Figure 5.1. Illustration of a finite-element mesh in radial coordinates. 107
Figure 5.2. Illustration of finite elements in 2D and 3D, with centroids. 111
Figure 5.3. Schematic diagram of SUTRA output to the ".lst" file. 117
Figure 5.4. Diagram showing SUTRA logic flow. 119
Figure 6.1. Diagram showing the radial finite-element mesh for the Theis example. 151
Figure 6.2. Plot showing match of Theis analytical solution with SUTRA solution. 153
Figure 6.3. Diagram showing the radial finite-element mesh for the constant-density solute and energy-transport examples. 154
Figure 6.4. Plot showing match of analytical solutions for radial solute transport of Hoopes and Harleman (1967), Gelhar and Collins (1971), and SUTRA. 155
Figure 6.5. Plot showing match of analytical solution for radial energy transport modified from Gelhar and Collins (1971) with SUTRA solution. 157
Figure 6.6. Diagram showing an idealized representation and finite-element mesh for the Rocky Mountain Arsenal example. 159
Figure 6.7. Plot showing the nearly steady-state conservative solute plume as simulated for the Rocky Mountain Arsenal example by SUTRA. 161
Figure 6.8. Plot showing the nearly steady-state solute plume (with solute half life ~ 20. years) as simulated for the Rocky Mountain Arsenal example by SUTRA. 161
Figure 6.9. Diagram showing boundary conditions and finite-element mesh for the Henry example. 163
Figure 6.10. Plot showing match of isochlors along the bottom of the aquifer for numerical results of Huyakorn and Taylor (1976) and SUTRA for the Henry example. 165
Figure 6.11. Plot showing match of isochlor contours for Henry analytical solution, INTERA code solution, and SUTRA solution. 166
Figure 6.12. Plot showing comparison of isochlor contours for the Henry problem obtained by various investigators. 166

Figure 6.13. Diagram showing radial two-dimensional finite-element mesh for the aquifer thermal energy storage example. 167
Figure 6.14. Plot of SUTRA results for aquifer thermal storage example after 30 days of hot water injection. 169
Figure 6.15. Plot of SUTRA results for aquifer thermal storage example after 90 days of hot water injection. 169
Figure 6.16. Plot of SUTRA results for aquifer thermal storage example after 30 days of pumping (120 days total elapsed time). 170
Figure 6.17. Plot of SUTRA results for aquifer thermal storage example after 60 days of pumping (150 days total elapsed time). 170
Figure 6.18. Plot of SUTRA results for aquifer thermal storage example after 90 days of pumping (180 days total elapsed time). 170
Figure 6.19. Plot showing propagation of moisture front for unsaturated flow and solute-transport example. 173
Figure 6.20. Plot showing propagation of solute slug for unsaturated flow and solute-transport example. 173
Figure 6.21. Diagram showing boundary conditions and finite-element mesh for the 2D formulation of the island problem. 175
Figure 6.22. Diagram showing plan view of the finite-element mesh for the 3D formulation of the island problem. 175
Figure 6.23. Diagram showing oblique view of the finite-element mesh for the 3D model of the island problem. 176
Figure 6.24. Plot showing comparison of results from the 2D and 3D models of the island problem; solute concentrations at t = 20 yr. 178
Figure 6.25. Plot showing comparison of results from the 2D and 3D models of the island problem; water saturations and water table below sea level at t = 20 yr. 178
Figure 6.26. Plot showing areal view of results from the 3D model of the island problem; solute concentrations at 35m below sea level at t = 20 yr. 179
Figure 7.1. Illustration showing minimization of bandwidth by careful numbering of nodes. .. 200
Figure B.1. Illustration showing allocation of sources and boundary fluxes in equal-sized elements. 263

CONVERSION FACTORS

Multiply	By	To obtain
kilograms (kg)	2.205	pounds, avoirdupois (lb)
meters (m)	3.281	feet (ft)
Joules (J)	0.2389	calories (cal)
Pascals (Pa)	1.450×10^{-4}	pounds per square inch (psi)
cubic meters (m^3)	264.2	U.S. gallons (gal)

Temperature in degrees Celsius (°C) can be converted to degrees Fahrenheit (°F) using the following equation:

$$°F = 1.8(°C) + 32$$

Abstract

SUTRA (***S***aturated-***U***nsaturated ***Tra***nsport) is a computer program that simulates fluid movement and the transport of either energy or dissolved substances in a subsurface environment. The original version of SUTRA was released in 1984 (Voss, 1984). The version described in this report (SUTRA 2.1) is an upgrade that adds to the previous version (SUTRA 2.0, originally designated as version 2D3D.1) (Voss and Provost, 2002) the ability to use irregular 3D meshes, conveniently incorporate input data from separate files, define schedules to control time stepping and observation output, interpolate observations in time and space, and output observations in an alternative format. The code employs a two- or three-dimensional finite-element and finite-difference method to approximate the governing equations that describe the two interdependent processes that are simulated:

1) fluid density-dependent saturated or unsaturated ground-water flow; and
2) either
   (a) transport of a solute in the ground water, in which the solute may be subject to: equilibrium adsorption on the porous matrix, and both first-order and zero-order production or decay; or
   (b) transport of thermal energy in the ground water and solid matrix of the aquifer.

SUTRA may also be used to simulate simpler subsets of the above processes. As the primary calculated result, SUTRA provides fluid pressures and either solute concentrations or temperatures, as they vary with time, everywhere in the simulated subsurface system.

SUTRA flow simulation may be employed for two-dimensional (2D) areal, cross sectional and three-dimensional (3D) modeling of saturated ground-water flow systems, and for cross sectional and 3D modeling of unsaturated zone flow. Solute-transport simulation using SUTRA may be employed to model natural or man-induced chemical-species transport including processes of solute sorption, production, and decay. For example, it may be applied to analyze ground-water contaminant transport problems and aquifer restoration designs. In addition, solute-transport simulation with SUTRA may be used for modeling of variable-density leachate movement, and for cross sectional modeling of saltwater intrusion in aquifers at near-well or regional scales, with either dispersed or relatively sharp transition zones between freshwater and saltwater. SUTRA energy-transport simulation may be employed to model thermal regimes in aquifers, subsurface heat conduction, aquifer thermal-energy storage systems, geothermal reservoirs, thermal pollution of aquifers, and natural hydrogeologic convection systems.

Mesh construction, which is quite flexible for arbitrary geometries, employs quadrilateral finite elements in 2D Cartesian or radial-cylindrical coordinate systems, and hexahedral finite elements in 3D systems. Permeabilities may be anisotropic and may vary in both direction and magnitude throughout the system, as may most other aquifer and fluid properties. Boundary conditions, sources and sinks may be time dependent. A number of input data checks are made to verify the input data set. An option is available for storing intermediate results and restarting a simulation at the intermediate time. Output options include fluid velocities, fluid mass and solute mass or energy budgets, and time-varying observations at points in the system. Both the mathematical basis for SUTRA and the program structure are highly general, and are modularized to allow for straightforward addition of new methods or processes to the simulation. The FORTRAN-90 coding stresses clarity and modularity rather than efficiency, providing easy access for later modifications.

INTRODUCTION

Chapter 1: Introduction

1.1 Purpose and Scope

SUTRA (***S***aturated-***U***nsaturated ***Tra***nsport) is a computer program that simulates fluid movement and the transport of either energy or dissolved substances in a subsurface environment. The original version of SUTRA was released in 1984 (Voss, 1984). The version described in this report (SUTRA 2.1) is an upgrade that adds to the previous version (SUTRA 2.0, originally designated as 2D3D.1) (Voss and Provost, 2002) the ability to use irregular 3D meshes, conveniently incorporate input data from separate files, define schedules to control time stepping and observation output, interpolate observations in time and space, and output observations in an alternative format. The code employs a 2D or 3D finite-element and finite-difference method to approximate the governing equations that describe the two interdependent processes that are simulated:

1) Fluid-density-dependent saturated or unsaturated ground-water flow; and either
2) (a) transport of a solute in the ground water, in which the solute may be subject to: equilibrium adsorption on the porous matrix, and both first-order and zero-order production or decay; or
   (b) transport of thermal energy in the ground water and solid matrix of the aquifer.

SUTRA provides, as the primary calculated result, fluid pressures and either solute concentrations or temperatures, as they vary with time, everywhere in the simulated subsurface system. SUTRA may also be used to simulate simpler subsets of the above process.

This report describes the physical-mathematical basis and the numerical methodology of the SUTRA computer code. The report can be divided into four groups, which may be read depending on the reader's background and interest.

The overview of simulation with SUTRA and methods may be obtained from
Chapter 1—Introduction.

The basics, at a fundamental level, for a reader who will carry out simulations with SUTRA, may be obtained by additional reading of
Chapter 2—Physical-Mathematical Basis of SUTRA Simulation, which gives a complete and detailed description of processes that SUTRA simulates and describes each physical parameter required by SUTRA input data;
Chapter 3—Fundamentals of Numerical Algorithms, which gives an introduction to the numerical aspects of simulation with SUTRA;
Chapter 6—Simulation Examples; and
Chapter 7—Simulation Setup.

The complete details of SUTRA methodology are given in the following additional sections:
Chapter 4—Numerical Methods; and
Chapter 5—Other Methods and Algorithms.
Chapter 4 provides the detail upon which program modifications may be based, and portions of Chapter 5 are valuable background for certain simulation applications.

Additional details are contained in the appendices:
Appendix A—List of Symbols; and

Appendix B— SUTRA Input Data List.

Appendix A contains a complete listing of all nomenclature and symbols used in this report. Appendix B describes in detail the SUTRA input datasets.

1.2 The Model

SUTRA is based on a general physical, mathematical and numerical structure implemented in the computer code in a modular design. This allows straightforward modifications and additions to the code. Eventual modifications may be, for example, the addition of nonequilibrium sorption (such as two-site models), equilibrium chemical reactions or chemical kinetics, or addition of over- and underburden heat-loss functions, a wellbore model, or confining bed leakage.

The SUTRA model stresses general applicability, numerical robustness and accuracy, and clarity in coding. Computational efficiency is somewhat diminished to preserve these qualities. The modular structure of SUTRA, however, allows implementation of any eventual changes that may improve efficiency. Such modifications may be in the configuration of the matrix equations, in the solution procedure for these equations, or in the finite-element integration routines. Furthermore, the general nature and flexibility of the input data allows easy adaptability to user-friendly and graphical input-output programming. The modular structure would also ease major changes such as modifications for simultaneous energy and solute-transport simulations.

1.3 SUTRA Processes

Simulation using SUTRA is in two or three spatial dimensions. A pseudo-3D quality is provided for 2D, in that the thickness of the 2D region in the third direction may vary from point to point. A 2D simulation may be done either in the areal plane or in a cross sectional view. The 2D spatial coordinate system may be either Cartesian (x,y) or radial-cylindrical (r,z). Areal simulation is usually physically unrealistic for variable-density fluid and for unsaturated flow problems. The 3D spatial coordinate system is Cartesian (x,y,z).

Ground-water flow is simulated through numerical solution of a fluid mass-balance equation. The ground-water system may be either saturated, or partly or completely unsaturated. Fluid density may be constant, or vary as a function of solute concentration or fluid temperature.

SUTRA tracks the transport of either solute mass or energy in flowing ground water through a unified equation, which represents the transport of either solute or energy. Solute transport is simulated through numerical solution of a solute mass-balance equation where solute concentration may affect fluid density. The single solute species may be transported conservatively, or it may undergo equilibrium sorption (through linear, Freundlich, or Langmuir isotherms). In addition, the solute may be produced or decay through first- or zero-order processes. Energy transport is simulated through numerical solution of an energy-balance equation. The solid grains of the aquifer matrix and fluid are locally assumed to have equal temperature, and fluid density and viscosity may be affected by the temperature.

Most aquifer material, flow, and transport parameters may vary in value throughout the simulated region. Sources and boundary conditions of fluid, solute and energy may be specified to vary with time or may be constant.

SUTRA dispersion processes include diffusion and two types of fluid velocity-dependent dispersion. The standard dispersion model for isotropic media assumes direction-independent values of longitudinal and transverse dispersivity. A flow-direction-dependent dispersion process for anisotropic media is also provided. This process assumes that longitudinal and transverse dispersivities vary depending on the orientation of the flow direction relative to the principal axes of aquifer permeability.

1.4 Some SUTRA Applications

SUTRA may be employed in one-, two-, or three-dimensional analyses. Flow and transport simulation may be either steady state, which requires only a single solution step, or transient, which requires a series of time steps in the numerical solution. Single-step steady-state solutions are usually not appropriate for nonlinear problems with variable density, saturation, viscosity or nonlinear sorption.

SUTRA flow simulation may be employed for 2D areal, cross sectional, and fully 3D modeling of saturated ground-water flow systems and unsaturated-zone flow. Hydraulic aquifer tests may be analyzed using flow simulation. SUTRA solute-transport simulation may be employed to model natural or man-induced chemical-species transport including processes of solute sorption, production and decay. Such simulation may be used to analyze ground-water contaminant-transport problems and aquifer restoration designs. SUTRA solute-transport simulation may also be used for modeling of variable-density leachate movement, and for cross sectional modeling of seawater intrusion and other saline-water migration in aquifers at near-well or regional scales with either dispersed or relatively sharp transition zones between freshwater and saltwater. SUTRA energy-transport simulation may be employed to model thermal regimes in aquifers, subsurface heat conduction, aquifer thermal-energy storage systems, geothermal reservoirs, thermal pollution of aquifers, and natural hydrogeologic convection systems. A review of published SUTRA applications is given in Voss (1999).

1.5 SUTRA Numerical Methods

SUTRA simulation is based on a hybridization of finite-element and integrated-finite-difference methods employed in the framework of a method of weighted residuals. The method is robust and accurate when employed with proper spatial and temporal discretization. Standard finite-element approximations are employed only for terms in the balance equations that describe fluxes of fluid mass, solute mass, and energy. All other nonflux terms are approximated with a finite-element mesh version of the integrated-finite-difference methods. The hybrid method is the simplest and most economical approach that preserves the mathematical elegance and geometric flexibility of finite-element simulation, while taking advantage of finite-difference efficiency.

SUTRA employs a special finite-element method for calculation of fluid velocities in variable density fluids. Fluid velocities, when calculated with standard finite-element methods for systems with variable fluid density, may display spurious numerically generated components within each element. These errors are due to fundamental numerical inconsistencies in spatial and temporal approximations for the pressure gradient and density-gravity terms, which are involved in velocity calculation. Spurious velocities can significantly add to the dispersion of solute or energy. This false dispersion makes accurate simulation of all systems impossible, except those with very low vertical concentration or temperature gradients, even when fine vertical spatial discretization is employed. Velocities as calculated in SUTRA, however, are

based on a consistent spatial and temporal discretization, which is described in this report and by Voss and Souza (1987). The consistently evaluated velocities allow stable and accurate transport simulation (even at steady state) for systems with large vertical gradients of concentration or temperature. An example of such a system that SUTRA successfully simulates is a cross sectional regional model of a coastal aquifer wherein the transition zone between horizontally flowing freshwater and deep stagnant saltwater is relatively narrow (Voss and Souza, 1987).

The time discretization used in SUTRA is based on a backwards finite-difference approximation for the time derivatives in the balance equations. Some nonlinear coefficients are evaluated at the new time level of solution by projection, and others are evaluated at the previous time level for noniterative solutions. All coefficients are evaluated at the new time level for iterative solutions.

The finite-element method used in SUTRA allows the simulation of irregular regions with irregular internal discretization in 2D and, beginning with this version, in 3D. This is made possible through use of quadrilateral elements with four corner nodes in 2D and hexahedral elements with eight corner nodes in 3D. Coefficients and properties of the system may vary in value throughout the mesh. Manual construction and data preparation for 2D and 3D meshes requires considerable labor; instead, preprocessing software such as the interactive graphical user interface SutraGUI (Winston and Voss, 2002) should be used for this purpose.

SUTRA includes an optional numerical method, based on asymmetric finite-element weighting functions, that results in "upstream weighting" of advective transport and unsaturated fluid flux terms. Although upstream weighting has been employed to achieve stable, non-oscillatory solutions to transport problems and unsaturated flow problems, the method is not recommended for general use as it merely changes the physical system being simulated by increasing the magnitude of the dispersion process. A practical use of the method is, however, to provide a simulation of the sharpest concentration or temperature variations possible with a given mesh. This is obtained by specifying a simulation with no physical diffusion or dispersion, and with 50% upstream weighting. The results may be interpreted as the solution with the minimum amount of dispersion possible for a stable result in the particular mesh in use. In general simulation analyses of transport, upstream weighting is discouraged. The normal non-upstream methods provided by SUTRA are based on symmetric weighting functions. These methods are robust and accurate when the finite-element mesh is properly designed for a particular simulation, and should be used for most transport simulations.

1.6 SUTRA as an Analytical Tool

SUTRA will provide clear, accurate answers only to well-posed, well-defined, and well-discretized simulation problems. In less well-defined systems, SUTRA simulation can help visualize a conceptual model of the flow and transport regime, and can aid in deciding between various conceptual models. In such less well-defined systems, simulation can help answer questions such as: Is an inaccessible aquifer boundary, which is ten kilometers offshore either leaky or impermeable? How leaky? Does this boundary affect the primary analysis of onshore water supply? This mode of modeling is called 'hypothesis testing.'

SUTRA is not useful for making exact predictions of future responses of typical hydrologic systems that are not well defined. Rather, SUTRA is useful for hypothesis testing and for helping to understand the physics of such a system. On the other hand, developing an understanding of a system based on simulation analysis can help make a set of worthwhile predictions that are

predicated on uncertainty of both the physical model design and model parameter values. In particular, transport simulation that relies on large amounts of dispersion must be considered an uncertain basis for prediction because of the idealized mathematical assumptions inherent in the SUTRA dispersion process.

Because a simulation-based prediction made with certainty is often inappropriate, an "if-then" prediction may be more realistic. A reasonable type of result of SUTRA simulation analysis may thus be: "Based on the uncertainty in location and type of boundary condition A, and uncertainty in the distribution of values for parameters B and C, the following predictions are made. The extreme, but reasonable combination of A, B, and C results in prediction X; the opposite reasonable extreme combination of A, B, and C results in prediction Y; the combination of best estimates of A, B, and C results in prediction Z, and is considered most likely."

In some cases, the available real data on a system may be so poor that a simulation using SUTRA is so ambiguously defined that no prediction at all can be made. In this instance, the simulation may be used to point out the need for particular types of data collection. The model could be used to advantage in visualizing possible regimes of system behavior rather than to determine which is accurate.

SUTRA FUNDAMENTALS

Chapter 2: Physical-Mathematical Basis of SUTRA Simulation

The physical mechanisms that drive thermal energy transport and solute transport in the subsurface environment are described by nearly identical mathematical expressions. SUTRA takes advantage of this similarity, and with a simple program structure provides for simulation of either energy or solute transport. In fact, SUTRA simulation combines two physical models, one to simulate the flow of ground water, and the second to simulate the movement of either thermal energy or a single solute in the ground water.

Note: All symbols are defined in **Appendix A – List of Symbols**.

The primary variable upon which the flow model is based is fluid pressure, p [$M/(L \cdot s^2)$] = p(x,y[,z],t). The latter expression means for 2D, p(x,y,t), and for 3D, p(x,y,z,t). Pressure may vary spatially in the ground-water system, as well as with time. Pressure is expressed as a combination of fluid mass units, [M], length units, [L], and time units in seconds, [s]. Fluid density may vary depending on the local value of fluid temperature or solute concentration. Variation in fluid density, aside from fluid pressure differences, may itself drive flows. The effects of gravity acting on fluids with different density must therefore be accounted for in the flow field.

The flow of ground water, in turn, is a fundamental mechanism upon which the physical models of energy transport and solute transport are based. The primary variable characterizing the thermal energy distribution in a ground-water system is fluid temperature, T [°C] = T(x,y[,z],t), in degrees Celsius, which may vary spatially and with time. The primary variable characterizing the state of solute distribution in a ground-water system is solute mass fraction, $C[M_s/M]$ = C(x,y[,z],t), which may also vary spatially and with time. The units are a ratio of solute mass, $[M_s]$ to fluid mass, [M]. The term "solute mass fraction" may be used interchangeably with "solute concentration", and no difference should be implied. Note that "solute volumetric concentration", $c[M_s/L_f^3]$, (mass of solute, M_s, per volume of fluid, L_f^3), is not the primary variable characterizing solute transport referred to either in this report or in output from the SUTRA model. Note that the measure of solute mass $[M_s]$ may be in units such as [mg], [kg], or [lbm], and may differ from the measure, [M], of fluid mass.

SUTRA allows only the transport of either thermal energy or a single solute to be modeled in a given simulation. Thus, when simulating energy transport, a constant value of solute concentration is assumed in the ground water. When simulating solute transport, a constant ground-water temperature is assumed.

When SUTRA simulation is carried out in two space dimensions, parameters vary only in these two directions (x,y). However, the region of space to be simulated may be defined as 3D, when the assumption is made that all SUTRA parameters and coefficients have a constant value in the third space direction. A SUTRA simulation may be carried out over a region defined over two space coordinates (x,y) in which the thickness of the region measured in the third coordinate direction (z) varies depending on (x,y) position.

2.1 Physical Properties of Solid Matrix and Fluid

Fluid physical properties

The ground-water fluid density and viscosity may vary depending on pressure, temperature and solute concentration. These fundamental variables are defined as follows:

p(x,y[,z],t)	$[M/(L\cdot s^2)]$	fluid pressure
T(x,y[,z],t)	[ºC]	fluid temperature (degrees Celsius)
C(x,y[,z],t)	$[M_S/M]$	fluid solute mass fraction (or solute concentration) (mass solute per mass total fluid)

As a point of reference, "solute volumetric concentration" is defined in terms of fluid density, ρ:

c(x,y[,z],t)	$[M_S/L_f^3]$	solute volumetric concentration (mass solute per volume total fluid)
ρ(x,y[,z],t)	$[M/L_f^3]$	fluid density

$$c = \rho C \tag{2.1}$$

$$\rho = \rho_w + c \tag{2.2}$$

Total fluid density is the sum of pure water density, ρ_w, and c. Note again that "solute concentration" refers to solute mass fraction, C, and not c. Fluid density is a weak function of pressure and depends primarily upon fluid solute concentration and temperature. The approximate density models employed by SUTRA are first order Taylor expansions (in either T or C) about a base (reference) density, but other density models may be substituted through minor modifications to the program. For energy transport:

$$\rho = \rho(T) \cong \rho_o + \frac{\partial \rho}{\partial T}\left(T - T_o\right) \tag{2.3}$$

ρ_o	$[M/L_f^3]$	base fluid density at $T=T_o$
T_o	[°C]	base fluid temperature

where ρ_o is the base fluid density at a base (reference) temperature of T_o, and $\partial\rho/\partial T$ is a constant value of density change with temperature. For the range 20°C to 60°C, $\partial\rho/\partial T$ is approximately -0.375 $[kg/(m^3\cdot°C)]$; however, this factor varies and should be carefully chosen for the temperature range of interest.

For solute transport:

$$\rho = \rho(C) \cong \rho_o + \frac{\partial \rho}{\partial C}\left(C - C_o\right) \tag{2.4}$$

ρ_o	$[M/L_f^3]$	base fluid density at $C=C_o$
C_o	$[M_S/M]$	base fluid solute concentration

where ρ_o is the base fluid density at base concentration, C_o. (Usually, $C_o = 0$, and the base density is that of pure water.) The factor $\partial\rho/\partial C$ is a constant value of density change with concentration. For example, for mixtures of freshwater and seawater at 20°C, when C is the mass fraction of total dissolved solids, $C_o = 0$, and $\rho_o = 998.2$ [kg/m^3], then the factor, $\partial\rho/\partial C$, is approximately 700. [kg/m^3].

Fluid viscosity, μ [$M/L_f \cdot s$], is a weak function of pressure and of concentration (for all except very high concentrations), and depends primarily on fluid temperature. For energy transport the viscosity of pure water is given in m-k-s units by:

$$\mu(T) \cong (239.4 \times 10^{-7})\, 10^{\left(\frac{248.37}{T+133.15}\right)} \qquad \left[\text{kg}/(\text{m} \cdot \text{s})\right] \tag{2.5}$$

(The units may be converted to those desired via a scale factor in the program's input data.) For solute transport, viscosity is taken to be constant. For example, at 20°C in m-k-s units,

$$\mu(C)\big|_{T=20°C} = 1.0 \times 10^{-3} \qquad [\text{kg}/(\text{m} \cdot \text{s})] \tag{2.6}$$

Properties of fluid within the solid matrix

The total volume of a porous medium is composed of a matrix of solid grains typically of solid earth materials, and of void space, which includes the entire remaining volume that the solid does not fill. The volume of void space may be fully or partly filled with gas or liquid, and is commonly referred to as the pore volume. Porosity is defined as a volume of voids in the soil matrix per total volume of voids plus matrix:

ε(x,y[,z],t)	[1]	porosity (volume of voids per total volume)

where [1] indicates a dimensionless quantity.

It should be noted that SUTRA employs only one type of porosity, ε. In some instances there may be need to distinguish between a porosity for pores which take part in fluid flow (effective porosity) and pores which contain both stagnant and flowing fluid (total porosity). (Modifications may be made by the user to include this process.)

The fraction of total volume filled by the fluid is εS_w where:

$S_w(x,y[,z],t)$	[1]	water saturation (saturation) (volume of water per volume of voids)

When $S_w = 1$, the void space is completely filled with fluid and is said to be saturated. When $S_w < 1$, the void space is only partly water filled and is referred to as being unsaturated.

When $S_w < 1$, water adheres to the surface of solid grains by surface tension effects, and the fluid pressure is less than atmospheric. Fluid pressure, p, is measured with respect to background or atmospheric pressure. The negative pressure is defined as capillary pressure, which exists only for $p < 0$:

$p_c(x,y[,z],t)$	$[M/(L\cdot s^2)]$	capillary pressure

$$p_c = -p \quad \text{when } p < 0$$

$$p_c = 0 \quad \text{when } p \geqq 0 \qquad (2.7)$$

In a saturated porous medium, as fluid (gauge) pressure drops below zero, air may not directly enter the void space, but may enter suddenly when a critical capillary pressure is reached. This pressure, p_{cent}, is the entry pressure (or bubble pressure):

p_{cent}	$[M/(L\cdot s^2)]$	entry capillary pressure

Typical values for p_{cent} range from about 1.0×10^3 $[kg/(m\cdot s^2)]$ for coarse sand to approximately 5.0×10^3 $[kg/(m\cdot s^2)]$ for fine silty sand.

The relation between fluid saturation and capillary pressure in a given medium is typically determined by laboratory experiment, and except for the portion near bubble pressure, tends to have an exponential character (Figure 2.1). Different functional relations exist for different materials as measured in the laboratory. In addition, a number of general functions with parameters to be fitted to laboratory data are available. Because of the variety of possible functions, no particular function is set by SUTRA; any desired function may be specified for simulation of unsaturated flow. For example, a general function with three fitting parameters is (Van Genuchten, 1980):

$$S_w = S_{wres} + (1 - S_{wres})\left[\frac{1}{1 + (ap_c)^n}\right]^{\left(\frac{n-1}{n}\right)} \qquad (2.8)$$

where S_{wres} is a residual saturation below which saturation is not expected to fall (because the fluid becomes immobile), and both a and n are parameters. The values of these parameters depend upon a number of factors and these must be carefully chosen for a particular material.

The total mass of fluid contained in a total volume, VOL, of solid matrix plus pore space is $(\varepsilon S_w \rho)$VOL. The actual amount of total fluid mass contained depends solely on fluid pressure, p, and solute concentration, C, or fluid temperature, T. A change in total fluid mass in a volume, assuming VOL is constant, is expressed as follows:

$$\mathrm{VOL}\cdot \mathrm{d}\left(\varepsilon S_w \rho\right) = \mathrm{VOL}\cdot\left[\frac{\partial\left(\varepsilon S_w \rho\right)}{\partial p}\,\mathrm{dp} + \frac{\partial\left(\varepsilon S_w \rho\right)}{\partial U}\,\mathrm{dU}\right] \tag{2.9}$$

where U represents either C or T. Saturation, S_w, is entirely dependent on fluid pressure, and porosity, ε, does not depend on concentration or temperature:

$$\mathrm{VOL}\cdot \mathrm{d}\left(\varepsilon S_w \rho\right) = \mathrm{VOL}\cdot\left[\left(S_w\,\frac{\partial(\varepsilon\rho)}{\partial p} + \varepsilon\rho\,\frac{\partial S_w}{\partial p}\right)\mathrm{dp} + \varepsilon S_w\,\frac{\partial \rho}{\partial U}\,\mathrm{dU}\right] \tag{2.10}$$

The factor, $\partial S_w/\partial p$, is obtained by differentiation of the chosen saturation-capillary pressure relation. For the example function given as (2.8),

$$\frac{\mathrm{dS_w}}{\mathrm{dp}} = \left[\frac{a(n-1)(1-S_{wres})(ap_c)^{(n-1)}}{\left(1+(ap_c)^n\right)^{(2n-1)/n}}\right] \tag{2.11}$$

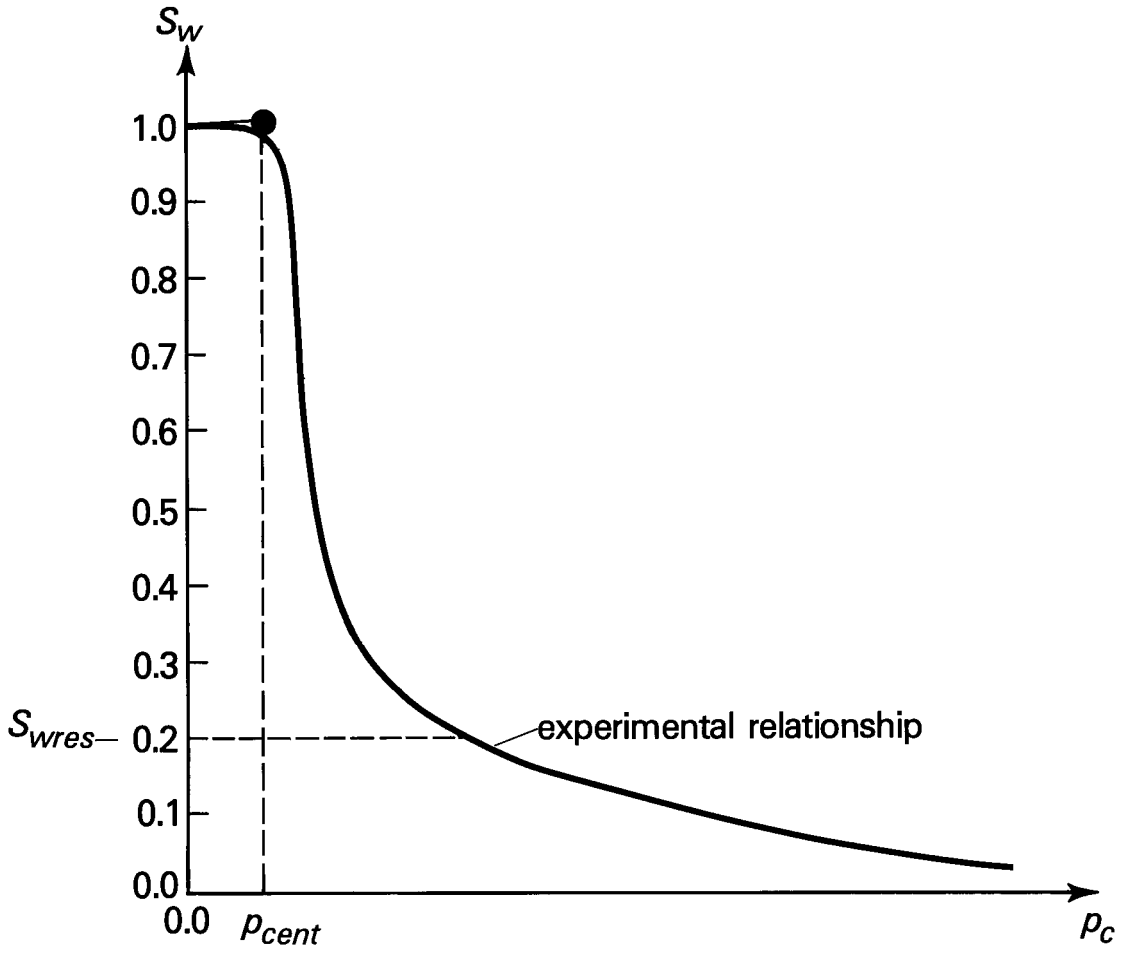


Figure 2.1. Saturation-capillary pressure relation (schematic). S_w is saturation, S_{wres} is residual saturation, and p_{cent} is air entry pressure (bubble pressure).

The factor, $\partial\rho/\partial U$, is a constant value defined by the assumed density models, given by equations (2.3) and (2.4).

Aquifer storativity under fully saturated conditions is related to the factor, $\partial(\varepsilon\rho)/\partial p$, by definition, as follows (Bear, 1979):

$$\frac{\partial(\varepsilon\rho)}{\partial p} \equiv \rho S_{op} \tag{2.12}$$

where:

$$S_{op} \equiv \frac{1}{VOL}\left(\frac{\Delta VOL_w}{\Delta p}\right) \tag{2.13}$$

$S_{op}(x,y)$ $[M/(L\cdot s^2)]^{-1}$ specific pressure storativity

The specific pressure storativity, S_{op}, is the volume of water released from saturated pore storage due to a unit drop in fluid pressure per total solid matrix plus pore volume. Note that the common specific storativity, S_o $[L^{-1}]$, which when multiplied by confined aquifer thickness gives the well known storage coefficient, S[1], is related to S_{op} as, $S_o = \rho|\underline{g}|S_{op}$, where $|\underline{g}|$ $[L/s^2]$ is the magnitude of the gravitational acceleration vector. The common specific storativity, S_o, is analogous to the specific pressure storativity, S_{op}, used in SUTRA, except that S_o expresses the volume of water released from pore storage due to a unit drop in hydraulic head.

SUTRA employs an expanded form of the specific pressure storativity based on fluid and bulk porous matrix compressibilities. The relation is obtained as follows by expanding (2.12):

$$\rho S_{op} \equiv \rho\frac{\partial\varepsilon}{\partial p} + \varepsilon\frac{\partial\rho}{\partial p} \tag{2.14}$$

The coefficient of compressibility of water is defined by

$$\beta \equiv \frac{1}{\rho}\frac{\partial\rho}{\partial p} \tag{2.15}$$

β $[M/(L\cdot s^2)]^{-1}$ fluid compressibility

which allows the last term of (2.14) to be replaced by $\varepsilon\rho\beta$.

For pure water at 20°C, $\beta \sim 4.47 \times 10^{-10}$ $[kg/(m\cdot s^2)]^{-1}$. As the volume of solid grains, VOL_s, in a volume, VOL, of porous solid matrix plus void space is $VOL_s = (1-\varepsilon)\cdot VOL$, the factor, $\partial\varepsilon/\partial p$, may be expressed as:

$$\frac{\partial\varepsilon}{\partial p} \equiv \frac{(1-\varepsilon)}{VOL}\frac{\partial(VOL)}{\partial p} \tag{2.16}$$

which assumes that individual solid grains are relatively incompressible. The total stress at any point in the solid matrix-fluid system is the sum of effective (intergranular) stress, σ' [M/(L·s^2)], and fluid pore pressure, p, in systems where the total stress remains nearly constant, $d\sigma' = -dp$, and any drop in fluid pressure increases intergranular stress by a like amount. This consideration allows (2.16) to be expressed in terms of bulk porous matrix compressibility, as:
$\partial\varepsilon/\partial p = (1-\varepsilon)\alpha$, where

$$\alpha \equiv -\frac{1}{VOL}\frac{\partial(VOL)}{\partial\sigma'} \tag{2.17}$$

α	[M/(L·s^2)]$^{-1}$	porous matrix compressibility
σ'	[M/(L·s^2)]	intergranular stress

Factor α ranges from $\alpha \sim 10^{-10}$ [kg/(m·s^2)]$^{-1}$ for sound bedrock to about $\alpha \sim 10^{-7}$ [kg/(m·s^2)]$^{-1}$ for clay (Freeze and Cherry, 1979). Thus equation (2.14) may be rewritten as
$\rho S_{op} = \rho(1-\varepsilon)\alpha + \varepsilon\rho\beta$, and, in effect, the specific pressure storativity, S_{op}, is expanded as:

$$S_{op} = (1-\varepsilon)\alpha + \varepsilon\beta \tag{2.18}$$

The porosity value itself is held constant for SUTRA, despite relation (2.16), although this may cause small errors in some cases (Goode, 1992). A more thorough discussion of storativity is presented by Bear (1979).

2.2 Saturated-Unsaturated Ground-Water Flow

Fluid flow and flow properties

Fluid movement in porous media where fluid density varies spatially may be driven by differences either in fluid pressure or by unstable variations in fluid density. Pressure-driven flows, for example, are directed from regions of higher than hydrostatic fluid pressure toward regions of lower than hydrostatic pressure. Density-driven flows occur when gravity forces act on denser regions of fluid causing them to flow downward relative to fluid regions that are less dense. A stable density configuration drives no flow, and is one in which fluid density remains constant or increases with depth.

The mechanisms of pressure and density driving forces for flow are expressed for SUTRA simulation by a general form of Darcy's law, which is commonly used to describe flow in porous media:

$$\underline{v} = -\left(\frac{\underline{\underline{k}}k_r}{\varepsilon S_w \mu}\right) \cdot \left(\nabla p - \rho \underline{g}\right) \tag{2.19a}$$

where:

$\underline{v}$ (x,y[,z],t)	[L/s]	average fluid velocity
$\underline{\underline{k}}$ (x,y[,z])	[L^2]	solid matrix permeability (in 2D, a 2 x 2 matrix of values; in 3D, a 3 x 3 matrix of values)
k_r (x,y[,z],t)	[1]	relative permeability to fluid flow (assumed to be independent of direction)
$\underline{g}$	[L/s^2]	gravitational acceleration (gravity vector)

The gravity vector is defined in relation to the direction in which vertical elevation is measured:

$$\underline{g} = -\left|\underline{g}\right| \underline{\nabla}(\text{ELEVATION}) \qquad (2.19b)$$

where $\left|\underline{g}\right|$ is the magnitude of the gravitational acceleration vector. For example, if the y-space-coordinate is oriented directly upwards, then $\underline{\nabla}$(ELEVATION) is a vector of values (for x, y and z directions, respectively): (0,1,0), and $g = (0, -\left|\underline{g}\right|, 0)$. If for example, 'directly upwards' is within in the x-y plane at a 60° angle to the x-axis, then $\underline{\nabla}$ (ELEVATION) = $((1/2), (3^{1/2}/2), 0)$ and $\underline{g} = (-(1/2)\left|\underline{g}\right|, -(3^{1/2}/2)\left|\underline{g}\right|, 0)$. The z-component is ignored for 2D analysis.

The average fluid velocity, $\underline{v}$, is the velocity of fluid with respect to the stationary solid matrix. The velocity is referred to as an "average", because true velocities in a porous medium vary from point to point due to variations in the permeability and porosity of the medium at a spatial scale smaller than that at which measurements are made. The so-called Darcy velocity, $\underline{q}$, for the sake of reference, is $\underline{q} = \varepsilon S_w \underline{v}$. This value is always less than the true average fluid velocity, $\underline{v}$, and thus, is not a true indicator of the speed of water movement. "Darcy velocity", $\underline{q}$, is actually a 'flux' of fluid, representing the volume of fluid crossing an area of porous medium per time.

Fluid velocity, even for a given pressure and density distribution, may take on different values depending on how mobile the fluid is within the solid matrix. Fluid mobility depends on the combination of permeability, $\underline{\underline{k}}$, relative permeability, k_r, and viscosity, μ, which occurs in equation (2.19a). Permeability is a measure of the ease of fluid movement through interconnected voids in the solid matrix when all voids are completely saturated. Relative permeability expresses what fraction of the total permeability remains when the voids are only partly fluid-filled and only part of the total interconnected void space is connected by continuous fluid channels. Viscosity directly expresses ease of fluid flow; a less viscous fluid flows more readily under a driving force.

As a point of reference, in order to relate the general form of Darcy's law, (2.19a), back to a better known form dependent on hydraulic head, the dependence of flow on density and saturation must be ignored. When the solid matrix is fully saturated, $S_w = 1$, the relative

permeability to flow is unity $k_r = 1$. When, in addition, fluid density is constant, the right side of (2.19a) expanded by (2.19b) may be multiplied and divided by $\rho|\underline{g}|$:

$$\underline{v} = \frac{-\underline{\underline{k}}\rho|\underline{g}|}{\varepsilon\mu} \cdot \left[\underline{\nabla}\left(\frac{p}{\rho|\underline{g}|} \right) + \underline{\nabla}(\text{ELEVATION}) \right] \tag{2.20a}$$

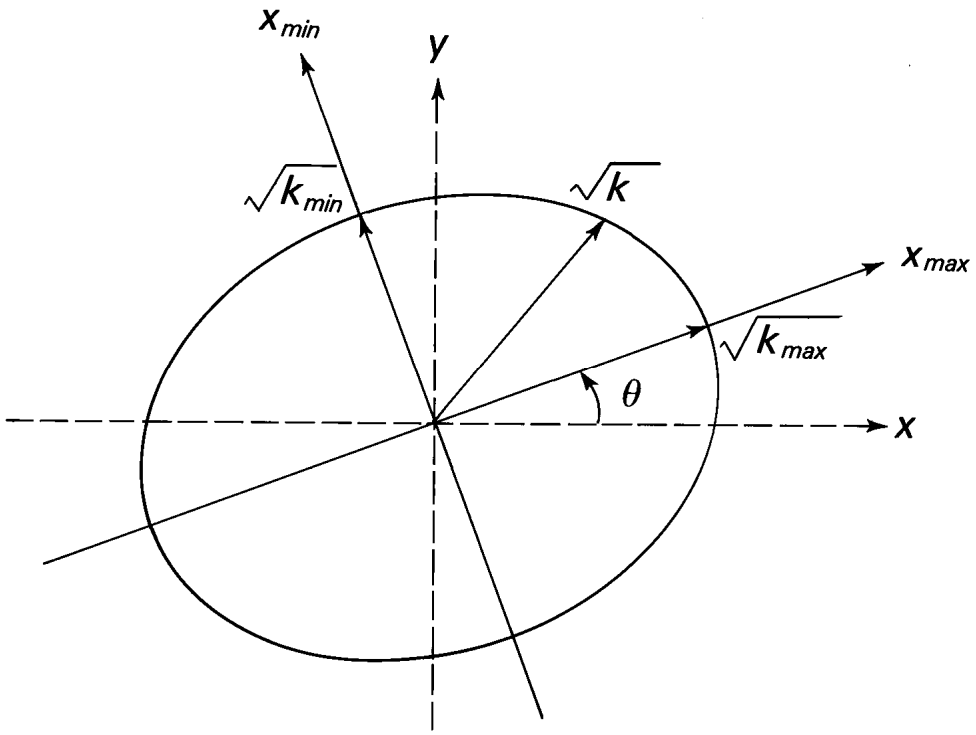
The hydraulic conductivity, $\underline{\underline{K}}(x,y[,z],t)$ [L/s], may be identified in this equation as $\underline{\underline{K}} = (\underline{\underline{k}}\rho|\underline{g}|)/\mu$; pressure head, $h_p(x,y[,z],t)$ [L], is $h_p = p/(\rho|\underline{g}|)$. Hydraulic head, $h(x,y[,z],t)$ [L], is defined as $h = h_p + \text{ELEVATION}$. Thus, for constant density, saturated flow,

$$\underline{v} = -\left(\frac{\underline{\underline{K}}}{\varepsilon} \right) \cdot \underline{\nabla}\, h \tag{2.20b}$$

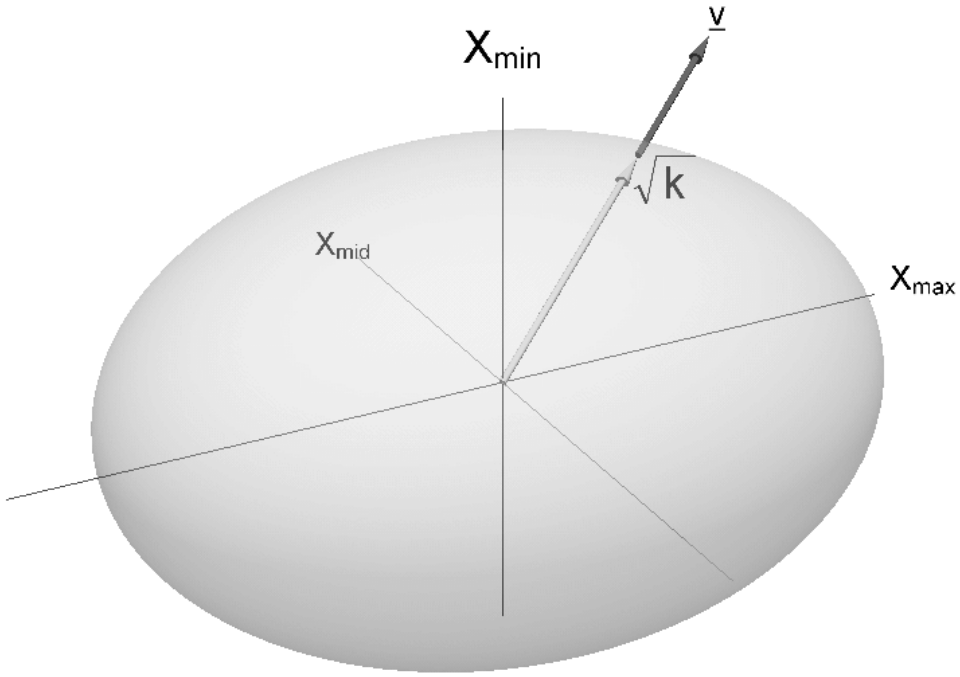
which is Darcy's law written in terms of the hydraulic head. Even in this basic form of Darcy's law, flow may depend on solute concentration and temperature. The hydraulic conductivity, through viscosity, is highly dependent on temperature, and measurably, but considerably less on concentration. In cases where density or viscosity is not constant, therefore, hydraulic conductivity, $\underline{\underline{K}}$, is not a fundamental parameter describing ease of flow through the solid matrix. Permeability, $\underline{\underline{k}}$, is, in most situations, essentially independent of pressure, temperature and concentration and therefore is the appropriate fundamental parameter describing ease of flow in the SUTRA model.

Permeability, $\underline{\underline{k}}$, describes ease of fluid flow in a saturated solid matrix. When permeability in a particular small volume of solid matrix depends on the direction of the flow, the permeability is said to be anisotropic. Direction-independent permeability is called isotropic. It is commonly assumed that permeability is the same for flow forward or backward along a particular line in space. When permeability is anisotropic, there is always one particular direction, x_{max}, along which permeability has a maximum value, k_{max} [L^2], and another direction, x_{min}, along which permeability has a minimum value, k_{min} [L^2]. These two principal directions are mutually perpendicular. In 3D, there exists a third principal direction, x_{mid}, which is perpendicular to the first two, and in which the permeability has an intermediate, or "middle," value, k_{mid} [L^2].

The permeability tensor, $\underline{\underline{k}}$, of Darcy's law, equation (2.19), has four components in 2D and nine components in 3D. The values of these tensorial components depend on the effective permeabilities in the x, y, and z coordinate directions, which are not necessarily aligned with the principal directions of permeability. The required coordinate rotations are carried out automatically by SUTRA according to the method described in section 5.1, "Rotation of Permeability Tensor."



(a)



(b)

Figure 2.2. Definition of anisotropic permeability and effective permeability, k, in (a) 2D and (b) 3D. In 2D, x_{max} and x_{min} are the directions of maximum and minimum permeability, k_{max} and k_{min}, and θ is the angle of the maximum permeability direction from the x-axis. In 3D, x_{mid} is an additional direction that corresponds to a middle permeability, k_{mid}. (For the sake of visual clarity, the 3D diagram omits the (x, y, z)-coordinate axes and the angles that relate coordinate systems.)

At any given point in space, an anisotropic permeability field in 2D is completely described by the permeability ellipse shown in Figure 2.2a, in which

$k_{max}(x,y)$	$[L^2]$	absolute maximum value of permeability
$k_{min}(x,y)$	$[L^2]$	absolute minimum value of permeability
$\theta(x,y)$	[°]	angle from +x-coordinate axis to direction of maximum permeability, x_{max}

The lengths of the semi-major and semi-minor axes of the ellipse are defined as $k_{max}^{1/2}$ and $k_{min}^{1/2}$, respectively, and the length of any radius is $k^{1/2}$, where k is the effective permeability for flow along that direction. The angle θ orients the principal directions, x_{max}, and x_{min}, to the x and y coordinate directions. In the case of isotropic permeability, $k_{max} = k_{min}$, and θ is arbitrary.

In 3D, an anisotropic permeability field is completely described by the permeability ellipsoid shown in Figure 2.2b, in which

$k_{max}(x,y,z)$	$[L^2]$	maximum value of permeability
$k_{mid}(x,y,z)$	$[L^2]$	middle value of permeability
$k_{min}(x,y,z)$	$[L^2]$	minimum value of permeability
$\theta_1(x,y,z)$	[°]	first angle used to define the directions of maximum, middle, and minimum permeability
$\theta_2(x,y,z)$	[°]	second angle used to define the directions of maximum, middle, and minimum permeability
$\theta_3(x,y,z)$	[°]	third angle used to define the directions of maximum, middle, and minimum permeability

The lengths of the principal semi-axes of the ellipsoid are defined as $k_{max}^{1/2}$, $k_{mid}^{1/2}$, and $k_{min}^{1/2}$, and the length of any radius is $k^{1/2}$, where k is the effective permeability for flow along that direction. The angles θ_1, θ_2, and θ_3 orient the principal directions, x_{max}, x_{mid}, and x_{min}, to the x, y, and z coordinate directions. In the case of isotropic permeability, $k_{max} = k_{mid} = k_{min}$, and θ_1, θ_2, and θ_3 are arbitrary. **For the definition of θ_1, θ_2, and θ_3, see the note titled "Permeability" in the description of dataset 15B in Appendix B, in which they are called ANGLE1, ANGLE2, and ANGLE3, respectively.**

The discussion of isotropic and anisotropic permeability, $\underline{\underline{k}}$, applies as well to flow in an unsaturated solid matrix, $S_w < 1$, although unsaturated flow has additional unique properties which require definition. When fluid capillary pressure, p_c, is less than entry capillary pressure, p_{cent}, the void space is saturated $S_w = 1$, and local porous medium flow properties are not pressure-dependent but depend only on void space geometry and connectivity. When $p_c > p_{cent}$,

then air or another gas has entered the matrix and the void space is only partly fluid filled, $S_w < 1$. In this case, the ease with which fluid can pass through the solid matrix depends on the remaining cross section of well-connected fluid channels through the matrix, as well as on surface tension forces at fluid-gas, and fluid-solid interfaces. When saturation is so small such that no interconnected fluid channels exist and residual fluid is scattered about and tightly bound in the smallest void spaces by surface tension, flow ceases entirely. The relative permeability to flow, k_r, which is a measure of this behavior, varies from a value of zero or near zero at the residual fluid saturation, S_{wres}, to a value of one at saturation, $S_w = 1$. A relative permeability-saturation relation (Figure 2.3) is typically determined for a particular solid matrix material in the laboratory as is the relation, $S_w(p_c)$. Relative permeability is assumed in SUTRA to be independent of direction in the porous media.

SUTRA allows any desired function to be specified which gives the relative permeability in terms of saturation or pressure. A general function, for example, based on the saturation-capillary pressure relation given as an example in (2.8), is (Van Genuchten, 1980):

$$k_r = S_w^{*\,1/2}\left\{1-\left[1-S_w^{*\left(\frac{n}{n-1}\right)}\right]^{\left(\frac{n-1}{n}\right)}\right\}^2 \qquad (2.21a)$$

where the a dimensionless saturation, S_w^* is given by:

$$S_w^* = \frac{S_w - S_{wres}}{1 - S_{wres}} \qquad (2.21b)$$

Flow in the gaseous phase that fills the remaining void space not containing fluid when $S_w < 1$ is assumed not to contribute significantly to total solute or energy transport which is due primarily to fluid flow and other transport processes through both fluid and solid matrix. Furthermore, it is assumed that pressure differences within the gas do not drive significant fluid flow. These assumptions are justified in most common situations when gas pressure is approximately constant throughout the solid matrix system. Should gas pressure vary appreciably in a field system, simulation with SUTRA, which is by definition a single phase flow and transport model, must be critically evaluated against the possible necessity of employing a multiphase fluid flow and transport model.

Fluid mass balance

The so-called "flow simulation" provided by SUTRA is in actuality a calculation of how the amount of fluid mass contained within the void spaces of the solid matrix changes with time. In a particular volume of solid matrix and void space, the total fluid mass $(\varepsilon S_w \rho)\cdot VOL$, may change with time due to ambient ground-water inflows or outflows; injection or withdrawal wells; changes in fluid density caused by changing temperature or concentration; or changes in saturation. SUTRA flow simulation is, in fact, a fluid mass balance which keeps track of the fluid mass contained at every point in the simulated ground-water system as it changes with time due to flows, wells, and saturation or density changes.

The fluid mass balance is expressed as the sum of pure water and pure solute mass balances for a solid matrix in which there is negligible net movement (i.e., no subsidence and no compaction):

$$\frac{\partial(\varepsilon S_w \rho)}{\partial t} \equiv -\underline{\nabla} \cdot (\varepsilon S_w \rho \underline{v}) + Q_p + \Upsilon \qquad (2.22)$$

where:

$Q_p(x,y[,z],t)$	$[M/(L^3 \cdot s)]$	fluid mass source (including pure water mass plus solute mass dissolved in source water)
$\Upsilon(x,y[,z],t)$	$[M/(L^3 \cdot s)]$	solute mass source (e.g., dissolution of solid matrix or desorption)

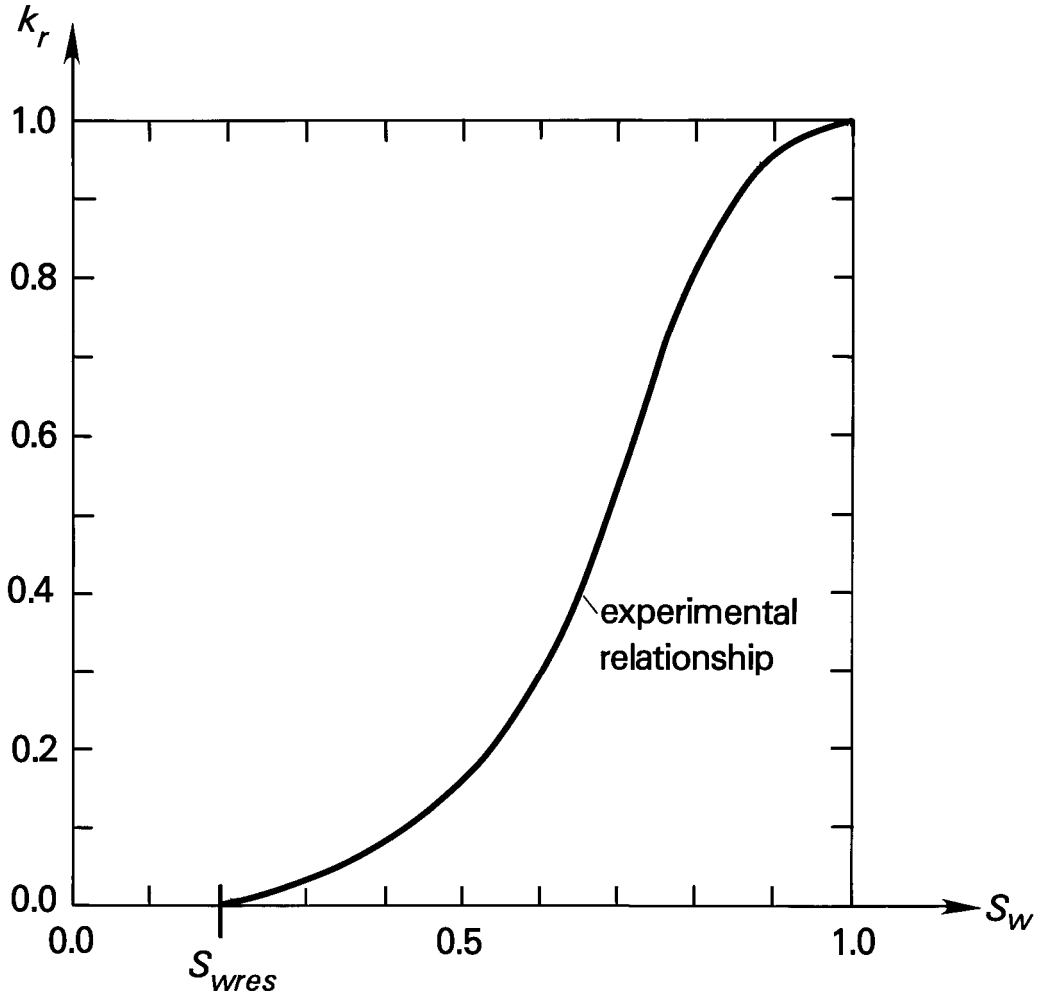


Figure 2.3. Relative permeability-saturation relation (schematic). k_r is the relative permeability.

The term on the left may be recognized as the total change in fluid mass contained in the void space with time. The term involving $\underline{\nabla}$ represents contributions to local fluid mass change due to excess of fluid inflows over outflows at a point. The fluid mass source term, Q_p, accounts for external additions of fluid including pure water mass plus the mass of any solute dissolved in the source fluid. The pure solute mass source term, Υ, may account for external additions of pure solute mass not associated with a fluid source. In most cases, this contribution to the total mass is small compared to the total pure water mass contributed by fluid sources, Q_p. Pure solute sources, Υ, are therefore neglected in the fluid mass balance, but may be readily included in SUTRA for special situations. Note that solute mass sources are not neglected in the solute mass balance, which is discussed in section 2.4.

While (2.22) is the most fundamental form of the fluid mass balance, it is necessary to express each mechanism represented by a term of the equation, in terms of the primary variables, p, C, and T. As SUTRA allows variation in only one of C or T at a time, the letter U is employed to represent either of these quantities. The development from equation (2.9) to (2.18) allows the time derivative in (2.22) to be expanded:

$$\frac{\partial(\varepsilon S_w \rho)}{\partial t} = \left(S_w \rho S_{op} + \varepsilon \rho \frac{\partial S_w}{\partial p} \right) \frac{\partial p}{\partial t} + \left(\varepsilon S_w \frac{\partial \rho}{\partial U} \right) \frac{\partial U}{\partial t} \tag{2.23}$$

While the concepts upon which specific pressure storativity, S_{op}, is based, do not exactly hold for unsaturated media, the error introduced by summing the storativity term with the term involving $(\partial S_w / \partial p)$ is insignificant as $(\partial S_w / \partial p) >>> S_{op}$.

The exact form of the fluid mass balance as implemented in SUTRA is obtained from (2.22) by neglecting Υ, substituting (2.23) and employing Darcy's law, (2.19), for $\underline{v}$:

$$\left(S_w \rho S_{op} + \varepsilon \rho \frac{\partial S_w}{\partial p} \right) \frac{\partial p}{\partial t} + \left(\varepsilon S_w \frac{\partial \rho}{\partial U} \right) \frac{\partial U}{\partial t} - \underline{\nabla} \cdot \left[\left(\frac{\underline{\underline{k}} \, k_r \rho}{\mu} \right) \cdot \left(\underline{\nabla} p - \rho \underline{g} \right) \right] = Q_p \tag{2.24}$$

2.3 Energy Transport in Ground Water

Subsurface energy-transport mechanisms

Energy is transported in the water-solid matrix system by flow of ground water, and by thermal conduction from higher to lower temperatures through both the fluid and solid. The actual flow velocities of the ground water from point to point in the 3D space of an aquifer may vary considerably about an average velocity, $\underline{v}(x, y[, z], t)$, calculated from Darcy's law (2.19a). As the true, not average, velocity field is usually too complex to measure in real systems, an additional transport mechanism approximating the effects of mixing of different temperature ground waters moving both faster and slower than the average velocity, $\underline{v}$, is hypothesized. This mechanism, called energy dispersion, is employed in SUTRA as the best currently available, though approximate, description of the mixing process. In the simple dispersion model employed, dispersion, in effect, adds to the thermal conductivity value of the fluid-solid medium in particular directions dependent upon the direction of fluid flow. In other words, mixing due to

the existence of nonuniform, non-average velocities in three dimensions about the average flow, $\underline{v}$, is conceptualized as a diffusion-like process with anisotropic diffusivities.

The model has been shown, in fact, to well describe transport in purely homogeneous porous media with uniform one-dimensional flows. In heterogeneous field situations with nonuniform flow in, for example, irregular bedding or fractures, the model holds only at the predetermined scale at which dispersivities have been determined and it must currently be considered as a necessary approximation, and be very carefully applied when extrapolating to other scales of transport.

Solid matrix-fluid energy balance

The simulation of energy transport provided by SUTRA is actually a calculation of the time rate of change of the amount of energy stored in the solid matrix and fluid. In any particular volume of solid matrix plus fluid, the amount of energy contained is $[\varepsilon S_w \rho e_w + (1-\varepsilon)\rho_s e_s]\cdot VOL$, where:

e_w	[E/M]	energy per unit mass water
e_s	$[E/M_G]$	energy per unit mass solid matrix
ρ_s	$[M_G/L_G^3]$	density of solid grain in solid matrix

and where [E] are energy units $[M\cdot L^2/s^2]$.

The stored energy in a volume may change with time due to: ambient water with a different temperature flowing in, well water of a different temperature injected, changes in the total mass of water in the block, thermal conduction (energy diffusion) into or out of the volume, energy dispersion in or out, and energy production or loss due to nuclear, chemical or biological reactions.

This balance of changes in stored energy with various energy fluxes is expressed as follows:

$$\frac{\partial\left[\varepsilon S_w \rho e_w + (1-\varepsilon)\rho_s e_s\right]}{\partial t} = -\underline{\nabla}\cdot\left(\varepsilon S_w \rho e_w \underline{v}\right) + \underline{\nabla}\cdot\left[\underline{\underline{\lambda}}\cdot\underline{\nabla}T\right] + \underline{\nabla}\cdot\left[\varepsilon S_w \rho c_w \underline{\underline{D}}\cdot\underline{\nabla}T\right] + Q_p c_w T^* + \varepsilon S_w \rho \gamma_o^w + (1-\varepsilon)\rho_s \gamma_o^s \quad (2.25)$$

$\lambda(x,y[,z],t)$	$[E/(s\cdot L\cdot °C)]$	bulk thermal conductivity of solid matrix plus fluid
$\underline{\underline{I}}$	[1]	identity tensor (ones on diagonal, zeroes elsewhere) (in 2D, a 2 x 2 matrix of values; in 3D, a 3 x 3 matrix of values)
c_w	$[E/(M\cdot °C)]$	specific heat of water (c_w~4.182 x 10^3[J/(kg·°C)] at 20°C)
$\underline{\underline{D}}(x,y[,z],t)$	$[L^2/s]$	dispersion tensor (in 2D, a 2 x 2 matrix of values; in 3D, a 3 x 3 matrix of values)

$T^*(x,y[,z],t)$	[°C]	temperature of source fluid
$\gamma_o^w(x,y[,z],t)$	[E/(M·s)]	energy source in fluid
$\gamma_o^s(x,y[,z],t)$	[E/(M_G·s)]	energy source in solid grains

The time derivative expresses the total change in energy stored in both the solid matrix and fluid per unit total volume. The term involving $\underline{v}$ expresses contributions to locally stored energy from average-uniform flowing fluid (average energy advection). The term involving bulk thermal conductivity, λ, expresses heat conduction contributions to local stored energy and the term involving the dispersivity tensor, $\underline{\underline{D}}$, approximately expresses the contribution of irregular flows and mixing, which are not accounted for by average energy advection. The term involving Q_p accounts for the energy added by a fluid source with temperature, T^*. The last terms account for energy production in the fluid and solid, respectively, due to endothermic reactions, for example.

While models that are more complex are available and may be implemented if desired, SUTRA employs a volumetric average approximation for bulk thermal conductivity, λ:

$$\lambda \equiv \varepsilon S_w \lambda_w + (1-\varepsilon)\lambda_s \tag{2.26}$$

λ_w	[E/(s·L·°C)]	fluid thermal conductivity ($\lambda_w \sim 0.6$ [J/(s·m·°C)] at 20°C)
λ_s	[E/(s·L·°C)]	solid thermal conductivity ($\lambda_w \sim 3.5$ [J/(s·m·°C)] at 20°C, for sandstone)

The specific energy content (per unit mass) of the fluid and the solid matrix depends on temperature as follows:

$$e_w = c_w T \tag{2.27a}$$

$$e_s = c_s T \tag{2.27b}$$

c_s	[E/(M_G·°C)]	solid grain specific heat ($c_s \sim 8.4 \times 10^2$[J/(kg·°C)] for sandstone at 20°C)

An expanded form of the solid matrix-fluid energy balance is obtained by substitution of (2.27a, b) and (2.26) into (2.25). This yields:

$$\frac{\partial}{\partial t}\left[\varepsilon S_w \rho c_w + (1-\varepsilon)\rho_s c_s\right] T + \underline{\nabla}\cdot\left(\varepsilon S_w \rho c_w \underline{v} T\right) - \underline{\nabla}\cdot\left\{\left[S_w\lambda_w + (1-\varepsilon)\lambda_s\right]\underline{\underline{I}} + \varepsilon S_w \rho c_w \underline{\underline{D}}\right\}\cdot\underline{\nabla} T = Q_p c_w T^* + \varepsilon S_w \rho \gamma_o^w + (1-\varepsilon)\rho_s \gamma_o^s \tag{2.28}$$

2.4 Solute Transport in Ground Water

Subsurface solute-transport mechanisms

Solute mass is transported through the porous medium by flow of ground water (solute advection) and by molecular or ionic diffusion, which while small on a field scale, carries solute mass from areas of high to low concentrations. The actual flow velocities of the ground water from point to point in 3D space of an aquifer may vary considerably about an average velocity, $\underline{v}$, which is calculated from Darcy's law (2.19a). As the true, not-average, velocity field is usually too complex to measure in real systems, an additional transport mechanism approximating the effects of mixing of waters with different concentrations moving both faster and slower than the average velocity, $\underline{v}(x,y[,z],t)$, is hypothesized. This mechanism, called solute dispersion, is employed in SUTRA as the best currently available, though approximate, description of the mixing process. In the simple dispersion model employed, dispersion, in effect, significantly adds to the molecular diffusivity value of the fluid in particular directions dependent upon the direction of fluid flow. In other words, mixing due to the existence of nonuniform, non-average velocities in three dimensions about the average flow, $\underline{v}$, is conceptualized as a diffusion-like process with anisotropic diffusivities.

The model has been shown, in fact, to describe transport well in purely homogeneous porous media with uniform one-dimensional flows. In heterogeneous field situations with nonuniform flows in, for example, irregular bedding or fractures, the model holds only at the predetermined scale at which dispersivities have been determined and it must be considered as a currently necessary approximation, and be very carefully applied when extrapolating to other scales of transport.

Solute and adsorbate mass balances

SUTRA solute-transport simulation accounts for a single species mass stored in fluid solution as well as solute and species mass stored as adsorbate on the surfaces of solid matrix grains. Solute concentration, C, and adsorbate concentration, $C_s(x,y[,z],t)$ $[M_s/M_G]$, (where $[M_s]$ denotes units of solute mass, and $[M_G]$ denotes units of solid grain mass), are related through equilibrium adsorption isotherms. The species mass stored in solution in a particular volume of solid matrix may change with time due to: ambient water with a different concentration flowing in, well water injected with a different concentration, changes in the total fluid mass in the block, solute diffusion or dispersion in or out of the volume, transfer of dissolved species to adsorbed species (or reverse), or a chemical or biological reaction causing solute production or decay. The species mass stored as adsorbate on the surface of solid grains in a particular block of solid matrix may change with time due to a gain of adsorbed species by transfer of solute from the fluid (or reverse), or a chemical or biological reaction causing adsorbate production or decay.

The separate balances for a single species stored in solution (solute) and on the solid grains (adsorbate), are expressed, respectively, as follows:

$$\frac{\partial(\varepsilon S_w \rho C)}{\partial t} = -f - \underline{\nabla} \cdot (\varepsilon S_w \rho \underline{v} C) + \underline{\nabla} \cdot \left[\varepsilon S_w \rho \left(D_m \underline{\underline{I}} + \underline{\underline{D}}\right) \cdot \underline{\nabla} C\right] + \varepsilon S_w \rho \Gamma_w + Q_p C^* \qquad (2.29)$$

$$\frac{\partial[(1-\varepsilon)\rho_s C_s]}{\partial t} = +f + (1-\varepsilon)\rho_s \Gamma_s \qquad (2.30)$$

f(x,y[,z],t)	$[M_S/(L^3 \cdot s)]$	volumetric adsorbate source (gain of absorbed species by transfer from fluid per unit total volume)
D_m	$[L^2/s]$	apparent molecular diffusivity of solute in solution in a porous medium including tortuosity effects, ($D_m \sim 1. \times 10^{-9}$ $[m^2/s]$ for NaCl at 20.°C)
$\underline{\underline{I}}$	[1]	identity tensor (ones on diagonal, zero elsewhere) (in 2D, a 2 x 2 matrix of values; in 3D, a 3 x 3 matrix of values)
$\underline{\underline{D}}$ (x,y[,z],t)	$[L^2/s]$	dispersion tensor (in 2D, a 2 x 2 matrix of values; in 3D, a 3 x 3 matrix of values)
Γ_w(x,y[,z],t)	$[M_s/M \cdot s]$	solute mass source in fluid (per unit fluid mass) due to production reactions
C^*(x,y[,z],t)	$[M_s/M]$	solute concentration of fluid sources (mass fraction)
C_s(x,y[,z],t)	$[M_s/M_G]$	specific concentration of adsorbate on solid grains (mass adsorbate/(mass solid grains plus adsorbate))
ρ_s	$[M_G/L_G^3]$	density of solid grains in solid matrix
Γ_s(x,y[,z],t)	$[M_s/M_G \cdot s]$	adsorbate mass source (per unit solid matrix mass) due to production reactions within adsorbed material itself.

where $[L_G^3]$ is the volume of solid grains.

Equation (2.29) is the solute mass balance in terms of the dissolved mass fraction (solute concentration), C. The time derivative expresses the total changes in solute mass with time in a volume due to the mechanisms represented by terms on the right side of the equation. The term involving f(x,y[,z],t) represents the loss of solute mass from solution which becomes fixed on the

solid grain surfaces as adsorbate. The adsorbate source, f, may, in general, depend on solute concentration, C, adsorbate concentration, C_s, and the rate of change of these concentrations, depending on either an equilibrium adsorption isotherm or on nonequilibrium adsorption processes. SUTRA algorithms are structured to directly accept nonequilibrium sorption models as an addition to the code. However, the current version of SUTRA assumes equilibrium sorption as shown in the following section, "Adsorption and production/decay processes."

The term involving fluid velocity, $\underline{v}$, represents average advection of solute mass into or out of the local volume. The term involving molecular diffusivity of solute, D_m, and dispersivity, $\underline{\underline{D}}$, expresses the contribution of solute diffusion and dispersion to the local changes in solute mass. The diffusion contribution is based on a true physical process often negligible at the field scale. The dispersion contribution approximates the effect of solute advection and mixing in irregular flows, which are not accounted for by solute advected by the average velocity. The solute mass source term involving $\Gamma_w(x,y[,z],t)$, the solute mass production rate per unit mass of fluid, expresses the contribution to dissolved species mass of chemical, biological or radioactive reactions in the fluid. The last term accounts for dissolved species mass added by a fluid source with concentration, C^*.

Equation (2.30) is the balance of mass, which has been adsorbed by solid grain surfaces in terms of species concentration on the solid (specific adsorbate concentration), C_s. The change in total adsorbate mass is expressed by the time derivative term. It may increase due to species leaving solution as expressed by adsorbate source term, f. The adsorbed mass may also change due to a production of adsorbate mass (per unit solid matrix mass), Γ_s by radioactive or chemical processes within the adsorbate. Note that mass becomes immobile once adsorbed, and is affected only by possible desorption or chemical and biological processes.

The total mass of a species in a volume is given by the sum of solute mass and adsorbate mass. A balance of the total mass of a species is obtained by addition of (2.30) and (2.29). The general form of the total species mass balance used in SUTRA is this:

$$\frac{\partial(\varepsilon S_w \rho C)}{\partial t} + \frac{\partial[(1-\varepsilon)\rho_s C_s]}{\partial t} = -\underline{\nabla}\cdot(\varepsilon S_w \rho \underline{v} C) + \underline{\nabla}\cdot\left[\varepsilon S_w \rho\left(D_m \underline{\underline{I}} + \underline{\underline{D}}\right)\cdot\underline{\nabla} C\right] + \varepsilon S_w \rho \Gamma_w + (1-\varepsilon)\rho_s \Gamma_s + Q_p C^* \quad (2.31)$$

Equation (2.31) is the basis for SUTRA solute-transport simulation. In cases of solute transport where adsorption does not occur ($C_s = 0$), the adsorbate source term, f, simply has the value zero ($f = 0$), and the terms that stem from equation (2.30) are ignored. Further discussion of solute and adsorbate mass balances may be found in Bear (1979).

Adsorption and production/decay processes

The volumetric adsorbate source, f, of (2.29) and (2.30) may be expressed in terms of a specific sorption rate, f_s, as:

$$f = (1-\varepsilon)\rho_s f_s \quad (2.32a)$$

$f_s(x,y[,z],t)$	$[M_S/M_G \cdot s]$	specific solute mass adsorption rate (per unit mass solid matrix)

A particular nonequilibrium (kinetic) model of sorption is obtained by defining the functional dependence of the sorption rate, f_s, on other parameters of the system. For example, for a linear reversible nonequilibrium sorption model, the expression is: $f_s = m_1(C - m_2C_s)$, where m_1 and m_2 are sorption parameters. This particular model and a number of other nonequilibrium sorption models are accommodated by a general expression for f_s, as follows:

$$f_s \equiv \kappa_1 \frac{\partial C}{\partial t} + \kappa_2 C + \kappa_3 \tag{2.32b}$$

where: $\kappa_1 = \kappa_1(C,C_s)$, $\kappa_2 = \kappa_2(C,C_s)$, $\kappa_3 = \kappa_3(C,C_s)$.

$\kappa_1(C,C_s)$	$[M/M_G]$	first general sorption coefficient
$\kappa_2(C,C_s)$	$[M/M_G \cdot s]$	second general sorption coefficient
$\kappa_3(C,C_s)$	$[M_S/M_G \cdot s]$	third general sorption coefficient

Through a suitable definition of the general coefficients, $\kappa_i(C,C_s)$, a number of nonequilibrium sorption models may be obtained. For example, the linear reversible nonequilibrium model mentioned above requires the definitions: $\kappa_1 \equiv 0$, $\kappa_2 \equiv m_1$, and $\kappa_3 \equiv -m_1m_2C_s$. The general coefficients κ_1, κ_2, and κ_3 are included in the SUTRA code (as CS1, CS2 and CS3, in Subroutine ADSORB) to provide generality for possible inclusion of such nonequilibrium (kinetic) sorption models.

The equilibrium sorption models are based on definition of the general coefficients through the following relation:

$$\frac{\partial C_s}{\partial t} = \kappa_1 \frac{\partial C}{\partial t} \tag{2.33}$$

Only general sorption coefficient κ_1 need be defined based on various equilibrium sorption isotherms as shown in the following. The other coefficients are set to zero: $\kappa_2 = \kappa_3 = 0$.

The linear equilibrium sorption model is based on the linear sorption isotherm assuming constant fluid density:

$$C_s = (\chi_1 \rho_o) C \tag{2.34a}$$

$$\frac{\partial C_s}{\partial t} = (\chi_1 \rho_o) \frac{\partial C}{\partial t} \tag{2.34b}$$

where:

χ_1	$[L_f^3/M_G]$	linear distribution coefficient

and ρ_o is the fluid base density. For linear sorption, general coefficient κ_1 takes on the definition:

$$\kappa_1 = \chi_1 \rho_o \tag{2.34c}$$

The Freundlich equilibrium sorption model is based on the following isotherm, which assumes a constant fluid density, ρ_o:

$$C_s = \chi_1 (\rho_o C)^{\left(\frac{1}{\chi_2}\right)} \tag{2.35a}$$

$$\frac{\partial C_s}{\partial t} = \left(\frac{\chi_1}{\chi_2}\right)(\rho_o C)^{\left(\frac{1-\chi_2}{\chi_2}\right)} \rho_o \frac{\partial C}{\partial t} \tag{2.35b}$$

where:

χ_1	$[L_f^3/M_G]$	a Freundlich distribution coefficient
χ_2	[1]	Freundlich coefficient

(When $\chi_2 = 1$, the Freundlich isotherm is equivalent to the linear isotherm.) For Freundlich sorption, then, the general coefficient κ_1 takes the definition:

$$\kappa_1 = \left(\frac{\chi_1}{\chi_2}\right) \rho_o^{\left(\frac{1}{\chi_2}\right)} C^{\left(\frac{1-\chi_2}{\chi_2}\right)} \tag{2.35c}$$

The Langmuir equilibrium sorption model is based on the following isotherm, which assumes a constant fluid density, ρ_o:

$$C_s = \frac{\chi_1 (\rho_o C)}{1 + \chi_2 (\rho_o C)} \tag{2.36a}$$

$$\frac{\partial C_s}{\partial t} = \frac{\chi_1 \rho_o}{(1 + \chi_2 \rho_o C)^2} \frac{\partial C}{\partial t} \tag{2.36b}$$

where:

χ_1	$[L_f^3/M_G]$	a Langmuir distribution coefficient
χ_2	$[L_f^3/M_S]$	Langmuir coefficient

For very low solute concentrations, C, Langmuir sorption becomes linear sorption with linear distribution coefficient χ_1. For very high solute concentrations, C, the concentration of adsorbate mass, C_S, approaches an upper limit equal to (χ_1/χ_2). The general SUTRA coefficient κ_1 is defined for Langmuir sorption as:

$$\kappa_1 = \frac{\chi_1 \rho_o}{(1 + \chi_2 \rho_o C)^2} \tag{2.36c}$$

The production terms for solute, Γ_w, and adsorbate, Γ_s, allow for first-order mass production (or decay) such as linear BOD (biochemical oxygen demand) or radioactive decay, biological or chemical production, and zero-order mass production (or decay).

$$\Gamma_w = \gamma_1^w C + \gamma_o^w \tag{2.37a}$$

$$\Gamma_s = \gamma_1^s C_s + \gamma_o^s \tag{2.37b}$$

where:

γ_1^w	$[s^{-1}]$	first order mass production rate of solute
γ_o^w	$[(M_s/M)/s]$	zero-order solute mass production rate
γ_1^s	$[s^{-1}]$	first-order mass production rate of adsorbate
γ_o^s	$[(M_s/M_G)/s]$	zero-order adsorbate mass production rate

2.5 Dispersion

Pseudotransport mechanism

Dispersion is a pseudo-transport process representing mixing of fluids that actually travel through the solid matrix at velocities different from the average velocity in two or three spatial dimensions, $\underline{v}$, calculated from Darcy's law (2.19). Dispersion is a pseudoflux in that it only represents deviations from an average advective flux of energy or solute mass and as such does not represent a true mechanism of transport. Should it be possible to represent the true, complex, nonhomogeneous velocity field in, for example, the layers of an irregularly bedded field system, then the dispersion process need not be invoked to describe the transport, as the local variations in advection would provide the true picture of the transport taking place. However, as available data almost never allow for such a detailed velocity description, an approximate description must be employed, which helps to account for observed temperatures or concentrations different from that expected based on the average fluid advection.

Research trends have been to develop dispersion models for various hydrogeological conditions, and SUTRA may be updated to include new results as they become available. Currently (2007), SUTRA dispersion is based on a generalization for anisotropic media of the standard description for dispersion in isotropic, homogeneous porous media. Because any inconsistencies that may arise in applying this dispersion model to a particular field situation often would not be apparent due to the poor quality or small amount of measured data, the user is warned to exercise good judgment in interpreting results when large amounts of so-called dispersion are required to explain the field measurements. In any case, the user is advised to consult up-to-date literature on field-scale dispersion before employing this transport model.

Isotropic-media dispersion model

The dispersion tensor, $\underline{\underline{D}}$, which appears in both the energy (2.28) and solute (2.31) balances, is usually expressed for flow in systems with isotropic permeability and isotropic spatial distribution of inhomogeneities in aquifer materials in 2D as

$$\underline{\underline{D}} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \tag{2.38a}$$

and in 3D as

$$\underline{\underline{D}} = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{yx} & D_{yy} & D_{yz} \\ D_{zx} & D_{zy} & D_{zz} \end{bmatrix} \tag{2.38b}$$

where, in both, $\underline{\underline{D}}$ is symmetric (i.e., $D_{xy}=D_{yx}$, $D_{xz}=D_{zx}$, and $D_{zy}=D_{yz}$).
In 2D, the diagonal elements are

$$D_{xx} = \left(\frac{1}{v^2}\right)\left(d_L v_x^2 + d_T v_y^2\right) \tag{2.39a}$$

$$D_{yy} = \left(\frac{1}{v^2}\right)\left(d_T v_x^2 + d_L v_y^2\right) \tag{2.39b}$$

and the off-diagonal elements are

$$D_{ij} = \left(\frac{1}{v^2}\right)\left(d_L - d_T\right)\left(v_i v_j\right) \tag{2.39c}$$

$i \neq j$, i=x,y
j=x,y

In 3D, the diagonal elements are

$$D_{xx} = \left(\frac{1}{v^2}\right)\left(d_L v_x^2 + d_T v_y^2 + d_T v_z^2\right) \tag{2.39d}$$

$$D_{yy} = \left(\frac{1}{v^2}\right)\left(d_T v_x^2 + d_L v_y^2 + d_T v_z^2\right) \tag{2.39e}$$

$$D_{zz} = \left(\frac{1}{v^2}\right)\left(d_T v_x^2 + d_T v_y^2 + d_L v_z^2\right) \tag{2.39f}$$

and the off-diagonal elements are

$$D_{ij} = \left(\frac{1}{v^2}\right)\left(d_L - d_T\right)\left(v_i v_j\right) \tag{2.39g}$$

$i \neq j$, i=x,y,z
j=x,y,z

The variables that determine the elements of the dispersion tensor are defined as follows:

v(x,y[,z],t)	[L/s]	magnitude of velocity $\underline{v}$
v_X(x,y[,z],t)	[L/s]	magnitude of x-component $\underline{v}$
v_Y(x,y[,z],t)	[L/s]	magnitude of y-component $\underline{v}$
v_Z(x,y,z,t)	[L/s]	magnitude of z-component $\underline{v}$
d_L(x,y[,z],t)	[L^2/s]	longitudinal dispersion coefficient
d_T(x,y[,z],t)	[L^2/s]	transverse dispersion coefficient

The longitudinal and transverse dispersion coefficients, d_L and d_T [L^2/s], are analogous to typical diffusion coefficients. What is special is that these are directional in nature. The quantity d_L acts as a diffusion coefficient that causes dispersion forward and backward along the local direction of fluid flow. The quantity d_T acts as a diffusion coefficient that causes dispersion symmetrically in the directions perpendicular to the local flow direction, and is called the transverse dispersion coefficient. Thus, in 3D, if d_L and d_T were of equal value, a spherical mass of tracer released in ground water flowing, on the average, uniformly and unidirectionally would disperse in a perfectly symmetric spherical manner as it moved downstream. Similarly, in 2D, a circular disk of tracer would disperse in a perfectly symmetric circular manner as it moved downstream. However, if $d_L > d_T$ then the tracer would disperse in an ellipsoidal (3D) or elliptical (2D) manner, with the long axis oriented in the flow direction as it moved downstream.

The sizes of the dispersion coefficients in this model for dispersion in isotropic permeability systems are dependent upon the absolute local magnitude of average velocity in a flowing system (Bear, 1979):

$$d_L = \alpha_L v \qquad (2.40a)$$

$$d_T = \alpha_T v \qquad (2.40b)$$

$\alpha_L(x,y[,z])$	[L]	longitudinal dispersivity of solid matrix
$\alpha_T(x,y[,z])$	[L]	transverse dispersivity of solid matrix

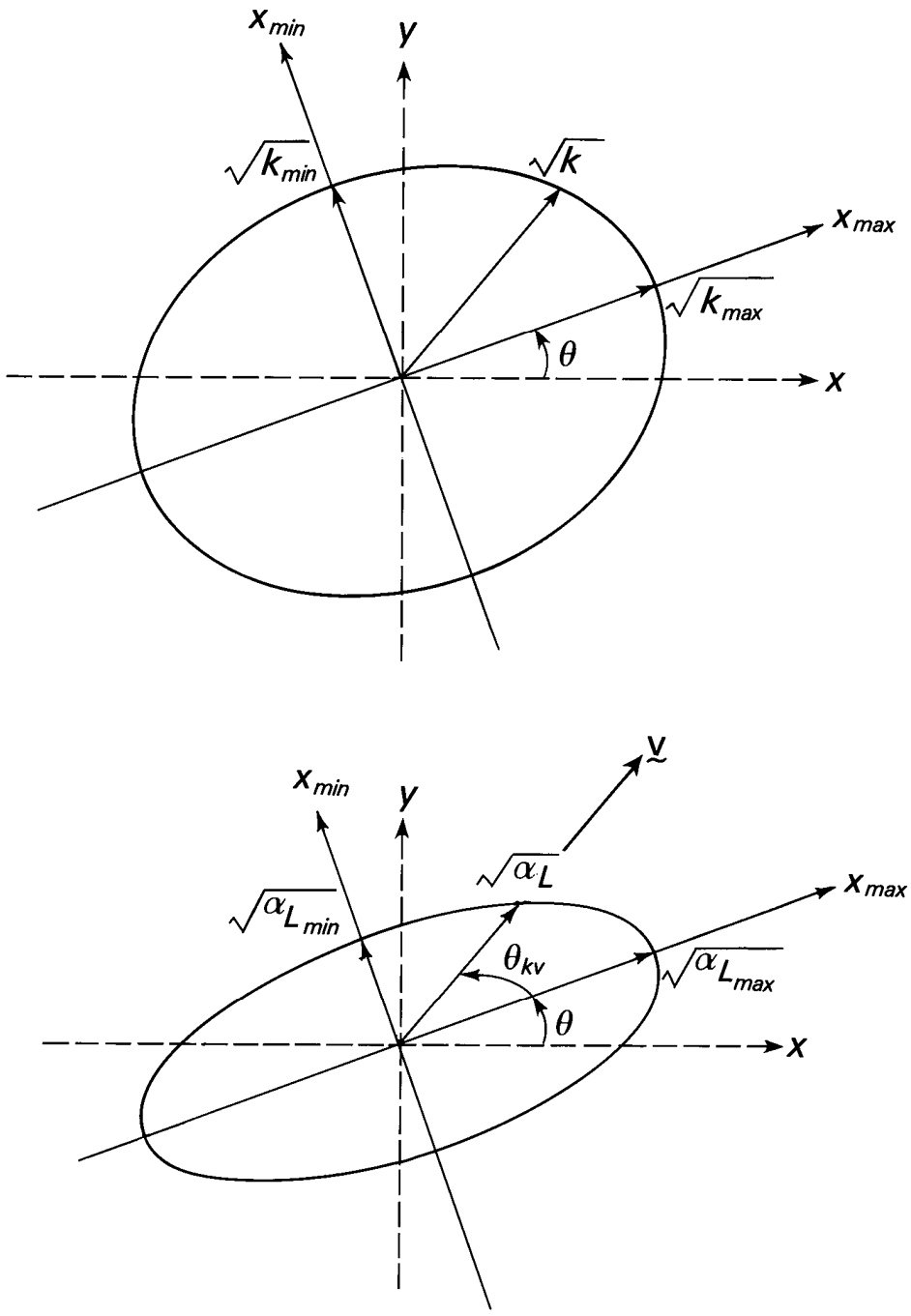
When the isotropic-media dispersion model is applied to a particular field situation where aquifer inhomogeneities are much smaller than the field transport scale, then dispersivities α_L and α_T may be considered to be fundamental transport properties of the system just as, for example, permeability is a fundamental property for flow through porous media. In cases where inhomogeneities are large or scales of transport vary, dispersivities are not a fundamental system property. In this case, simulated dispersion effects must be interpreted with care, because dispersivity values are the only means available to represent the dispersive characteristics of a given system to be simulated when using the SUTRA code.

Anisotropic-media dispersion model – overview

In a system with anisotropic spatial distribution of heterogeneities in aquifer materials, the same amount of dispersion may not occur for flow in all directions, even when the magnitude of flow velocity, v, is the same. For example, in a layered aquifer, the amount of dispersion would not necessarily be the same for flow parallel to the layers and flow perpendicular to the layers.

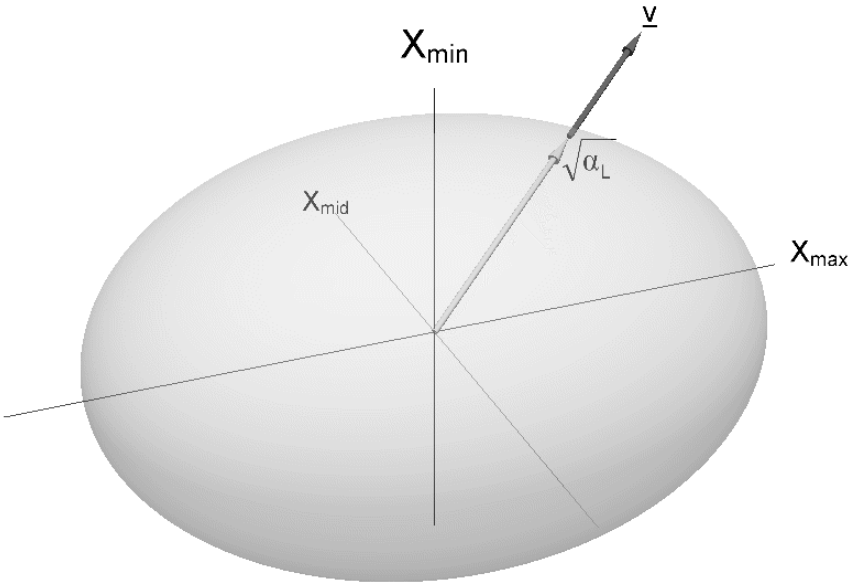
The isotropic-media dispersion model, described in the previous section, does not account for this possibility, and its basic parameters, α_L and α_T, are independent of flow direction. A model for anisotropic media is hypothesized that provides key aspects of expected dispersion behavior. Although the hypothesized model described below is neither completely general nor rigorous, it provides a practical means of describing dispersion in anisotropic media.

Dispersive spreading is assumed to occur along a set of mutually perpendicular directions (one of which is the flow direction) in a symmetric manner (i.e., equal amounts in opposite directions along each direction), just as in the isotropic case. Following the discussion in the previous section referring to equations (2.40a) and (2.40b), but in a 2D anisotropic medium, a circular disk of tracer would become elliptical, with the longest axis in the flow direction, as it moved downstream, if $\alpha_L > \alpha_T$. However, it may not spread the same way if it moved with the same speed, v, in a different direction. For example, if $\alpha_L = \alpha_T$ (equal amounts of longitudinal and transverse dispersion) for one flow direction, but $\alpha_L > \alpha_T$ (greater longitudinal than transverse dispersion) for all other flow directions, then the tracer would spread in a circular manner if the flow occurred in the direction where $\alpha_L = \alpha_T$, but would spread as an ellipse for flow in any other direction. In such a case, the values of both α_L and α_T would depend on the direction of flow. Such a medium is considered anisotropic with respect to dispersion.

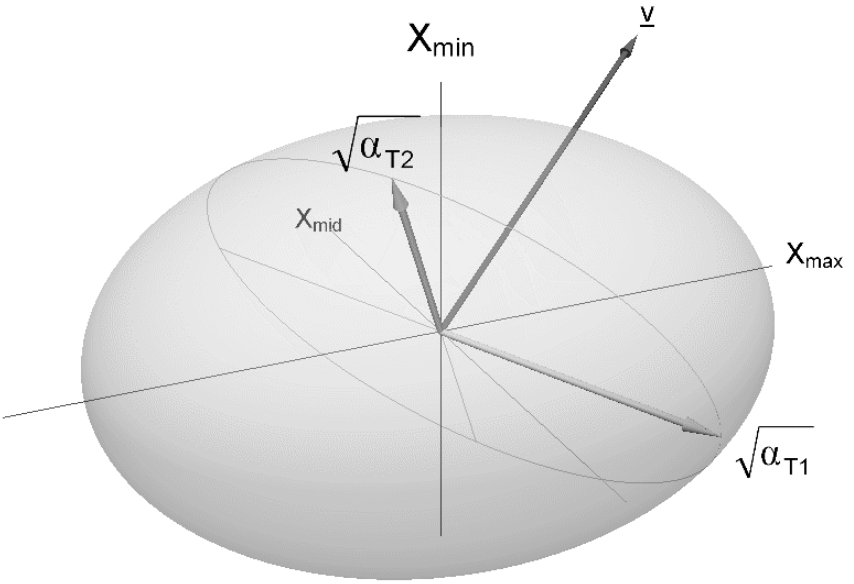


(a)

Figure 2.4a. Definition of flow-direction-dependent longitudinal dispersivity, $\alpha_L(\theta_{kv})$, in 2D. The upper figure is described in Figure 2.2a. In the lower figure, α_{Lmax} and α_{Lmin} are the longitudinal dispersivity for flow in the maximum and minimum directions, respectively. θ_{kv} is the angle from the maximum permeability direction to the flow direction and α_L is the longitudinal dispersivity in that flow direction.



(b)



(c)

Figure 2.4b,c. Definition of flow-direction-dependent (b) longitudinal dispersivity, $\alpha_L(\theta_{kv1}, \theta_{kv2})$, and (c) transverse dispersivities, $\alpha_{T1}(\theta_{kv1}, \theta_{kv2})$ and $\alpha_{T2}(\theta_{kv1}, \theta_{kv2})$, in 3D. (For the sake of visual clarity, the (x,y,z)-coordinate axes are not shown, and the angles θ_{kv1} and θ_{kv2} are not labelled.) As in Figure 2.2b, x_{max}, x_{mid}, and x_{min} are the principal directions of permeability. In the lower figure, the slicing ellipse that defines the transverse dispersivities lies in the plane that is perpendicular to the flow direction, $\underline{v}$, and that passes through the center of the ellipsoid. The two transverse dispersivities and their directions are given by the radii along the major and minor axes of the slicing ellipse.

In a 3D anisotropic medium, different amounts of transverse dispersion may occur in two mutually perpendicular directions (the transverse directions), which are both perpendicular to the flow direction. If the transverse dispersivities for each of these tranverse directions, α_{T1} and α_{T2}, are not equal to each other or to α_L, then a sphere of tracer would spread in an ellipsoidal shape with three unequal axes as it moved downstream. To further complicate the process in 3D, although the transverse directions are both perpendicular to the flow direction and to each other, their direction may be rotated at any angle about the flow direction. Their direction is here assumed to be a function of the flow direction and the principal permeability directions of the medium. Thus, the orientation of the ellipse that represents the transverse cross section of the tracer ellipsoid is also a function of the flow direction and principal permeability directions of the medium.

Dispersion in 2D and 3D anisotropic media is described in SUTRA by ad-hoc models that allow dispersivity values to change as a function of flow direction relative to the principal permeability directions of the medium. The model in 2D is based on selecting dispersivities, α_L and α_T, as functions of flow direction from two 'dispersivity ellipses'. The 3D model is based on selection of α_L from one 'dispersivity ellipsoid', and the two transverse dispersivity values, α_{T1} and α_{T2}, and their associated directions from another 'dispersivity ellipsoid', all as a function of flow direction. These models are described below.

Anisotropic-media dispersion model – details

In an anisotropic medium, the 2D and 3D dispersion tensors take the form (2.38a,b), and they are assumed symmetric, as for an isotropic medium. In 2D, the tensor has two mutually perpendicular principal directions of dispersion, about which dispersion is symmetric; an initially circular mass of tracer will disperse in an elliptical manner, with the axes of the ellipse falling along the principal dispersion directions. In 3D, the tensor has three mutually perpendicular principal directions of dispersion, about which dispersion is symmetric; an initially spherical mass of tracer will disperse in an ellipsoidal manner, with the axes of the ellipsoid falling along the principal dispersion directions. As a rule, the principal dispersion directions are not the same as the principal directions of permeability; the former directions can depend on the flow direction, whereas the latter do not.

The principal directions of the 2D dispersion tensor are denoted here by the unit vectors $\underline{V}$ and $\underline{U}$. In 3D, a third direction is added and is denoted by the unit vector $\underline{W}$. The dispersion coefficients that correspond to $\underline{V}$, $\underline{U}$, and $\underline{W}$ are d_L, d_{T1} (called simply d_T in 2D), and d_{T2}, respectively. The diagonal elements of the 2D dispersion tensor are

$$D_{xx} = d_L V_x^2 + d_T U_x^2 \tag{2.41a}$$

$$D_{yy} = d_L V_y^2 + d_T U_y^2 \tag{2.41b}$$

and the off-diagonal elements are

$$D_{ij} = d_L V_i V_j + d_T U_i U_j \qquad (2.41c)$$
$i \neq j$, i=x,y
j=x,y

In 3D, the diagonal elements are

$$D_{xx} = d_L V_x^2 + d_{T1} U_x^2 + d_{T2} W_x^2 \qquad (2.41d)$$

$$D_{yy} = d_L V_y^2 + d_{T1} U_y^2 + d_{T2} W_y^2 \qquad (2.41e)$$

$$D_{zz} = d_L V_z^2 + d_{T1} U_z^2 + d_{T2} W_z^2 \qquad (2.41f)$$

and the off-diagonal elements are

$$D_{ij} = d_L V_i V_j + d_{T1} U_i U_j + d_{T2} W_i W_j \qquad (2.41g)$$
$i \neq j$, i=x,y,z
j=x,y,z

To complete the dispersion model, it must be stated how the principal directions of dispersion, $\underline{V}$, $\underline{U}$, [and $\underline{W}$], and their corresponding dispersion coefficients, d_L, d_{T1}, [and d_{T2}], vary with flow direction. For SUTRA, an ad-hoc model is postulated in which one principal direction coincides with the flow direction (as it does for an isotropic medium), that is, $\underline{V}=\underline{v}/v$. Thus, $\underline{V}$ points in the longitudinal direction, and d_L becomes the longitudinal dispersion coefficient. (Note that this is only a simplifying assumption. In general, the principal directions of dispersion are not necessarily parallel to and perpendicular to the flow direction.) In 2D, the vector $\underline{U}$ is transverse to the flow, d_T becomes the transverse dispersion coefficient, and equations (2.41a-c) reduce to the isotropic form (2.39a-c), except that d_L and d_T are now functions of flow direction. In 3D, the vectors $\underline{U}$ and $\underline{W}$ are both transverse to the flow, and d_{T1} and d_{T2} become the two transverse dispersion coefficients. As in the isotropic-media dispersion model, dispersion coefficients are obtained by multiplying dispersivities by the magnitude of the flow velocity:

$$d_L = \alpha_L v \qquad (2.42a)$$

$$d_{T1} = \alpha_{T1} v \qquad [d_T = \alpha_T v \text{ in 2D}] \qquad (2.42b)$$

$$d_{T2} = \alpha_{T2} v \qquad (2.42c)$$

In the 2D SUTRA dispersion model, the longitudinal and transverse dispersivities, α_L and α_T, are determined by the radii of two ellipses whose axes are assumed to be aligned with the maximum and minimum permeability directions. The procedure is illustrated for α_L in Figure 2.4a. The semi-major and semi-minor axes of the ellipse are of length $(\alpha_{Lmax})^{1/2}$ and $(\alpha_{Lmin})^{1/2}$, respectively. The radius of the ellipse along the flow direction, $\underline{V}$, is then $(\alpha_L)^{1/2}$. The transverse dispersivity, α_T, is determined by computing the radius along the transverse direction, $\underline{U}$, of an ellipse with semi-major and semi-minor axes of length $(\alpha_{Tmax})^{1/2}$ and $(\alpha_{Tmin})^{1/2}$, respectively. **This convention for computing α_T was introduced in SUTRA version 2.0 (2D3D.1) (Voss and Provost, 2002); in prior versions (Voss, 1984), α_T was determined by computing the**

radius along the flow direction. This convention is modified in version 2.0 and higher to make the 2D dispersion model consistent with the 3D model discussed below. Note also that the subscripts "max" and "min" refer only to the maximum and minimum permeability directions. These are not intended to imply the relation in magnitude of α_{Lmax} and α_{Lmin} (the longitudinal dispersivities for flow in the max and min permeability directions, respectively), nor of α_{Tmax} and α_{Tmin} (the transverse dispersivities for flow in the max and min permeability directions, respectively).

Let $(\hat{x},\hat{y})$ represent 2D coordinates aligned with the max and min permeability directions, that is, $(\hat{x},\hat{y})=(x_{max}, x_{min})$. If $\hat{\underline{V}}$ and $\hat{\underline{U}}$ represent the vectors $\underline{V}$ and $\underline{U}$, respectively, expressed in $(\hat{x},\hat{y})$-coordinates, then in 2D the longitudinal dispersivity is given by

$$\frac{1}{\alpha_L} = \frac{\hat{V}_{\hat{x}}^2}{\alpha_{Lmax}} + \frac{\hat{V}_{\hat{y}}^2}{\alpha_{Lmin}} = \frac{\cos^2\theta_{kv}}{\alpha_{Lmax}} + \frac{\sin^2\theta_{kv}}{\alpha_{Lmin}} \tag{2.43a}$$

or

$$\alpha_L = \frac{\alpha_{Lmax}\alpha_{Lmin}}{\alpha_{Lmin}\cos^2\theta_{kv} + \alpha_{Lmax}\sin^2\theta_{kv}} \tag{2.43b}$$

where

$\alpha_{Lmax}(x,y)$	[L]	squared radius of the longitudinal dispersivity ellipse in the maximum permeability direction, x_{max},
$\alpha_{Lmin}(x,y)$	[L]	squared radius of the longitudinal dispersivity ellipse in the minimum permeability direction, x_{min},
$\theta_{kv}(x,y,t)$	[L]	angle from maximum permeability direction, x_{max}, to the flow direction, $\underline{V}=\underline{v}/v$.

The transverse dispersivity in 2D is given by

$$\frac{1}{\alpha_T} = \frac{\hat{U}_{\hat{x}}^2}{\alpha_{Tmax}} + \frac{\hat{U}_{\hat{y}}^2}{\alpha_{Tmin}} = \frac{\sin^2\theta_{kv}}{\alpha_{Tmax}} + \frac{\cos^2\theta_{kv}}{\alpha_{Tmin}} \tag{2.44a}$$

or

$$\alpha_T = \frac{\alpha_{Tmax}\alpha_{Tmin}}{\alpha_{Tmin}\sin^2\theta_{kv} + \alpha_{Tmax}\cos^2\theta_{kv}} \tag{2.44b}$$

where

$\alpha_{Tmax}(x,y)$	[L]	squared radius of the transverse dispersivity ellipse in the maximum permeability direction, x_{max},
$\alpha_{Tmin}(x,y)$	[L]	squared radius of the transverse dispersivity ellipse in the minimum permeability direction, x_{min}.

This form of longitudinal dispersivity dependence on direction of flow relative to the principal permeability directions is similar to that obtained for a transversely isotropic medium in a stochastic analysis of macrodispersion by Gelhar and Axness (1983). For 2D flow along the principal permeability directions, the longitudinal and transverse dispersivities take the following values:

2D flow direction	longitudinal dispersivity	transverse dispersivity
x_{max}	α_{Lmax}	α_{Tmin}
x_{min}	α_{Lmin}	α_{Tmax}

In the 3D SUTRA dispersion model, which is a generalization of the 2D model, the longitudinal dispersivity, α_L, is determined by the radius of an ellipsoid whose axes are assumed to be aligned with the max, mid, and min permeability directions. The procedure for α_L is illustrated in Figure 2.4b. The transverse dispersivities, α_{T1} and α_{T2}, are determined using a second ellipsoid that is distinct from the one used to compute α_L, but whose axes are also assumed to be aligned with the max, mid, and min permeability directions. The values of α_{T1} and α_{T2} are derived from the principal radii of the ellipse (hereafter called the "slicing ellipse") formed by the intersection of the ellipsoid with the plane that is normal to the flow direction and passes through the center of the ellipsoid. The procedure for α_{T1} and α_{T2} is illustrated in Figure 2.4c. The semi-axes of the transverse dispersivity ellipsoid in the max, mid, and min permeability directions are of length $(\alpha_{Tmax})^{1/2}$, $(\alpha_{Tmid})^{1/2}$, and $(\alpha_{Tmin})^{1/2}$, respectively. The principal radii of the slicing ellipse are then $(\alpha_{T1})^{1/2}$ and $(\alpha_{T2})^{1/2}$. Note also that the subscripts "max," "mid," and "min" refer only to the maximum, middle, and minimum permeability directions. These are not intended to imply the relation in magnitude of α_{Lmax}, α_{Lmid}, and α_{Lmin} (the longitudinal dispersivities for flow in the max, mid, and min permeability directions, respectively), nor of α_{Tmax}, α_{Tmid}, and α_{Tmin} (the transverse dispersivities for flow in the max, mid, and min permeability directions, respectively).

Let $(\hat{x},\hat{y},\hat{z})$ represent 3D coordinates aligned with the max, mid, and min permeability directions, that is, $(\hat{x},\hat{y},\hat{z})=(x_{max}, x_{mid}, x_{min})$. If $\hat{V}$, $\hat{U}$, and $\hat{W}$ represent the vectors V, U, and W, respectively, expressed in $(\hat{x},\hat{y},\hat{z})$-coordinates, then in 3D the longitudinal dispersivity is given by

$$\frac{1}{\alpha_L}=\frac{\hat{V}_{\hat{x}}^2}{\alpha_{Lmax}}+\frac{\hat{V}_{\hat{y}}^2}{\alpha_{Lmid}}+\frac{\hat{V}_{\hat{z}}^2}{\alpha_{Lmin}}=\left(\frac{\cos^2\theta_{kv1}}{\alpha_{Lmax}}+\frac{\sin^2\theta_{kv1}}{\alpha_{Lmid}}\right)\cos^2\theta_{kv2}+\left(\frac{1}{\alpha_{Lmin}}\right)\sin^2\theta_{kv2} \qquad (2.45a)$$

or

$$\alpha_L = \frac{\alpha_{Lmax}\alpha_{Lmid}\alpha_{Lmin}}{\alpha_{Lmin}\left(\alpha_{Lmid}\cos^2\theta_{kv1} + \alpha_{Lmax}\sin^2\theta_{kv1}\right)\cos^2\theta_{kv2} + \alpha_{Lmax}\alpha_{Lmid}\sin^2\theta_{kv2}}, \tag{2.45b}$$

where

$\alpha_{Lmax}(x,y)$	[L]	squared radius of the longitudinal dispersivity ellipsoid in the maximum permeability direction, x_{max},
$\alpha_{Lmid}(x,y)$	[L]	squared radius of the longitudinal dispersivity ellipsoid in the middle permeability direction, x_{mid},
$\alpha_{Lmin}(x,y)$	[L]	squared radius of the longitudinal dispersivity ellipsoid in the minimum permeability direction, x_{min},
$\theta_{kv1}(x,y,t)$	[L]	angle from maximum permeability direction, x_{max}, to the flow direction, $\underline{V}=\underline{v}/v$, measured within the ($x_{max}$, x_{mid})-plane,
$\theta_{kv2}(x,y,t)$	[L]	angle upward from the (x_{max} , x_{mid})-plane to the flow direction, $\underline{V}=\underline{v}/v$.

The transverse dispersivities in 3D are given by

$$\frac{1}{\alpha_{T1}} = \frac{\hat{U}_{\hat{x}}^2}{\alpha_{Tmax}} + \frac{\hat{U}_{\hat{y}}^2}{\alpha_{Tmid}} + \frac{\hat{U}_{\hat{z}}^2}{\alpha_{Tmin}}, \tag{2.46a}$$

$$\frac{1}{\alpha_{T2}} = \frac{\hat{W}_{\hat{x}}^2}{\alpha_{Tmax}} + \frac{\hat{W}_{\hat{y}}^2}{\alpha_{Tmid}} + \frac{\hat{W}_{\hat{z}}^2}{\alpha_{Tmin}}. \tag{2.46b}$$

where

$\alpha_{Tmax}(x,y)$	[L]	squared radius of the transverse dispersivity ellipsoid in the maximum permeability direction, x_{max},
$\alpha_{Tmid}(x,y)$	[L]	squared radius of the transverse dispersivity ellipsoid in the middle permeability direction, x_{mid},
$\alpha_{Tmin}(x,y)$	[L]	squared radius of the transverse dispersivity ellipsoid in the minimum permeability direction, x_{min}.

The geometric construction that determines α_{T1} and α_{T2} and their corresponding eigenvectors $\underline{\hat{U}}$ and $\underline{\hat{W}}$ (namely, the ellipse formed by the intersection of a plane with an ellipsoid) is analogous to that which determines indices of refraction and directions of vibration in the optics of biaxial crystals. Taking advantage of this analogy, SUTRA uses the Biot-Fresnel construction to compute $\underline{\hat{U}}$ and $\underline{\hat{W}}$ and, in turn, α_{T1} and α_{T2}. The procedure is described in texts on optical mineralogy (see, for example, Nesse (1986)) and is not described here except to note that the unit flow vector, $\underline{\hat{V}}$, is analogous to the wave normal; the other two eigenvectors, $\underline{\hat{U}}$ and $\underline{\hat{W}}$, are analogous to the two vibration directions; the flow directions for which $\alpha_{T1}=\alpha_{T2}$ are analogous to the optic axes; $(\alpha_{Tmax})^{1/2}$, $(\alpha_{Tmid})^{1/2}$, and $(\alpha_{Tmin})^{1/2}$ are analogous to the principal indices of refraction; and $(\alpha_{T1})^{1/2}$ and $(\alpha_{T2})^{1/2}$ are analogous to the indices of refraction corresponding to the vibration directions.

For 3D flow along the principal permeability directions, the longitudinal and transverse dispersivities take the following values:

3D flow direction	longitudinal dispersivity	transverse dispersivities
x_{max}	α_{Lmax}	α_{Tmid}, α_{Tmin}
x_{mid}	α_{Lmid}	α_{Tmax}, α_{Tmin}
x_{min}	α_{Lmin}	α_{Tmax}, α_{Tmid}

Guidelines for applying dispersion model

Some informal guidelines may be given concerning values of dispersivities when other data are not available. Longitudinal dispersivities may be considered to either be on the order of the size of the largest hydrogeologic or flow inhomogeneities along the transport reach, or the distance between inhomogeneities, whichever is the greater value. For transport in pure homogeneous sand, longitudinal dispersivity is on the order of grain size. This is the type of situation where the isotropic-media dispersion model describes well the observed transport behavior. In the case of a sandy aquifer containing well-distributed inclusions of less permeable material, the longitudinal dispersivity required to correct an average advective transport that has passed by many of the inclusions would be on the order of the larger of either inclusion size or distance between inclusions.

Should the dispersivity, estimated on the basis of the size of inhomogeneities or distance between them, be greater than about one tenth of the longest transport reach, then the meaningful use of a constant-dispersivity dispersion model must be questioned. In such a case, the ideal action to take would be to more explicitly define the field distribution of velocity by taking into account the actual geometry of inhomogeneities. This would correctly account for most of the transport that takes place as advective in nature, with much smaller contributions of the approximate dispersive process. Given a better-defined velocity field, and in the absence of other data, dispersivity should then be chosen based on the largest postulated inhomogeneities met along a given average stream tube. The size and distribution of inhomogeneities not explicitly taken into account by the average flow field may be postulated based on the best available knowledge of local geology.

Transverse dispersivity, α_T, is typically even less well known for field problems than longitudinal dispersivity. Values of α_T used in simulation are typically between one tenth and one third of α_L. In systems with anisotropic permeability, α_T may be less than one hundredth of α_L for flows along the maximum permeability direction (Gelhar and Axness, 1983). Should simulated transport in a particular situation be sensitive to the value of transverse dispersivity, further data collection is necessary and the transport model must be interpreted with great care.

The ad-hoc models for longitudinal dispersion in anisotropic media presented in the previous sections allow for simulation experiments with two or three principal longitudinal dispersivities that may be of special interest in systems with well-defined anisotropy values. Depending on the particular geometry of layers or inhomogeneities causing the permeability anisotropy, the longitudinal dispersivity in the minimum permeability direction, α_{Lmin}, may be either greater or smaller than those in the middle and maximum permeability directions, α_{Lmid} and α_{Lmax}. However, use of the anisotropic-media dispersion model is advised only when clearly required by field data, and the additional longitudinal dispersion parameters are not intended for general application without evaluation of their applicability in each particular case. Another use of the ad-hoc model is in the case of 2D cross sectional or 3D simulation wherein the lateral extent of the system is much greater than the vertical extent. In this case, lateral flow and transport may be affected by heterogeneities similar in size to the lateral scale, and vertical flow and transport may be affected by heterogeneities similar in size to the much smaller vertical scale. Here, it makes sense to employ the ad-hoc model to assign different longitudinal dispersivities for lateral and vertical flow.

2.6 Unified Description of Energy and Solute Transport

Unified energy-solute balance

The saturated-unsaturated ground-water energy balance (2.28) is simply an accounting of energy fluxes, sources and sinks which keeps track of how the energy per unit volume of solid matrix plus fluid, $[\varepsilon S_w \rho c_w + (1-\varepsilon)\rho_s c_s]T$, changes with time at each point in space. The saturated-unsaturated ground-water balance of solute plus adsorbate mass, (2.31), is similarly an accounting of solute and adsorbate fluxes, sources and sinks, which keeps track of how the species mass (solute plus adsorbate mass) per unit volume of solid matrix plus fluid, $(\varepsilon S_w \rho C + (1-\varepsilon)\rho_s C_s)$, changes with time at each point in space. Both balances, therefore, track a particular quantity per unit volume of solid matrix plus fluid.

The fluxes of energy and solute mass in solution, moreover, are caused by similar mechanisms. Both quantities undergo advection based on average flow velocity, $\underline{v}$. Both quantities undergo dispersion. Both quantities undergo diffusion; the diffusive solute mass flux is caused by molecular or ionic diffusion within the fluid, while the diffusive energy flux occurs by thermal conduction through both fluid and solid. Fluid sources and sinks give rise to similar sources and sinks of energy and solute mass. Energy and species mass may both be produced by zero-order processes, wherein energy may be produced by an exothermic reaction and solute may be produced, for example, by a biological process. The linear adsorption process affecting solutes is similar to the storage of energy in solid portion of an aquifer. Only the nonlinear sorption processes and first-order production of solute and adsorbate have no readily apparent analogy in terms of energy.

Thus, the balances of energy per unit volume, (2.28), and total species mass per unit volume, (2.31), may be expressed in a single unified balance in terms of a variable, U(x,y,t), which may represent either T(x,y,t) or C(x,y,t), as follows:

$$\begin{aligned}&\frac{\partial}{\partial t}\left(\varepsilon S_w \rho c_w U\right)+\frac{\partial}{\partial t}\left[(1-\varepsilon)\rho_s U_s\right]\\&\quad+\underline{\nabla}\cdot\left(\varepsilon S_w \rho c_w \underline{v}\, U\right)-\underline{\nabla}\cdot\left\{\rho c_w\left[\varepsilon S_w\left(\sigma_w \underline{\underline{I}}+\underline{\underline{D}}\right)+(1-\varepsilon)\sigma_s \underline{\underline{I}}\right]\cdot\underline{\nabla}U\right\}\\&\qquad=Q_p c_w U^* + \varepsilon S_w \rho \Gamma_w + (1-\varepsilon)\rho_s \Gamma_s\end{aligned} \tag{2.47}$$

where:

<u>for energy transport</u>

$$U \equiv T,\quad U_s \equiv c_s T,\quad U^* \equiv T^*,\quad \sigma_w \equiv \frac{\lambda_w}{\rho c_w},\quad \sigma_s \equiv \frac{\lambda_s}{\rho c_w}$$
$$\Gamma_w \equiv \gamma_o^w,\quad \Gamma_s \equiv \gamma_o^s \tag{2.47a}$$

<u>for solute transport</u>

$$U \equiv C,\quad U_s \equiv C_s,\quad U^* \equiv C^*,\quad \sigma_w \equiv D_m,\quad \sigma_s \equiv 0,\quad c_w \equiv 1 \tag{2.47b}$$

where C_s is defined by (2.34a), (2.35a) or (2.36a), depending on the isotherm.

By simple redefinition according to (2.47a) or (2.47b), equation (2.47) directly becomes the energy or species mass balance. This redefinition is automatically carried out by SUTRA as a result of whether the user specifies energy or solute-transport simulation.

<u>Fluid-mass-conservative energy-solute balance</u>

A further consideration is required before obtaining the form of the unified energy/solute balance as implemented in SUTRA. The amount of energy or solute per unit combined matrix-fluid volume may change either due to a change in the total fluid mass in the volume even when concentration and temperature remain constant (see relation (2.10)). Such a change in fluid mass may be caused by changes in fluid saturation, or by pressure changes affecting compressive storage.

The energy and solute balances as well as their unified form, (2.47), track both types of contributions to changes in total stored energy or solute mass. However, the fluid saturation and pressure change contribution to energy and solute balances are already implicitly accounted for by the fluid mass balance.

The fluid mass balance contribution to solute and energy balances is expressed by the product of the fluid mass balance, equation (2.22) (which tracks changes in fluid mass per unit volume), with $c_w U$ (which represents either energy or solute mass per unit fluid mass). Note that $c_w \equiv 1$ for solute transport. This product tracks energy or solute mass changes per unit volume due to fluid mass changes per unit volume:

$$(c_w U)\frac{\partial(\varepsilon S_w \rho)}{\partial t} + (c_w U)\underline{\nabla}\cdot(\varepsilon S_w \rho \underline{v}) = (c_w U) Q_p \tag{2.48}$$

where the solute mass source, Υ, is neglected. Comparison of (2.48) with (2.47) will reveal that the terms on the left of (2.48) also appear in the unified balance equation.

Prior to substituting (2.48) for the duplicate terms in (2.47), the search for redundant terms may be extended to a balance of species mass or energy stored in the solid matrix, rather than in the fluid. A simple mass balance for the solid matrix is:

$$\frac{\partial}{\partial t}\left[(1-\varepsilon)\rho_s\right] + \nabla\cdot\left[(1-\varepsilon)\rho_s \underline{v}_s\right] = 0 \tag{2.49}$$

$\underline{v}_s$		[L/s]	net solid matrix velocity

Due to the assumption that the net solid matrix velocity, $\underline{v}_s$, is negligible, the associated term of (2.49) is dropped. The contribution of this simple solid matrix mass balance to the unified solute-energy balance may again be obtained by taking the product of (2.49) with U_s:

$$(U_s)\frac{\partial}{\partial t}\left[(1-\varepsilon)\rho_s\right] = 0 \tag{2.50}$$

A comparison reveals that this term also appears in (2.47).

The redundant information in the unified energy-solute balance which keeps track of both solid matrix and fluid mass balance contributions may be directly removed from (2.47) by subtracting (2.48) and (2.50). The result is:

$$\begin{aligned} &\varepsilon S_w \rho c_w \frac{\partial U}{\partial t} + (1-\varepsilon)\rho_s \frac{\partial U_s}{\partial t} \\ &\quad + \varepsilon S_w \rho c_w \underline{v}\cdot\underline{\nabla}U - \underline{\nabla}\cdot\left\{\rho c_w\left[\varepsilon S_w\left(\sigma_w \underline{\underline{I}} + \underline{\underline{D}}\right) + (1-\varepsilon)\sigma_s \underline{\underline{I}}\right]\cdot\nabla\underline{U}\right\} \\ &\qquad = Q_p c_w (U^* - U) + \varepsilon S_w \rho \Gamma_w + (1-\varepsilon)\rho_s \Gamma_s \end{aligned} \tag{2.51}$$

where:

<u>for energy transport</u>

$$U \equiv T, \quad U_s \equiv c_s T, \quad U^* \equiv T^*, \quad \sigma_w \equiv \frac{\lambda_w}{\rho c_w}, \quad \sigma_s \equiv \frac{\lambda_s}{\rho c_w}, \quad \Gamma_w \equiv \gamma_o^w, \quad \Gamma_s \equiv \gamma_o^s \tag{2.51a}$$

<u>for solute transport</u>

$$U \equiv C, \quad U_s \equiv C_s, \quad U^* \equiv C^*, \quad \sigma_w \equiv D_m, \quad \sigma_s \equiv 0, \quad c_w \equiv 1 \tag{2.51b}$$

where C_s is defined by (2.34a), (2.35a) or (2.36a), depending on isotherm.

It is assumed in equation (2.51) that c_w and c_s are not time-dependent.

For numerical simulation, this equation may be termed a "fluid-mass conservative" form of the energy or species mass balance. When approximated numerically, the unified balance in the original form, (2.47), would contain approximation errors in both the fluid mass balance contributions (based on pressure and saturation changes) and the temperature or concentration change contribution. However, in the revised form, equation (2.51), the complete fluid mass balance contribution has already been analytically accounted for before any numerical approximation takes place. Thus, the total approximation error for the unified balance, (2.51), is significantly less as it is due to the temperature or concentration change contribution only.

The unified energy-species mass balance is brought to its final form by noticing that the form of the term, $\partial U_s/\partial t$, for energy transport, is the same as that for solute transport when using the equilibrium sorption relation (2.33), and that the form of the energy production terms is similar to that of relations (2.37a) and (2.37b) for the mass production process:

$$\begin{aligned}&\left[\varepsilon S_w \rho c_w + (1-\varepsilon)\rho_s c_s\right]\frac{\partial U}{\partial t}\\ &\quad + \varepsilon S_w \rho c_w \underline{v}\cdot\underline{\nabla}U - \underline{\nabla}\cdot\left\{\rho c_w\left[\varepsilon S_w\left(\sigma_w \underline{\underline{I}} + \underline{\underline{D}}\right) + (1-\varepsilon)\sigma_s \underline{\underline{I}}\right]\cdot\underline{\nabla}U\right\}\\ &\quad = Q_p c_w (U^* - U) + \varepsilon S_w \rho \gamma_1^w U + (1-\varepsilon)\rho_s \gamma_1^s U_s + \varepsilon S_w \rho \gamma_o^w + (1-\varepsilon)\rho_s \gamma_o^s\end{aligned} \tag{2.52}$$

where:

<u>for energy transport</u>

$$U \equiv T, \quad U^* \equiv T^*, \quad \sigma_w \equiv \frac{\lambda_w}{\rho c_w}, \quad \sigma_s \equiv \frac{\lambda_s}{\rho c_w}, \quad \gamma_1^w \equiv \gamma_1^s \equiv 0 \tag{2.52a}$$

<u>for solute transport</u>

$$U \equiv C, \quad U_s \equiv C_s, \quad U^* \equiv C^*, \quad \sigma_w \equiv D_m, \quad \sigma_s \equiv 0, \quad c_s \equiv \kappa_1, \quad c_w \equiv 1 \tag{2.52b}$$

where C_s is defined by (2.34a), (2.35a) or (2.36a), and κ_1 is defined by (2.34c), (2.35c) or (2.36c), depending on the isotherm.

The fluid-mass-conservative form of the unified energy-species mass balance, (2.52), is exactly that which is implemented in SUTRA.

Chapter 3: Fundamentals of Numerical Algorithms

SUTRA methodology is complex because: (1) density-dependent flow and transport requires two interconnected simulation models, (2) fluid properties are dependent on local values of temperature or concentration, (3) geometry of a field area and distributions of hydrogeologic parameters may be complex, and (4) hydrologic stresses on the system may be distributed in space and change with time. Furthermore, a tremendous amount of data must be evaluated by SUTRA with precision. This requires great computational effort, and considerable numerical intricacy is required to minimize this effort. The mathematically elegant finite-element and integrated-finite-difference hybrid method employed by SUTRA allows great numerical flexibility in describing processes and characteristics of flow and transport in hydrologic field systems. However, unlike simulation models based purely on the method of finite differences, the numerical aspects of which allow straightforward interpretation at an intuitive level, some finite-element aspects of SUTRA methodology require interpretation at a less physical level and from a more mathematical point of view.

The following description of SUTRA numerical methods uses a simplified, constant-density water-table aquifer case in 2D as an illustrative example. While precise mathematically, this example is not used to demonstrate an actual application of SUTRA, as SUTRA, in fact, does not simulate a moving water table. The example is only used as a device through which to explain the theory and use of the primary numerical methods employed in SUTRA, and the water table is invoked to allow discussion of a simple nonlinearity. The basic methods, which are only demonstrated here, are applied in detail in Chapter 4, "Numerical Methods," to the SUTRA fluid mass balance and unified energy-species mass balance.

The water-table aquifer fluid mass balance equation is useful for demonstration of basic numerical methods employed on SUTRA governing equations, because it displays some of the salient aspects of the SUTRA equations: a time derivative, a nonlinear term involving space-derivatives, and a source term. The simplified fluid mass balance equation is as follows:

$$S_o \frac{\partial h}{\partial t} - \underline{\nabla} \cdot \left(K \underline{\nabla} h \right) = Q^* \qquad (3.1)$$

where $Q^* = (Q_p/\rho)$

and

$S_o(x,y)$	$[L^{-1}]$	specific storativity
$h(x,y,t)$	[L]	hydraulic head (sum of pressure head and elevation head)
$K(x,y)$	[L/s]	hydraulic conductivity (assumed for this example to be isotropic)

$Q^*(x,y)$	$[s^{-1}]$	volumetric fluid source (volume fluid injected per time / volume aquifer) (assumed constant for this example)
$Q_p(x,y)$	$[M/(L^3 \cdot s)]$	fluid mass source (mass fluid injected per time / volume aquifer) (assumed constant for this example)
ρ	$[M/L^3]$	fluid density (assumed constant for this example)

This equation, (3.1), is obtained from the SUTRA fluid mass balance, (2.24), by assuming saturated conditions, constant concentration and temperature, constant fluid density, and using the definition of hydraulic conductivity, $K \equiv (k\rho|\underline{g}|)/\mu$, where $|\underline{g}|$ is the acceleration of gravity, and of hydraulic head, $h \equiv h_p$ + ELEVATION, where pressure head, $h_p \equiv p/(\rho|\underline{g}|)$. For clarity, hydraulic conductivity is assumed isotropic in this example. While (3.1) may be considered a fully 3D mass balance equation, it is assumed that flow takes place only areally in a water-table aquifer with a fixed impermeable base (at z-position, BASE(x,y)), and a moveable free surface (at z-position, h(x,y,t)). The z-direction is oriented vertically upward and the fluid is assumed to be in vertical hydrostatic equilibrium at any (x,y) position (no vertical flow). Aquifer thickness, B(x,y,t) [L], is measured as the distance along z from the free surface to the aquifer base, and may change with time. Aquifer transmissivity, T(x,y,t), is given by:

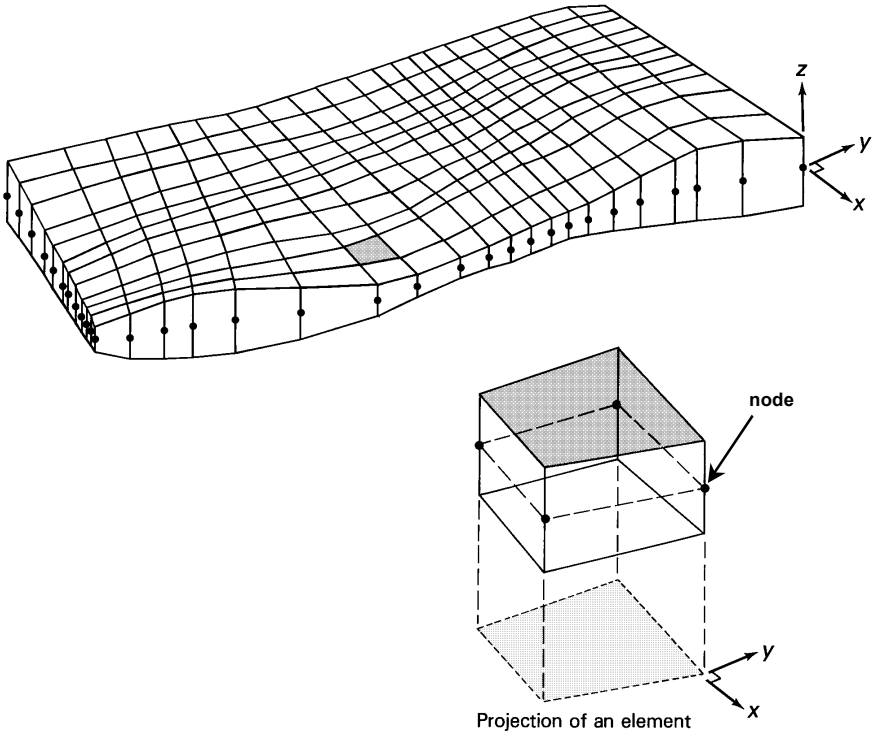
$$T \equiv KB \equiv K(h - BASE) \tag{3.2}$$

T(x,y,t)	$[L^2/s]$	aquifer transmissivity
B(x,y,t)	[L]	aquifer thickness
BASE(x,y)	[L]	elevation of aquifer base

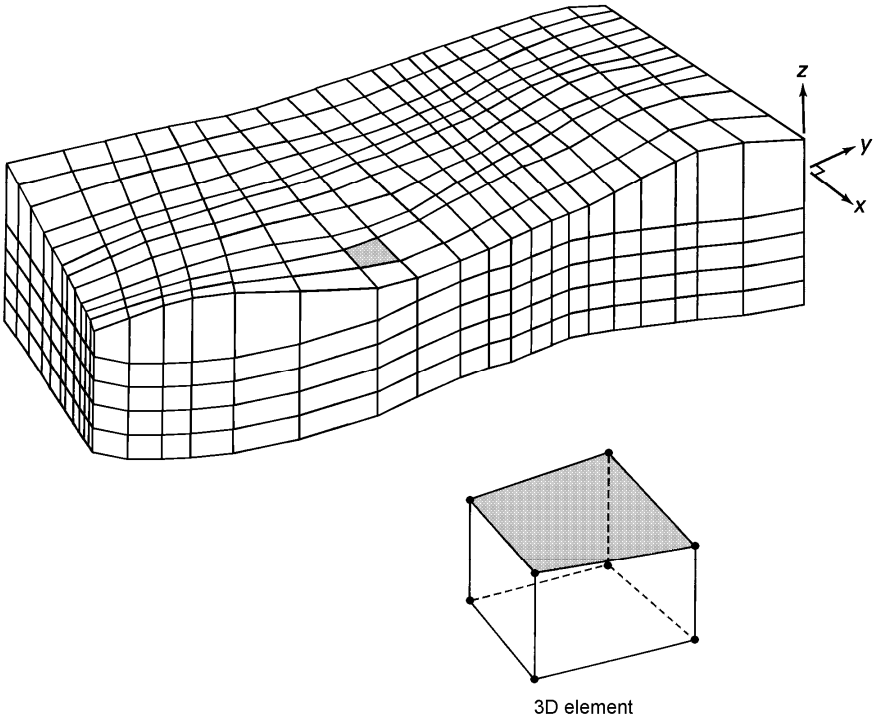
The above assumption, in effect, makes (3.1) a 2D mass balance equation that is applied to an aquifer of finite thickness. The 2D form of (3.1) describing an areal fluid mass balance for water-table aquifers in terms of a head-dependent transmissivity arises during the basic numerical analysis of (3.1) in section 3.3, "Integration of Governing Equation in Space."

3.1 Spatial Discretization by Finite Elements

Regardless of whether SUTRA is applied as a 2D or 3D model, the region of space in which flow and transport are to be simulated is defined in three space dimensions. For 2D simulation, the 3D bounded volume of an aquifer that is to be simulated by SUTRA is completely divided up into a single layer of contiguous blocks. For 3D simulation, the 3D bounded volume of an aquifer that is to be simulated by SUTRA is completely divided up into a set of contiguous

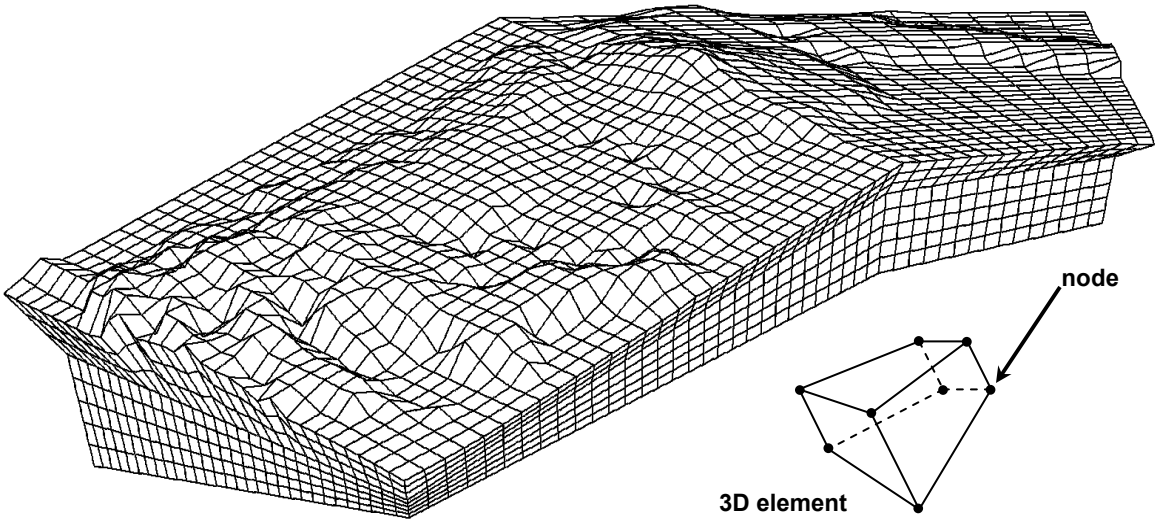


(a)

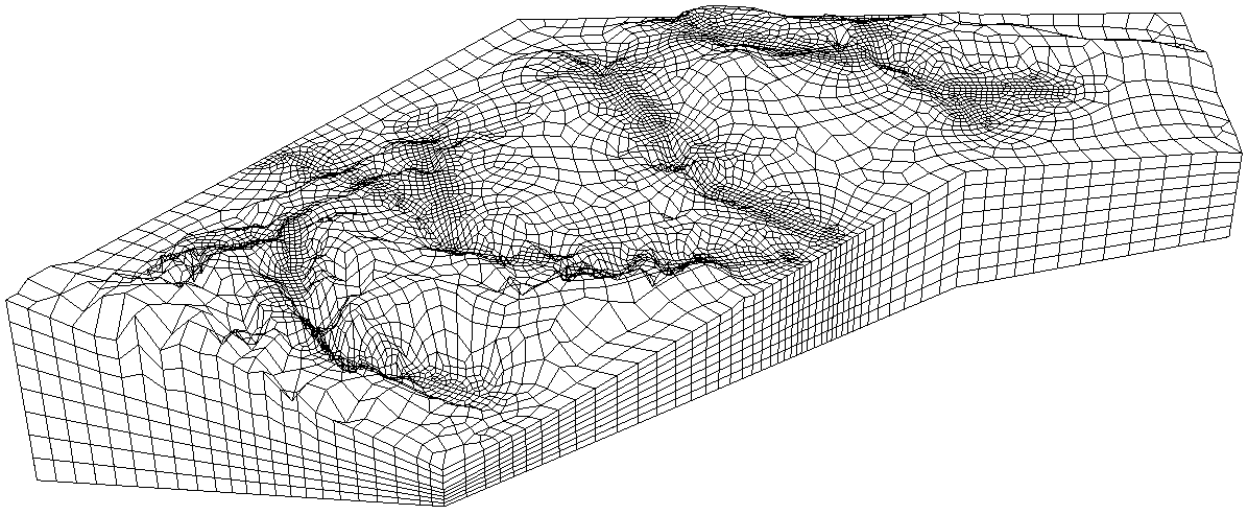


(b)

Figures 3.1a,b. (a) Two-dimensional finite-element mesh and quadrilateral element. (b) Three-dimensional vertically-aligned finite-element mesh (with a regular pattern of elements) and vertically-aligned generalized hexahedral element. In the lower figure, the z-direction is vertical and element edges in the vertical direction are aligned with the z-coordinate direction. External and internal surfaces of the mesh need not be planar.



(c)



(d)

Figures 3.1c,d. (c) Three-dimensional non-aligned finite-element mesh (with a regular pattern of elements) and non-aligned generalized hexahedral element. In general, the z-direction need not be vertical or aligned with the mesh, and the element edges need not be aligned with any coordinate direction. External and internal surfaces of the mesh need not be planar. (d) Three-dimensional vertically-aligned finite-element mesh (with an irregular pattern of elements in the two "horizontal" numbering directions).

blocks, which may or may not be organized in layers. The blocks are called "finite elements." The subdivision is not done simply in a manner that creates one block (element) for each portion of the aquifer system that has unique hydrogeological characteristics. Each hydrogeologic unit is in fact divided into many elements, giving the subdivided aquifer region the appearance of a fine mesh. Thus, subdivision of the aquifer region to be simulated into blocks is referred to as "creating the finite-element mesh."

The basic building block of a finite-element mesh is a finite element. The type of element employed by SUTRA for 2D simulation is a quadrilateral that has a finite thickness in the third

space dimension. This type of a quadrilateral element and a typical 2D mesh are shown in Figure 3.1a.

All twelve edges of the 2D quadrilateral element are perfectly straight. Four of these edges are parallel to the z-coordinate direction. The x-y plane (which contains the two coordinate directions of interest) bisects each of the edges parallel to z, so that the top and bottom surfaces of the element are mirror images of each other reflected about the central x-y plane in the element. The midpoint of each z-edge (the point where the x-y plane intersects) is referred to as a nodal point (or node). Thus, the element has a 3D shape, but always has only four nodes, each of which in fact, represents the entire z-edge on which it is located. The nodes mark the fact that, in this type of element, some aquifer parameters may be assigned a different value at each z-edge of the element. The lack of nodes outside of the x-y plane is what makes this element 2D; while some aquifer parameters may vary in value from node to node (i.e., from z-edge to z-edge), no parameters may be assigned varying values in the z-direction.

Within a 2D finite-element mesh there is only a single layer of elements, the nodes of which lie in the x-y plane. Nodal points are always shared by the elements adjoining the node. Only nodes at external corners of the mesh are not contained in more than one element. The top and bottom surfaces are at every (x,y) point equidistant from the x-y plane, but the thickness of the mesh, measured in the z-direction, may vary smoothly from point to point.

When projected on the x-y plane, as in Figure 3.1a, a 2D finite-element mesh composed of the type of elements used by SUTRA appears as a mesh of contiguous quadrilaterals with nodes at the corners; hence, the term, “quadrilateral element”. Although the mesh in Figure 3.1a is regularly connected (four elements attached to each internal node), 2D meshes may have irregular connections; in other words, any number of elements may be connected to a given node.

The type of element employed by SUTRA for 3D simulation is a generalized hexahedron. This type of hexahedral element and a typical 3D meshes are shown in Figures 3.1b-d.

All twelve edges of the 3D generalized hexahedral element are perfectly straight, although the six faces of the element need not be planar. The 3D SUTRA element differs from the 2D element in that none of the edges of the 3D element need be parallel to the z-coordinate direction, and the geometry of the 3D element is defined by eight (instead of four) nodes, each of which represents the intersection of three edges (i.e., a corner of the element).

Unlike a 2D mesh, a 3D SUTRA finite-element mesh is not restricted to a single layer of elements, nor do the nodes of a 3D mesh need to align in any way with the x-, y-, and z-coordinate directions. In SUTRA version 2.0 (2D3D.1), the 3D mesh had to be logically rectangular; it had to be possible (hypothetically) to reposition the nodes to form a regular mesh of cube-shaped elements without adding or deleting any connections between nodes. In other words, although the geometry of the mesh may have been irregular, the nodes had to be logically connected in the same manner as in a regular mesh. A 3D, logically rectangular mesh can be thought of as consisting of rows, columns, and layers of elements, with each row, column, and layer containing its full complement of elements. **The requirement that 3D meshes be logically rectangular has been relaxed in this version of SUTRA.** The 3D SUTRA mesh illustrated in Figure 3.1b is logically rectangular and vertically aligned; this mesh is an extrusion in the z-direction of the 2D mesh that makes up the top surface. The 3D SUTRA mesh illustrated in

Figure 3.1c is logically rectangular and non-aligned in the vertical direction. The 3D SUTRA mesh illustrated in Figure 3.1d is not logically rectangular. Rather, it is what is referred to in input dataset 2B as "layered" – see Appendix B.

3.2 Representation of Coefficients in Space

Aquifer parameters and coefficients that vary from point to point in an aquifer, such as specific storativity, S_o, and hydraulic conductivity, K, are represented in an approximate way in SUTRA. Parameters are either assigned a particular constant value in each element of a finite-element mesh (elementwise), or are assigned a particular value at each node in the mesh in two possible ways (nodewise or cellwise). Descriptions of elementwise, nodewise, and cellwise discretization are given for the 2D example that follows, based on (3.1). The discretization procedures in 3D are analogous to those in 2D.

In the water-table aquifer, for a simple example, a regular 2D mesh is used. The step-like appearance of elementwise assignment of K values over this simple mesh is shown in Figure 3.2. Nodewise assignment for head over this mesh results in a continuous surface of h values as shown in Figure 3.3, with linear change in value between adjoining nodes along (projected) element edges. Cellwise assignment is employed for specific storativity, S_o, and the time derivative, $\partial h/\partial t$. This results in a step-like appearance of the assigned values over the mesh similar to that of elementwise assignment in Figure 3.2, but each cell is centered on a node, not on an element. Cell boundaries are half way between opposite sides of an element and are shown for the regular mesh in Figure 3.4. Thus the spatial distributions of parameters, K, h and S_o, are discretized (i.e., assigned discrete values) in three different ways: K, elementwise, h, nodewise, and S_o, cellwise.

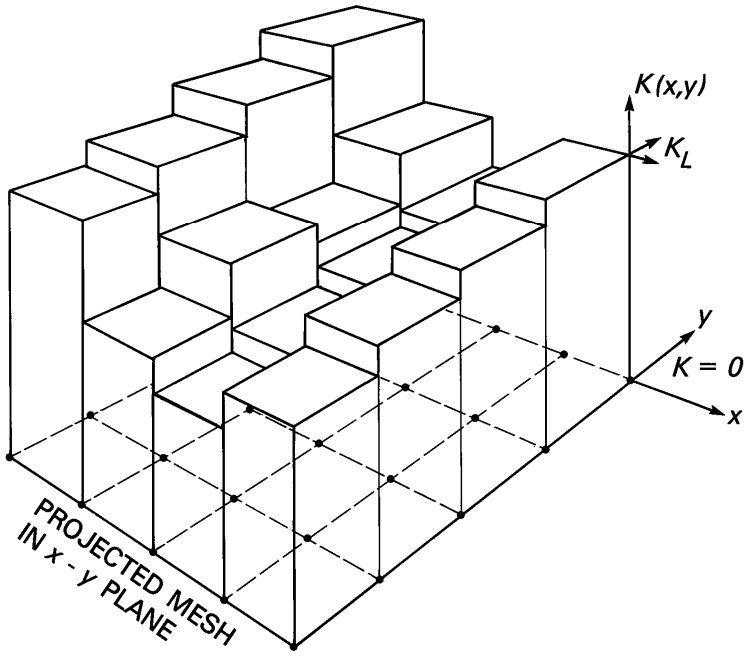


Figure 3.2. Elementwise discretization of coefficient K(x,y). K_L is the value of K in element L.

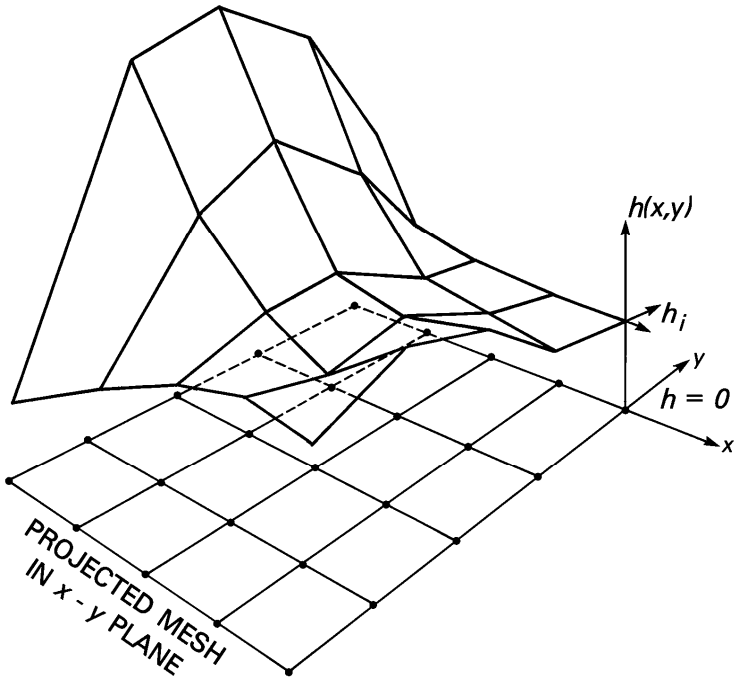


Figure 3.3. Nodewise discretization of coefficient h(x,y).

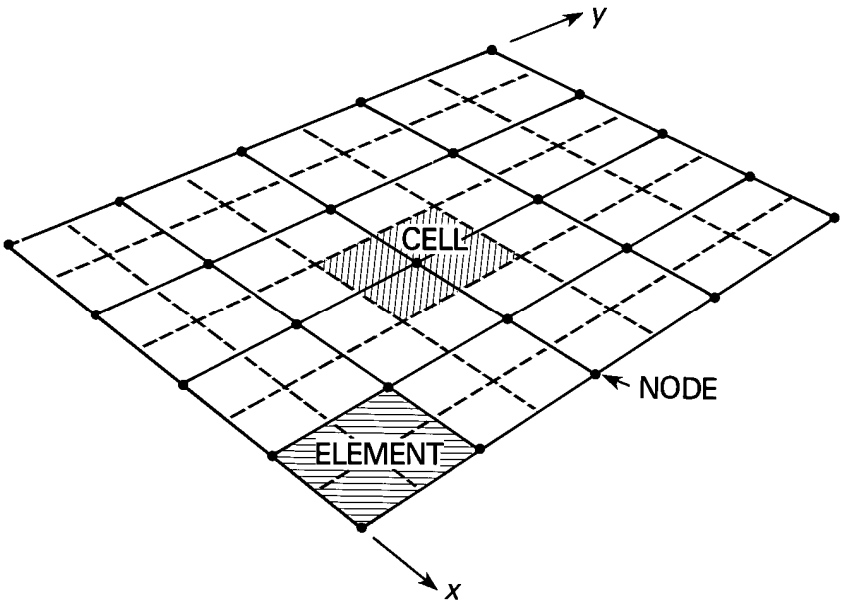


Figure 3.4. Cells, elements and nodes for a two-dimensional finite-element mesh composed of quadrilateral elements. Dashed lines connect the midpoints along the element sides.

Because the internal program logic depends on the type of discretization, SUTRA expects particular parameters or equation terms to be discretized elementwise, nodewise, or cellwise. The primary dependent variables of the SUTRA code, p, and T or C, (in this example case, only hydraulic head, h), are expressed nodewise when used in terms that calculate fluxes of fluid mass, solute mass or energy.

Elementwise discretization

The equation that gives the values, over the finite element mesh, of an elementwise parameter, may be expressed for the hydraulic conductivity of the present 2D example as:

$$K(x,y) \approx \sum_{L=1}^{NE} K_L(x,y) \tag{3.3}$$

where the elements have been numbered from one to NE (total number of elements in the mesh), and $K_L(x,y)$ [L/s] has the value of hydraulic conductivity of element L for (x,y) coordinates within the element, and a value of zero outside the element. Thus $K_L(x,y)$ is the flat-topped "box" standing on an element L, in Figure 3.2, and K(x,y) is represented in a discrete approximate way by the sum of all the "boxes". Note that $K_L(x,y)$ has the same value throughout each 2D element from the top to the bottom.

Nodewise discretization

The equation that gives the values, over the finite-element mesh, of a nodewise value, may be expressed for the 2D mesh as:

$$h(x,y,t) \approx \sum_{j=1}^{NN} h_j(t)\,\phi_j(x,y) \tag{3.4}$$

where the nodes have been numbered from one to NN (total number of nodes in the mesh). There are NN coefficients, $h_j(t)$, each of which is assigned the value of head at the coordinates (x_j,y_j) of node number j. These nodal head values may change with time to represent transient responses of the system. The function $\phi_j(x,y)$ is known as the "basis function". It is the basis functions that spread values of head between the nodes when head is defined only at the nodal points by values of h. There is one basis function $\phi_j(x,y)$ defined for each node, j, of the NN nodes in the mesh. Suffice it to say, at this point, that at the node j, to which it belongs, the basis function $\phi_j(x,y)$ has a value of one. At all other nodes i, $i \neq j$, in the mesh, it has a value of zero. It drops linearly in value from one to zero along each projected element edge to which the node j is connected. This means that even when all the NN products of h_j and $\phi_j(x,y)$ are summed (as in relation (3.4)), if the sum is evaluated at the coordinates (x_j,y_j) of node j, then h(x,y) exactly takes on the assigned value, h_j. This is because the basis function belonging to node j has a value of one at node j, and all other basis functions belonging to other nodes, i, $i \neq j$, have a value zero at node j, thereby dropping them from the summation in (3.4). Basis functions are described mathematically in section 4.1, "Basis and Weighting Functions."

Cellwise discretization

The equation that gives the values, over the finite-element mesh, of a cellwise parameter may be expressed for the specific storativity of the present 2D example as:

$$S_o(x,y) \approx \sum_{i=1}^{NN} S_i(x,y) \tag{3.5}$$

where $S_i(x,y)$ has the value of specific storativity of the cell centered on node i for (x,y) coordinates within the cell, and a value of zero outside the cell. Thus, $S_i(x,y)$ is a flat-topped "box" standing on a cell i in Figure 3.4, and $S_o(x,y)$ is represented in a discrete, approximate way by the sum of all the "boxes." Note $S_i(x,y)$ has the same value in the z-direction from the top to bottom of each 2D element.

Reviewing the example problem, K is assigned elementwise and both S_o and $\partial h/\partial t$ are assigned cellwise. Hydraulic head, h(x,y,t), and element thickness, B(x,y,t), measured in the z-direction, are both discretized nodewise, with the nodewise expansion for thickness:

$$B(x,y) \approx \sum_{i=1}^{NN} B_i(t)\ \phi_i(x,y) \tag{3.6}$$

The values $B_i(t)$ are the NN particular values of element thickness at the nodes, and these values may change with time in the present water-table example. Relation (3.6) should call to mind a vision of discretized values of thickness represented by a surface similar to that of Figure 3.3. The head surface of Figure 3.3 may stretch or shrink to move up or down as the head values at nodes, $h_i(t)$, change with time due to stresses on the aquifer system. The nodewise discretized surface may be viewed as the water table, and the element thickness, at any point (x,y), as the thickness of the water-table aquifer.

3.3 Integration of Governing Equation in Space

Approximate governing equation and weighted residuals method

The governing equation for the water-table example may be written in operator form as:

$$O(h) = S_o \frac{\partial h}{\partial t} - \underline{\nabla} \cdot (K \underline{\nabla} h) - Q^* = 0 \tag{3.7}$$

Certain variables in this equation are approximated through elementwise and nodewise discretization. Particular terms of the equation are approximated through cellwise discretization. The result is that neither the derivatives, nor the variables are described exactly. Relation (3.7) no longer exactly equals zero:

$$\langle\langle O(h) \rangle\rangle = R(x,y,t) \tag{3.8}$$

where $\langle\langle O(h) \rangle\rangle$ is the result of approximating the terms of the equation and the variables, and R(x,y,t) is the residual value of the approximated equation. When simulating a system with a numerical model based on approximation of the governing equation, $\langle\langle O(h) \rangle\rangle$, the residual, R, must be kept small everywhere in the simulated region and for the entire time of simulation in order to accurately reproduce the physical behavior predicted by the exact governing equation, (3.7).

In order to achieve a minimum error, a method of weighted residuals is applied to (3.8). The purpose of the method of weighted residuals is to minimize the error of approximation in particular subregions of the spatial domain to be simulated. This is done by forcing a weighted average of the residual to be zero over the subregions. This idea is the most abstract of those required to understand SUTRA methodology. The Galerkin method of weighted residuals chooses to use the "basis function", $\phi_i(x,y)$, mentioned in the previous section, as the weighting function for calculation of the average residual:

$$\int_V \left\langle\left\langle O(h)\right\rangle\right\rangle \phi_i(x,y)\ dV = \int_V R(x,y,t)\,\phi_i(x,y)\ dV = 0 \qquad i = \overline{1, NN} \tag{3.9}$$

where V is the volume of the region to be modeled. The model volume is completely filled by a single layer of quadrilateral finite elements. Relation (3.9) is actually NN relations, one for each of NN nodes in the finite element mesh as indicated by the notation, $i = \overline{1, NN}$.

In each relation, the integral sums the residual weighted by the basis function over a volume of space. Each integrated weighted residual is forced to zero over the region of space in which $\phi_i(x,y)$ is nonzero. This region includes only elements which contain node i, because of the manner in which the basis function is defined, as described earlier. Thus, over each of these NN subregions of a mesh, the sum of positive and negative residuals after weighting is forced to zero by relation (3.9). This, in effect, minimizes the average error in approximating the governing equation over each subregion.

After stating that the integral of weighted residuals must be zero for each subregion of the mesh as in (3.9), the derivation of the numerical methods becomes primarily a job of algebraic manipulation. The process is begun by substitution of the governing equation for $\left\langle\left\langle O(h)\right\rangle\right\rangle$ in (3.9):

$$\int_V \left\langle\left\langle S_o \frac{\partial h}{\partial t}\right\rangle\right\rangle \phi_i(x,y)\,dV - \int_V \left\langle\left\langle \underline{\nabla}\cdot(K\underline{\nabla}h)\right\rangle\right\rangle \phi_i(x,y)\,dV - \int_V \left\langle\left\langle Q^*\right\rangle\right\rangle \phi_i(x,y)\,dV = 0 \qquad i = \overline{1, NN} \tag{3.10}$$

The terms in double angle brackets are the approximate discrete forms of the respective terms in (3.7). These are expanded in the manipulations that follow. Relation (3.10) is discussed term by term in the following paragraphs.

Cellwise integration of time-derivative term

The first term involving the volume integral of the time derivative may be written in terms of the three space dimensions, x, y, and z. Although the governing equation and parameters vary only in two space dimensions, they apply to the complete 3D region to be modeled.

$$\int_V \left\langle\left\langle S_o \frac{\partial h}{\partial t}\right\rangle\right\rangle \phi_i(x,y)\,dV = \int_z\int_y\int_x \left\langle\left\langle S_o \frac{\partial h}{\partial t}\right\rangle\right\rangle \phi_i(x,y)\,dz\,dy\,dx$$
$$= \int_y\int_x \left\langle\left\langle S_o \frac{\partial h}{\partial t}\right\rangle\right\rangle \phi_i(x,y)\left[\int_z dz\right] dx\,dy \tag{3.11}$$

The rearrangement in the final term of (3.11) is possible because no parameter depends on z. In fact, referring to (3.2), the aquifer thickness, B(x,y,t), may be defined as:

$$B(x,y,t) = \int_{z(t)} dz = h(x,y,t) - BASE(x,y) \tag{3.12}$$

The final term of (3.11) is then:

$$\int_y\int_x \left\langle\left\langle S_o \frac{\partial h}{\partial t}\right\rangle\right\rangle \phi_i(x,y)\;\; B(x,y,t)\,dx\,dy \tag{3.13}$$

Now cellwise discretization is chosen for S_o and for $\partial h/\partial t$, making these terms take on a constant value for the region of each cell i. The region of cell i is the same region over which $S_i(x,y)$ is non-zero. Then, for any cell i, term (3.13) becomes:

$$S_i \frac{\partial h_i}{\partial t}\int_y\int_x \phi_i(x,y)\,B(x,y,t)\,dx\,dy \tag{3.14}$$

where S_i and $\partial h_i/\partial t$ are the values taken by S_o and $\partial h/\partial t$ in cell i.

It can be shown that the volume of cell i, denoted by $V_i(t)$, is, in fact, the integral in (3.14):

$$V_i(t) = \int_y\int_x \phi_i(x,y)\,B(x,y,t)\,dx\,dy \tag{3.15}$$

For a particular finite-element mesh, the volume $V_i(t)$ of each cell is determined by numerical integration of (3.15). Numerical integration by Gaussian quadrature is discussed in section 4.3, "Gaussian Integration."

Given the value of the specific storativity of each cell, S_i, the time derivative of head in each cell, $\partial h_i/\partial t$, and given the volume of each cell, $V_i(t)$, determined numerically, the first term of the weighted residual statement takes on its discrete approximation in space:

$$\int_V \left\langle\left\langle S_o \frac{\partial h}{\partial t}\right\rangle\right\rangle \phi_i(x,y)\,dV = S_i \frac{\partial h_i}{\partial t}\,V_i(t) \tag{3.16}$$

Elementwise integration of flux term and origin of boundary fluxes

Manipulation of the second integral in (3.10) begins with the application of Green's theorem, which is an expanded form of the divergence theorem. This converts the integral into two terms, one of which is evaluated only at the surface of the region to be simulated. Green's theorem is:

$$\int_V \left(\underline{\nabla} \cdot \underline{W}\right) A \, dV = \int_\Gamma \left(\underline{W} \cdot \underline{n}\right) A \, d\Gamma - \int_V \left(\underline{W} \cdot \underline{\nabla} A\right) dV \tag{3.17}$$

where A is a scalar and $\underline{W}$ is a vector quantity. The boundary of volume V is denoted by Γ, which includes both the edges and the upper and lower surfaces of the aquifer, and $\underline{n}$ is a unit outward normal vector to the boundary. Application of (3.17) to the second term in (3.10) results in:

$$\begin{aligned} -\int_V \left[\underline{\nabla} \cdot \left\langle\left\langle K \, \underline{\nabla} h \right\rangle\right\rangle\right] \phi_i(x,y) \, dV = &-\int_\Gamma \left[\left\langle\left\langle K \, \underline{\nabla} h \right\rangle\right\rangle \cdot \underline{n}\right] \phi_i \, d\Gamma \\ &+ \int_V \left\langle\left\langle K \, \underline{\nabla} h \right\rangle\right\rangle \cdot \underline{\nabla} \phi_i \, dV \end{aligned} \tag{3.18}$$

The first term on the right of (3.18) contains a fluid flux given by Darcy's law:

$$\varepsilon \, v_{OUT} = -K \underline{\nabla} h \cdot \underline{n} \tag{3.19}$$

where v_{OUT} is the outward velocity at the boundary normal to the bounding surface. Thus, the integral gives the total flow out across the bounding surface, Q_{OUT_i}, in the vicinity of a node i on the surface:

$$Q_{OUT_i} = \int_\Gamma \left(\varepsilon v_{OUT} \phi_{\tilde{i}}\right) d\Gamma \tag{3.20}$$

An inflow would have a negative value of Q_{OUT_i}, and the relation between an inflow, Q_{IN_i}, and outflow is: $Q_{IN_i} = -Q_{OUT_i}$. Thus, the first integral on the right of (3.18) represents flows across boundaries of the water-table aquifer model.

The second integral on the right of (3.18) may be expressed in three spatial coordinates:

$$\begin{aligned} &\int_V \left\langle\left\langle K \, \underline{\nabla} h \right\rangle\right\rangle \cdot \underline{\nabla} \phi_i \, dV = \int_x \int_y \int_z \left\langle\left\langle K \, \underline{\nabla} h \right\rangle\right\rangle \cdot \underline{\nabla} \phi_i \, dz \, dy \, dx \\ &= \int_x \int_y \left\langle\left\langle K \, \underline{\nabla} h \right\rangle\right\rangle \cdot \underline{\nabla} \phi_i \left[\int_z dz\right] dy \, dx = \int_x \int_y \left\langle\left\langle K \, \underline{\nabla} h \right\rangle\right\rangle \cdot \underline{\nabla} \phi_i \; B(x,y,t) \, dy \, dx \end{aligned} \tag{3.21}$$

No term varies in the z-direction, allowing the use of (3.12), which defines aquifer thickness B. Notice that the transmissivity as given by (3.2), $T = KB$, appears in the form of the integral just obtained.

Now the approximation $\langle\langle K\ \underline{\nabla}h\rangle\rangle$ is specified in the integral. Hydraulic head, h(x,y,t), is approximated in a nodewise manner as given by relation (3.4). The integral of (3.21) becomes:

$$\begin{aligned}\int_x\int_y \langle\langle K\ \underline{\nabla}h\rangle\rangle \cdot \nabla\phi_i\ B\ dy\ dx &= \int_x\int_y \left[\langle\langle K\rangle\rangle \underline{\nabla}\sum_{j=1}^{NN} h_j(t)\,\phi_j(x,y)\right]\cdot\underline{\nabla}\phi_i\ B\ dy\ dx \\ = \sum_{j=1}^{NN} h_j(t)\int_x\int_y \langle\langle K\rangle\rangle\left(\underline{\nabla}\phi_j\cdot\underline{\nabla}\phi_i\right) B\ dy\ dx &= \sum_{j=1}^{NN} h_j(t)\,I_{ij}(t)\end{aligned} \tag{3.22}$$

where $\langle\langle K\rangle\rangle$ is the elementwise approximation for K(x,y). The summation and $h_j(t)$ may be factored out of the integral because h_j is a value of head at a node and does not vary with x and y location. The integral is represented by $I_{ij}(t)$ and depends on time because aquifer thickness, B, is time-dependent for this water-table case. For each node i, there are apparently j=NN integrals which need to be evaluated. In fact, due to the way in which basis functions are defined, there are only a few which are nonzero, because $(\underline{\nabla}\phi_j \cdot \underline{\nabla}\phi_i)$ is nonzero only when nodes i and j are in the same finite element. When nodes i and j are in different elements, then $\underline{\nabla}\phi_j$ is zero in the element containing node i.

The integrals are evaluated numerically by Gaussian integration. This is accomplished by first breaking up the integral over the whole volume to be simulated, into a sum of integrals, one each over every finite element in the mesh:

$$I_{ij}(t) = \int_x\int_y \langle\langle K\rangle\rangle\left(\underline{\nabla}\phi_j\cdot\underline{\nabla}\phi_i\right) B\ dy\ dx = \sum_{L=1}^{NE}\ \int_{x_L}\int_{y_L}\ \langle\langle K\rangle\rangle\left(\underline{\nabla}\phi_j\cdot\underline{\nabla}\phi_i\right) B\ dy\ dx \tag{3.23}$$

There are NE elements in the mesh, L is the element number, and x_L and y_L are the x and y spatial domains of element L. Thus, for a given L, the integral over x_L and y_L is integrated only over the area of element L.

Now, the discrete elementwise approximation for hydraulic conductivity, as given by (3.3), allows one term for element L in the summation of (3.23) to be written as:

$$K_L \int_{x_L}\int_{y_L}\left(\underline{\nabla}\phi_j\cdot\underline{\nabla}\phi_i\right) B\ dy\ dx \tag{3.24}$$

Here, the thickness B is specified to vary nodewise. The formula for B in this example is obtained by substituting the nodewise expression for head, (3.4), into the definition of B, relation (3.2).

The integral over one element, as given by term (3.24), must be evaluated numerically. In order to do this, the coordinates of the element L, which has an arbitrary quadrilateral shape as suggested in Figure 3.1a, is transformed to a new coordinate system in which the element is a two-by-two square. Then, Gaussian integration is carried out to evaluate the integral. For a given combination of nodes i and j, this transformation and numerical integration is carried out for all elements in the mesh in which both nodes i and j appear. (There are 16 i-j combinations

evaluated in each quadrilateral element.) The elementwise pieces of the integral for each i-j combination are then summed according to (3.23) in order to obtain the value of the integral over the whole region. The summation is called the "assembly" process. This element transformation, integration of the 16 integrals arising in each element, and summation, makes up a large part of the computational effort of a finite-element model and requires the most complex algorithm in a finite-element model. It is in this way that the second term of (3.10) is evaluated. More information on finite-element integration and assembly may be found in numerical methods texts such as Wang and Anderson (1982), Pinder and Gray (1977), or Huyakorn and Pinder (1983). The details of this method as applied in SUTRA are given in Chapter 4, "Numerical Methods."

Cellwise integration of source

The last term of equation (3.10) deals with sources of fluid to the aquifer such as injection wells. The volume integral may, as before, be written in x, y, and z coordinates:

$$\begin{aligned} -\int_V Q^*(x,y)\,\phi_i(x,y)\,dV &= -\int_x\int_y\int_z Q^*\,\phi_i\,dz\,dy\,dx \\ &= -\int_x\int_y Q^*\,\phi_i\,B(x,y,t)\,dy\,dx \end{aligned} \tag{3.25}$$

where thickness B is introduced because Q^* and ϕ_i do not vary with z. It is assumed that all fluid entering the aquifer within the region of cell i, which surrounds node i, enters at node i. If Q_i^* [L^3/s] is defined as the volume of fluid entering cell i per unit time, then Q^* [s^{-1}], which is the volume of fluid entering the aquifer per unit volume of aquifer per unit time, is given as:

$$Q^*(x,y) = \sum_{i=1}^{NN}\left(\frac{Q_i^*}{V_i}\right) \tag{3.26}$$

This is a cellwise discretization for the source term, Q^*. For cell i:

$$-\int_x\int_y Q^*\,\phi_i\,B\,dy\,dx = -\left(\frac{Q_i^*}{V_i}\right)\int_x\int_y \phi_i\,B\,dy\,dx = -Q_i^* \tag{3.27}$$

Thus, all recharges within cell i due to areal infiltration, well injection or other types are allocated to the source at node i.

This completes the spatial integration of the governing equation for the example problem.

3.4 Time Discretization of Governing Equation

When the integrated terms of the governing equation are substituted in (3.10), the following results:

$$S_i \; V_i(t)\frac{dh_i}{dt} + \sum_{j=1}^{NN} I_{ij}(t)\, h_j(t) = Q_{IN_i} + Q_i^* \qquad i = \overline{1, NN} \tag{3.28}$$

These are NN integrated weighted residual approximations of the governing differential equation, one at each node i in the mesh. Because of the summation term in (3.28), the integrated approximate equation for a node, i, may involve the values of head, $h_j(t)$, at all other nodes in the mesh. The other terms in (3.28) involve only values at node i itself, at which the entire relation is evaluated.

All the parameters in (3.28) are no longer functions of the space coordinates. Each parameter takes on a particular value at each node in the mesh. Some of these values vary with time and a particular time for evaluation of these values needs to be specified. In addition, the time derivative requires discretization.

Time steps

Time is broken up into a series of discrete steps, or time steps. The length of a time step, Δt, is the difference in time between the beginning and the end of a time step:

$$\Delta t_{n+1} = t^{n+1} - t^n \tag{3.29}$$

where Δt_{n+1} is the length of the (n+1)th time step, t^n is the actual time at the beginning of the (n+1)th time step and t^{n+1} is the actual time at the end of this time step. The time steps are chosen to discretize the time domain before a simulation just as a mesh (or "spatial steps") is chosen to discretize space. The time step length may vary from step to step.

The entire spatially integrated governing equation, (3.28), is evaluated at the end of each time step, $t = t^{n+1}$. The time derivative of head in (2.28) is approximated, using a finite-difference approximation, as the change in head over a time step, divided by the time step length:

$$\frac{dh_i}{dt} = \frac{h_i\left(t^n + \Delta t_{n+1}\right) - h_i\left(t^n\right)}{\Delta t_{n+1}} \tag{3.30}$$

In order to simplify the notation, the head at the end of the time step, $h_i(t^n + \Delta t_{n+1})$ is denoted h_i^{n+1}, and the head at the beginning of the time step $h_i(t^n)$ is denoted h_i^n. Thus,

$$\frac{dh_i}{dt} = \frac{h_i^{n+1} - h_i^n}{\Delta t_{n+1}} \tag{3.31}$$

The parameters that depend on time in (3.28), $v_i(t)$ and $I_{ij}(t)$, are also evaluated at the time, t^{n+1}, at the end of a time step:

$$h_j(t)\Big|_{t^{n+1}} = h_j^{n+1} \tag{3.32a}$$

$$V_i(t)\Big|_{t^{n+1}} = V_i^{n+1} \tag{3.32b}$$

$$I_{ij}(t)\Big|_{t^{n+1}} = I_{ij}^{n+1} \tag{3.32c}$$

The sources, Q_{IN_i} and Q_i*, are assumed constant in time for the present example.

Resolution of nonlinearities

The variability in time of cell volume, V_i, and the integral, I_{ij}, depends on the changing thickness of the aquifer with time, B(x,y,t). The aquifer thickness at node i at the end of a time step, B_i^{n+1}, is not known until the head at the end of the time step is known, giving the water-table elevation. This typifies a nonlinear problem wherein the problem requires values of coefficients in order to be solved, but the values of these coefficients depend on the solution to be obtained. This circular problem is avoided in this example by using estimates of the coefficient values in the solution. An estimate of the head at the end of the next time step is obtained by a linear projection:

$$h_i^{proj} = h_i^n + \left(\frac{\Delta t_{n+1}}{\Delta t_n}\right)\left(h_i^n - h_i^{n-1}\right) \tag{3.33}$$

where h_i^{proj} is the projected or estimated head at the end of the as-yet-unsolved time step, which would have an exact value, h_i^{n+1}. Actually, in addition to projection, SUTRA also employs a simple iterative process to resolve nonlinearities. This is described in sections 4.4 and 4.5 under the subheading "Temporal discretization and iteration."

A projected thickness may then be determined from (3.33) as:

$$B_i^{n+1} \cong B_i^{proj} = h_i^{proj} - BASE_i \tag{3.34}$$

where B_i^{n+1} is the value of thickness needed to evaluate V_i^{n+1} and I_{ij}^{n+1}, B_i^{proj} is the estimated value of B_i^{n+1}, and $BASE_i$ is the value of BASE(x,y) at node i.

Now the spatially integrated equation, (3.28), may be written discretely in time:

$$S_i V_i^{n+1}\left(\frac{h_i^{n+1} - h_i^n}{\Delta t_{n+1}}\right) + \sum_{j=1}^{NN} I_{ij}^{n+1} h_j^{n+1} = Q_{IN_i} + Q_i^* \qquad i = \overline{1, NN} \tag{3.35}$$

where V_i^{n+1} and I_{ij}^{n+1} are evaluated based on projected thickness, B_i^{proj}.

3.5 Boundary Conditions and Solution of Discretized Equation

Matrix equation and solution sequence

The NN relations given by (3.35) may be rearranged and rewritten in matrix form:

$$\left(\frac{1}{\Delta t_{n+1}}\right)\begin{bmatrix} S_1V_1^{n+1} & 0 & 0 & \cdots & 0 \\ 0 & S_2V_2^{n+1} & 0 & \cdots & 0 \\ 0 & 0 & S_3V_3^{n+1} & & 0 \\ \vdots & \vdots & \vdots\ddots & & \vdots \\ 0 & 0 & 0 & \cdots & S_{NN}V_{NN}^{n+1} \end{bmatrix}\begin{Bmatrix} h_1^{n+1} \\ h_2^{n+1} \\ h_3^{n+1} \\ \vdots \\ h_{NN}^{n+1} \end{Bmatrix}$$

$$+\begin{bmatrix} I_{11}^{n+1} & I_{12}^{n+1} & I_{13}^{n+1} & I_{14}^{n+1} & . & . & . & . & I_{1,NN}^{n+1} \\ I_{21}^{n+1} & I_{22}^{n+1} & I_{23}^{n+1} & . & . & . & . & . & . \\ I_{31}^{n+1} & I_{32}^{n+1} & I_{33}^{n+1} & . & . & . & . & . & . \\ I_{41}^{n+1} & . & . & . & . & . & . & . & . \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ I_{NN,1}^{n+1} & . & . & . & . & . & . & . & I_{NN,NN}^{n+1} \end{bmatrix}\begin{Bmatrix} h_1^{n+1} \\ h_2^{n+1} \\ h_3^{n+1} \\ . \\ \vdots \\ h_{NN}^{n+1} \end{Bmatrix}$$

$$=\left(\frac{1}{\Delta t_{n+1}}\right)\begin{Bmatrix} S_1 & V_1^{n+1} & h_1^n \\ S_2 & V_2^{n+1} & h_2^n \\ S_3 & V_3^{n+1} & h_3^n \\ \vdots & & \\ S_{NN} & V_{NN}^{n+1} & h_{NN}^n \end{Bmatrix}+\begin{Bmatrix} Q_{IN_1} \\ Q_{IN_2} \\ Q_{IN_3} \\ \vdots \\ Q_{IN_{NN}}^* \end{Bmatrix}+\begin{Bmatrix} Q_1^* \\ Q_2^* \\ Q_3^* \\ \vdots \\ Q_{NN}^* \end{Bmatrix} \tag{3.36}$$

By adding the two matrices on the left side, and the vectors on the right side, a matrix equation is obtained which may be solved for the model heads at the new time level, t^{n+1}, on each time step:

$$
\begin{bmatrix}
\left(\frac{S_1V_1^{n+1}}{\Delta t_{n+1}}+I_{11}^{n+1}\right) & I_{12}^{n+1} & I_{13}^{n+1} & \cdots & I_{1,NN}^{n+1} \\
I_{21}^{n+1} & \left(\frac{S_2V_2^{n+1}}{\Delta t_{n+1}}+I_{22}^{n+1}\right) & I_{23}^{n+1} & \cdots & \vdots \\
I_{31}^{n+1} & I_{32}^{n+1} & \ddots & & \vdots \\
\vdots & \vdots & & \ddots & \vdots \\
I_{NN,1}^{n+1} & \cdots & \cdots & & \left(\frac{S_{NN}V_{NN}^{n+1}}{\Delta t_{n+1}}+I_{NN,NN}^{n+1}\right)
\end{bmatrix}
\begin{Bmatrix}
h_1^{n+1} \\ h_2^{n+1} \\ h_3^{n+1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ h_{NN}^{n+1}
\end{Bmatrix}
$$

$$
=\begin{Bmatrix}
\frac{S_1V_1^{n+1}h_1^n}{\Delta t_{n+1}} + Q_{IN_1} + Q_1^* \\
\frac{S_2V_2^{n+1}h_2^n}{\Delta t_{n+1}} + Q_{IN_2} + Q_2^* \\
\vdots \\
\vdots \\
\frac{S_{NN}V_{NN}^{n+1}h_{NN}^n}{\Delta t_{n+1}} + Q_{IN_{NN}} + Q_{NN}^*
\end{Bmatrix}
\tag{3.37}
$$

The solution progresses through time as follows: On a given time step, the nodal heads at the beginning of the step are known values and are placed in h_j^n in the right-hand-side vector of (3.37). The thickness-dependent values are determined based on the projection of B in (3.34) using the projected head of (3.33). The integrals and volumes are evaluated and the matrix and vector completed. The linear system of equations (3.37) is solved for the nodal heads at the end of the current time step using (banded) Gaussian elimination or an iterative sparse matrix equation solver. The new heads are then placed on the right side of (3.37) into h_j^n, and a new time step is begun.

Specification of boundary conditions

Before solving the matrix equation as described above, information about boundary conditions must be included. In the case of solving for heads, the boundary conditions take the form of either specified fluid fluxes across boundaries which are directly entered in the terms, Q_{IN_i}, or of particular head values specified at nodal locations. At a point of fixed head in an aquifer, a particular value of fluid inflow or outflow occurs at that point in order to keep the head constant when the aquifer is stressed. This is the flux of fluid that is added to the model aquifer in order to obtain fixed heads at nodes.

In order to illustrate how specified heads are implemented in SUTRA, consider the closed system of Figure 3.5 in which head at node i, h_i, is to have a specified value, h_{BC}, for all time. A well is removing water from the system at an internal node. A core of porous medium with conductance ν is connected to node i. The head outside the core is held at the specified value, h_{BC}. The head at node i, h_i, is calculated by the model. Under steady-state conditions, a flow of Q_{BC_i} [L^3/s] enters through the core at node i in order to balance the rate of fluid removal at the well. The resulting head at node i depends on the conductance value ν of the core. If ν is very small, then a large head drop is required across the core in order to supply fluid at the rate the pumping well requires. This results in h_i having quite a different value from h_{BC}. If, however, ν is very large, then the value of head at node i is very close to h_{BC}, as only a tiny head drop across the core supplies the fluid required by the well. Therefore, when the required flux is applied to a node through a highly conductive core, the outside of which is held at a specified head value, the node responds with a head value nearly equal to that specified. An advantage of specifying head this way is that when head at a node in the mesh is fixed, a calculation of the flux entering the mesh at this node is obtained at the same time:

$$Q_{BC_i} = \nu\left(h_{BC_i} - h_1^{n+1}\right) \tag{3.38}$$

where Q_{BC_i} is the inflow at node i resulting from the specified head boundary condition, ν is the conductance of the "core," and h_{BC_i} is the specified value of head at node i on the boundary.

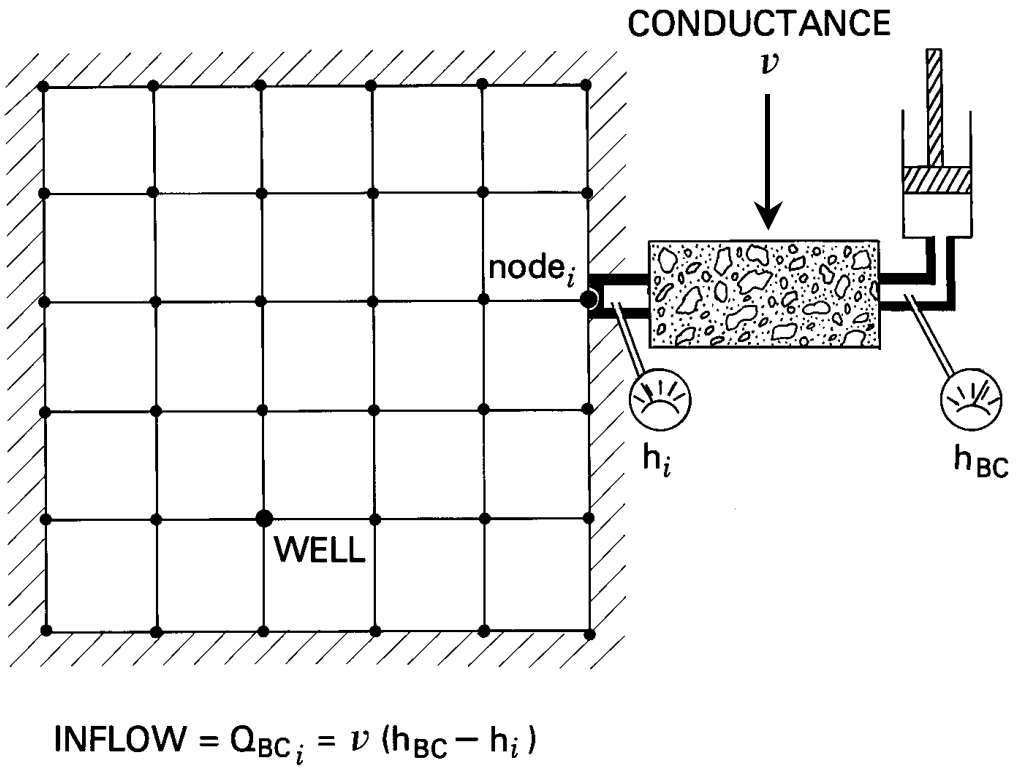


Figure 3.5. Schematic representation of specified head (or pressure) boundary condition. Specified concentration boundary conditions are implemented using an analogous construction.

The matrix equation (3.37) may be written in short form as:

$$\sum_{j=1}^{NN} M_{ij}^{n+1}\ h_j^{n+1} = \left(\frac{S_i\ V_i^{n+1}}{\Delta t_{n+1}}\right) h_i^n + Q_i^* + Q_{IN_i} + Q_{BC_i} \qquad i = \overline{1, NN} \tag{3.39}$$

wherein an additional flux Q_{BC_i} has been added only to the equations that represent the specified head nodes. At such a node, say node A, the equation is:

$$\sum_{j=1}^{NN} M_{Aj}^{n+1}\ h_j^{n+1} = \left(\frac{S_A\ V_A^{n+1}}{\Delta t_{n+1}}\right) h_A^n + Q_A^* + Q_{IN_A} + \nu\left(h_{BC_A} - h_A^{n+1}\right) \tag{3.40}$$

If ν is very large, then the last term dominates the equation and (3.40) becomes:

$$h_A^{n+1} \cong h_{BC_A} \tag{3.41}$$

Thus, the specified head is set at node A, but as h_A^{n+1} and h_{BC_A} are slightly different, a flux may be determined from (3.38).

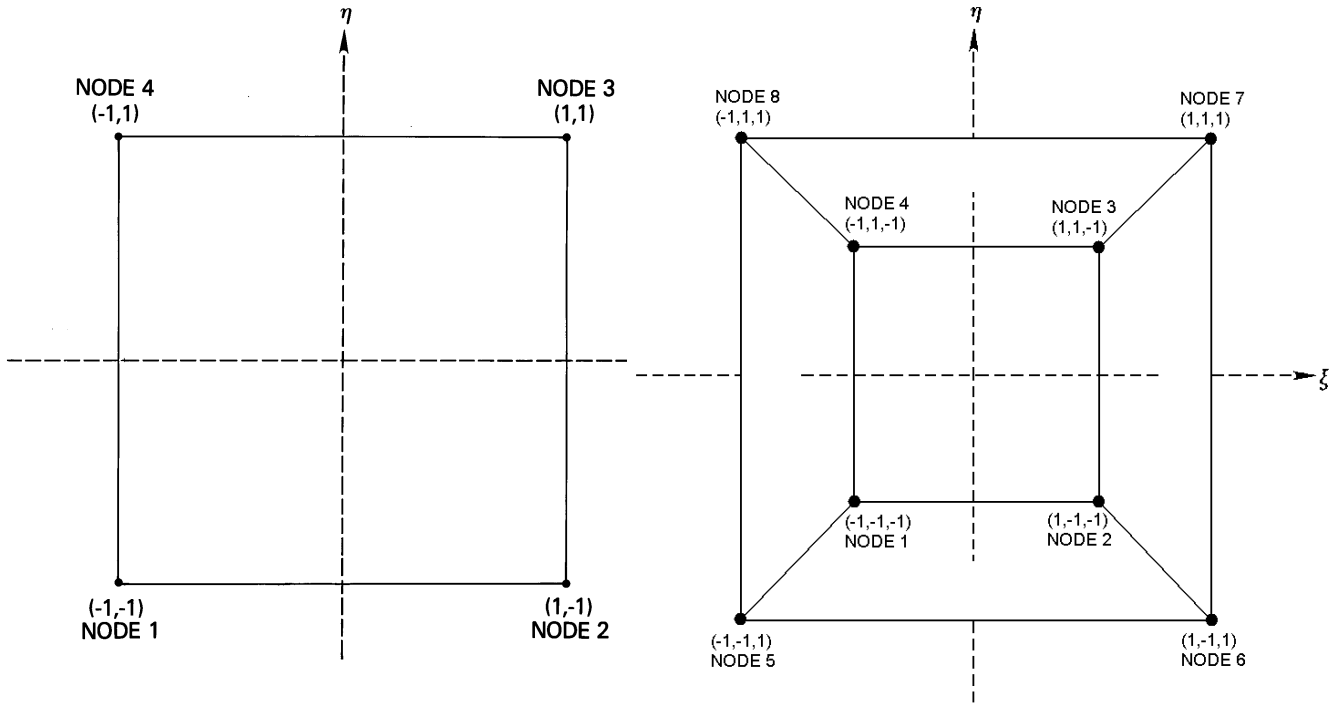
DETAILS OF SUTRA METHODOLOGY

Chapter 4: Numerical Methods

In this section, the numerical methods upon which SUTRA is based are presented in detail. The purpose of this presentation is to provide a complete reference for the computer code.

4.1 Basis and Weighting Functions

Basis functions, weighting functions and their derivatives are all described in local element geometry. In a 2D local coordinate system, every element takes the shape of a two-by-two square. The 2D local coordinates, ξ and η, are shown along with a generic local 2D finite element in Figure 4.1a. The origin of the local coordinate system is at the center of the element. Local node 1 always has local coordinates $(\xi, \eta) = (-1, -1)$. The other nodes are numbered counterclockwise from the first node as shown in Figure 4.1a.



(a) (b)

Figure 4.1. (a) Quadrilateral 2D finite element in local coordinate system (ξ, η). (b) Hexahedral 3D finite element in local coordinate system (ξ, η, ζ). The ζ-axis points directly out of the page.

The following one-dimensional (1D) basis functions are defined over the region of the 2D element:

$$\Xi_{-}(\xi) = \tfrac{1}{2}(1-\xi) \tag{4.1}$$

$$\Xi_{+}(\xi) = \tfrac{1}{2}(1+\xi) \tag{4.2}$$

$$H_{-}(\eta) = \tfrac{1}{2}(1-\eta) \tag{4.3}$$

$$H_{+}(\eta) = \tfrac{1}{2}(1+\eta) \tag{4.4}$$

These linear one-dimensional basis functions are continuous in ξ and η and range between zero and one as ξ and η range between +1 to –1. The one-dimensional functions are combined to create the bilinear basis functions used in 2D SUTRA simulations:

$$\Omega_1(\xi,\eta) = \Xi_{-} H_{-} \tag{4.5}$$

$$\Omega_2(\xi,\eta) = \Xi_{+} H_{-} \tag{4.6}$$

$$\Omega_3(\xi,\eta) = \Xi_{+} H_{+} \tag{4.7}$$

$$\Omega_4(\xi,\eta) = \Xi_{-} H_{+} \tag{4.8}$$

The 2D bilinear basis functions, when defined in the local element coordinate system, are denoted as $\Omega_i(\xi, \eta)$, i=1,2,3,4. There is one basis function defined for each node.

The basis function Ω_i, defined for node i, has a value of one at the node and a value of zero at the other nodes. The surface representing $\Omega_i(\xi, \eta)$ over an element is curved due to the product of ξ and η in equations (4.5) through (4.8). A trajectory in the surface parallel to an element side, however, is a perfectly straight line as shown in Figure 4.2. This is borne out in the derivatives of the bilinear basis functions, which depend on only one space coordinate:

$$\frac{\partial\Omega_1}{\partial\xi} = -\tfrac{1}{2} H_{-} \qquad \frac{\partial\Omega_1}{\partial\eta} = -\tfrac{1}{2}\Xi_{-} \tag{4.9}$$

$$\frac{\partial\Omega_2}{\partial\xi} = +\tfrac{1}{2} H_{-} \qquad \frac{\partial\Omega_2}{\partial\eta} = -\tfrac{1}{2}\Xi_{+} \tag{4.10}$$

$$\frac{\partial\Omega_3}{\partial\xi} = +\tfrac{1}{2} H_{+} \qquad \frac{\partial\Omega_3}{\partial\eta} = +\tfrac{1}{2}\Xi_{+} \tag{4.11}$$

$$\frac{\partial\Omega_4}{\partial\xi} = -\tfrac{1}{2} H_{+} \qquad \frac{\partial\Omega_4}{\partial\eta} = +\tfrac{1}{2}\Xi_{-} \tag{4.12}$$

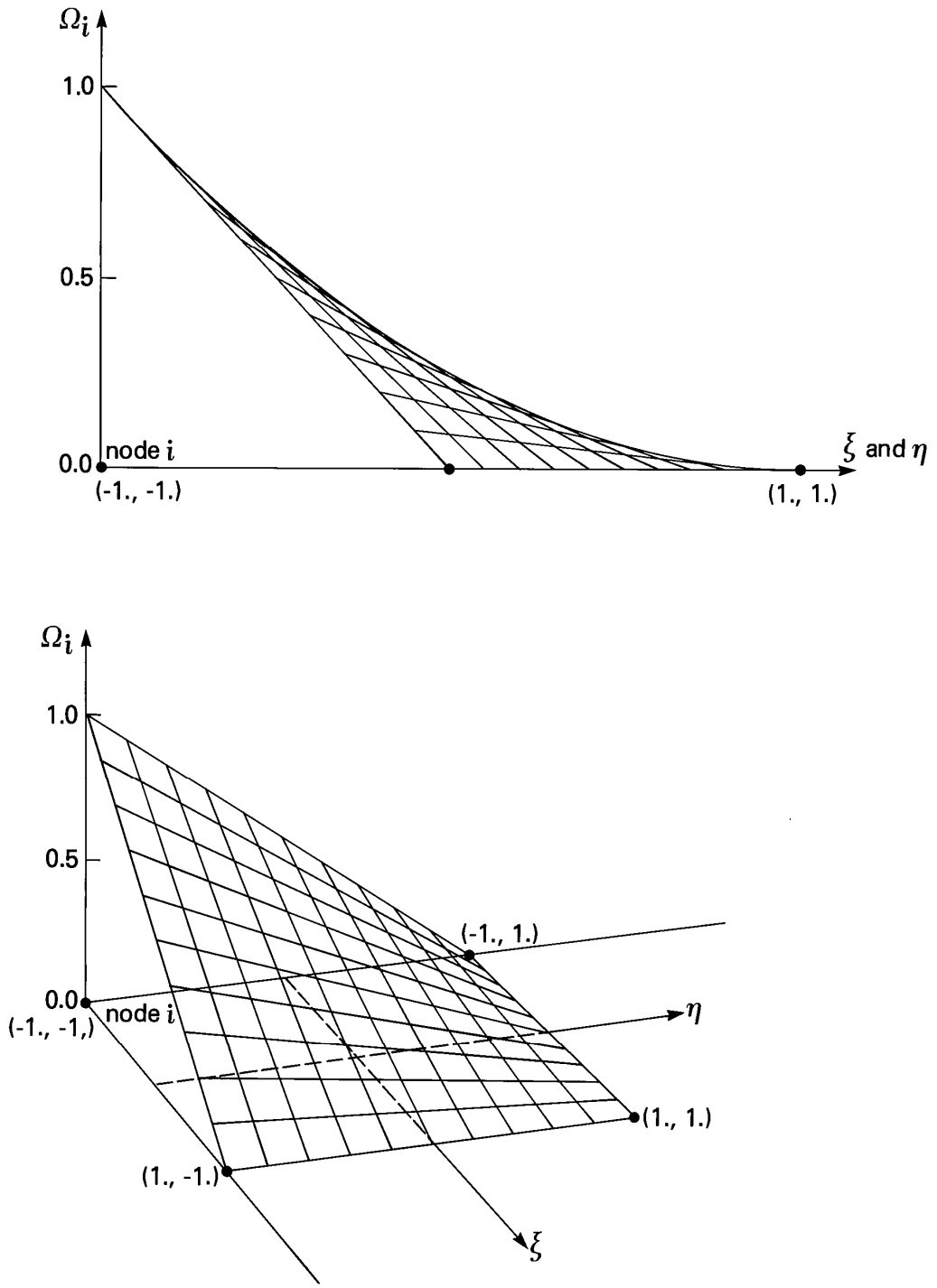


Figure 4.2. Perspectives of the 2D basis function $\Omega_i(\xi, \eta)$ at node i.

Asymmetric weighting functions are defined for use in a Galerkin-Petrov method (one version of which is described in Huyakorn and Pinder, 1983). These are not applied for nodewise discretization of parameters, but rather for weighting in the volume integrals of the governing equation. They may be used to give an "upstream weighting" to the advective flux term in the transport equations or to provide "upstream weighting" to the fluid flux term in the fluid mass balance when the medium is unsaturated. In 2D, the asymmetric functions are defined as follows:

$$\theta_1(\xi,\eta) = \left(\Xi_- - \Xi^*\right)\left(H_- - H^*\right) \tag{4.13}$$

$$\theta_2(\xi,\eta) = \left(\Xi_+ + \Xi^*\right)\left(H_- - H^*\right) \tag{4.14}$$

$$\theta_3(\xi,\eta) = \left(\Xi_+ + \Xi^*\right)\left(H_+ + H^*\right) \tag{4.15}$$

$$\theta_4(\xi,\eta) = \left(\Xi_- - \Xi^*\right)\left(H_+ + H^*\right) \tag{4.16}$$

where:

$$\Xi^* = 3a_\xi\,\Xi_-\,\Xi_+ \tag{4.17}$$

$$H^* = 3a_\eta\,H_-\,H_+ \tag{4.18}$$

The spatial derivatives are:

$$\frac{\partial\theta_1}{\partial\xi} = -\tfrac{1}{2}\left(1-3a_\xi\xi\right)\left(H_- - H^*\right) \qquad \frac{\partial\theta_1}{\partial\eta} = -\tfrac{1}{2}\left(1-3a_\eta\eta\right)\left(\Xi_- - \Xi^*\right) \tag{4.19}$$

$$\frac{\partial\theta_2}{\partial\xi} = +\tfrac{1}{2}\left(1-3a_\xi\xi\right)\left(H_- - H^*\right) \qquad \frac{\partial\theta_2}{\partial\eta} = -\tfrac{1}{2}\left(1-3a_\eta\eta\right)\left(\Xi_+ + \Xi^*\right) \tag{4.20}$$

$$\frac{\partial\theta_3}{\partial\xi} = +\tfrac{1}{2}\left(1-3a_\xi\xi\right)\left(H_+ + H^*\right) \qquad \frac{\partial\theta_3}{\partial\eta} = +\tfrac{1}{2}\left(1-3a_\eta\eta\right)\left(\Xi_+ + \Xi^*\right) \tag{4.21}$$

$$\frac{\partial\theta_4}{\partial\xi} = -\tfrac{1}{2}\left(1-3a_\xi\xi\right)\left(H_+ + H^*\right) \qquad \frac{\partial\theta_4}{\partial\eta} = +\tfrac{1}{2}\left(1-3a_\eta\eta\right)\left(\Xi_- - \Xi^*\right) \tag{4.22}$$

The parameters a_ξ and a_η determine the amount of asymmetry (or upstream weight) in each coordinate direction. When these parameters have a value of zero, then (4.13) through (4.22) reduce to the symmetric 2D basis functions and their derivatives, (4.5) through (4.12). The values of a_ξ and a_η depend on location in the element:

$$a_\xi(\xi,\eta) = (\mathrm{UP})\left(\frac{v_\xi}{|\underline{v}_{\mathrm{local}}|}\right) \tag{4.23}$$

$$a_\eta(\xi,\eta) = (\mathrm{UP})\left(\frac{v_\eta}{|\underline{v}_{\mathrm{local}}|}\right) \tag{4.24}$$

where UP is the fractional strength of upstream weighting desired (chosen by the model user), $v_\xi(\xi, \eta)$ and $v_\eta(\xi, \eta)$ are the components of fluid velocity given in terms of local element coordinates, and $|\underline{v}_{\mathrm{local}}(\xi, \eta)|$ is the magnitude of fluid velocity given in terms of local coordinates. Each velocity component may vary in value throughout the element. A description

of the calculation of fluid velocity is given in section 4.6, "Consistent Evaluation of Fluid Velocity."

Note that the 2D basis functions, weighting functions and their derivatives are calculated by the SUTRA subroutine "BASIS2".

In 3D SUTRA simulations, the basis functions depend on three local coordinates, ξ, η, and ζ. As in 2D, the origin of the local coordinate system is at the center of the element. The local node numbering in 3D is illustrated in Figure 4.1b. Local node one always has local coordinates $(\xi, \eta, \zeta) = (-1, -1, -1)$. The other nodes are numbered as follows, assuming a right-handed coordinate system. If the element is viewed from the outside, looking through one face, such that the element face farthest away has node 1 as its lower-left-hand corner, then nodes 2 – 4 are the remaining nodes in that element face, proceeding counterclockwise from node 1. Nodes 5 – 8 are then located on the nearest element face (the one being looked through and opposite the first face), such that nodes 5, 6, 7, and 8 are connected by element edges to nodes 1, 2, 3, and 4, respectively.

The following one-dimensional basis functions are defined over the region of the 3D element:

$$\Xi_-(\xi) = \tfrac{1}{2}(1-\xi) \tag{4.25}$$

$$\Xi_+(\xi) = \tfrac{1}{2}(1+\xi) \tag{4.26}$$

$$H_-(\eta) = \tfrac{1}{2}(1-\eta) \tag{4.27}$$

$$H_+(\eta) = \tfrac{1}{2}(1+\eta) \tag{4.28}$$

$$\Psi_-(\zeta) = \tfrac{1}{2}(1-\zeta) \tag{4.29}$$

$$\Psi_+(\zeta) = \tfrac{1}{2}(1+\zeta) \tag{4.30}$$

These linear one-dimensional basis functions are continuous in ξ, η, and ζ, and range between zero and one as ξ, η, and ζ range between +1 to –1. The one-dimensional functions are combined to create the trilinear basis functions used in 3D SUTRA simulations:

$$\Omega_1(\xi, \eta, \zeta) = \Xi_- H_- \Psi_- \tag{4.31}$$

$$\Omega_2(\xi, \eta, \zeta) = \Xi_+ H_- \Psi_- \tag{4.32}$$

$$\Omega_3(\xi, \eta, \zeta) = \Xi_+ H_+ \Psi_- \tag{4.33}$$

$$\Omega_4(\xi, \eta, \zeta) = \Xi_- H_+ \Psi_- \tag{4.34}$$

$$\Omega_5(\xi, \eta, \zeta) = \Xi_- H_- \Psi_+ \tag{4.35}$$

$$\Omega_6(\xi,\eta,\zeta) = \Xi_+ H_- \Psi_+ \tag{4.36}$$

$$\Omega_7(\xi,\eta,\zeta) = \Xi_+ H_+ \Psi_+ \tag{4.37}$$

$$\Omega_8(\xi,\eta,\zeta) = \Xi_- H_+ \Psi_+ \tag{4.38}$$

The 3D trilinear basis functions, when defined in the local element coordinate system, are denoted as $\Omega_i(\xi, \eta, \zeta)$, i=1,…,8. There is one basis function defined for each node.

The basis function Ω_i, defined for node i, has a value of one at the node and a value of zero at the other nodes. It varies linearly along the straight element edges that connect node i to its neighbors and has curvature in 3D analogous to that described above for 2D elements. The derivatives of the trilinear basis functions are as follows:

$$\frac{\partial\Omega_1}{\partial\xi} = -\tfrac{1}{2} H_- \Psi_- \qquad \frac{\partial\Omega_1}{\partial\eta} = -\tfrac{1}{2} \Xi_- \Psi_- \qquad \frac{\partial\Omega_1}{\partial\zeta} = -\tfrac{1}{2} \Xi_- H_- \tag{4.39}$$

$$\frac{\partial\Omega_2}{\partial\xi} = +\tfrac{1}{2} H_- \Psi_- \qquad \frac{\partial\Omega_2}{\partial\eta} = -\tfrac{1}{2} \Xi_+ \Psi_- \qquad \frac{\partial\Omega_2}{\partial\zeta} = -\tfrac{1}{2} \Xi_+ H_- \tag{4.40}$$

$$\frac{\partial\Omega_3}{\partial\xi} = +\tfrac{1}{2} H_+ \Psi_- \qquad \frac{\partial\Omega_3}{\partial\eta} = +\tfrac{1}{2} \Xi_+ \Psi_- \qquad \frac{\partial\Omega_3}{\partial\zeta} = -\tfrac{1}{2} \Xi_+ H_+ \tag{4.41}$$

$$\frac{\partial\Omega_4}{\partial\xi} = -\tfrac{1}{2} H_+ \Psi_- \qquad \frac{\partial\Omega_4}{\partial\eta} = +\tfrac{1}{2} \Xi_- \Psi_- \qquad \frac{\partial\Omega_4}{\partial\zeta} = -\tfrac{1}{2} \Xi_- H_+ \tag{4.42}$$

$$\frac{\partial\Omega_5}{\partial\xi} = -\tfrac{1}{2} H_- \Psi_+ \qquad \frac{\partial\Omega_5}{\partial\eta} = -\tfrac{1}{2} \Xi_- \Psi_+ \qquad \frac{\partial\Omega_5}{\partial\zeta} = +\tfrac{1}{2} \Xi_- H_- \tag{4.43}$$

$$\frac{\partial\Omega_6}{\partial\xi} = +\tfrac{1}{2} H_- \Psi_+ \qquad \frac{\partial\Omega_6}{\partial\eta} = -\tfrac{1}{2} \Xi_+ \Psi_+ \qquad \frac{\partial\Omega_6}{\partial\zeta} = +\tfrac{1}{2} \Xi_+ H_- \tag{4.44}$$

$$\frac{\partial\Omega_7}{\partial\xi} = +\tfrac{1}{2} H_+ \Psi_+ \qquad \frac{\partial\Omega_7}{\partial\eta} = +\tfrac{1}{2} \Xi_+ \Psi_+ \qquad \frac{\partial\Omega_7}{\partial\zeta} = +\tfrac{1}{2} \Xi_+ H_+ \tag{4.45}$$

$$\frac{\partial\Omega_8}{\partial\xi} = -\tfrac{1}{2} H_+ \Psi_+ \qquad \frac{\partial\Omega_8}{\partial\eta} = +\tfrac{1}{2} \Xi_- \Psi_+ \qquad \frac{\partial\Omega_8}{\partial\zeta} = +\tfrac{1}{2} \Xi_- H_+ \tag{4.46}$$

In 3D, the asymmetric functions are used to provide "upstream weighting" in a manner analogous to that described above for 2D. These functions are defined as follows:

$$\theta_1(\xi,\eta,\zeta) = \left(\Xi_- - \Xi^*\right)\left(\mathrm{H}_- - \mathrm{H}^*\right)\left(\Psi_- - \Psi^*\right) \tag{4.47}$$

$$\theta_2(\xi,\eta,\zeta) = \left(\Xi_+ + \Xi^*\right)\left(\mathrm{H}_- - \mathrm{H}^*\right)\left(\Psi_- - \Psi^*\right) \tag{4.48}$$

$$\theta_3(\xi,\eta,\zeta) = \left(\Xi_+ + \Xi^*\right)\left(\mathrm{H}_+ + \mathrm{H}^*\right)\left(\Psi_- - \Psi^*\right) \tag{4.49}$$

$$\theta_4(\xi,\eta,\zeta) = \left(\Xi_- - \Xi^*\right)\left(\mathrm{H}_+ + \mathrm{H}^*\right)\left(\Psi_- - \Psi^*\right) \tag{4.50}$$

$$\theta_5(\xi,\eta,\zeta) = \left(\Xi_- - \Xi^*\right)\left(\mathrm{H}_- - \mathrm{H}^*\right)\left(\Psi_+ + \Psi^*\right) \tag{4.51}$$

$$\theta_6(\xi,\eta,\zeta) = \left(\Xi_+ + \Xi^*\right)\left(\mathrm{H}_- - \mathrm{H}^*\right)\left(\Psi_+ + \Psi^*\right) \tag{4.52}$$

$$\theta_7(\xi,\eta,\zeta) = \left(\Xi_+ + \Xi^*\right)\left(\mathrm{H}_+ + \mathrm{H}^*\right)\left(\Psi_+ + \Psi^*\right) \tag{4.53}$$

$$\theta_8(\xi,\eta,\zeta) = \left(\Xi_- - \Xi^*\right)\left(\mathrm{H}_+ + \mathrm{H}^*\right)\left(\Psi_+ + \Psi^*\right) \tag{4.54}$$

where

$$\Xi^* = 3a_\xi\,\Xi_-\,\Xi_+ \tag{4.55}$$

$$\mathrm{H}^* = 3a_\eta\,\mathrm{H}_-\,\mathrm{H}_+ \tag{4.56}$$

$$\Psi^* = 3a_\zeta\,\Psi_-\,\Psi_+ \tag{4.57}$$

The spatial derivatives are as follows:

$$\frac{\partial\theta_1}{\partial\xi} = -\tfrac{1}{2}\left(1-3a_\xi\xi\right)\cdot\left(\mathrm{H}_- - \mathrm{H}^*\right)\cdot\left(\Psi_- - \Psi^*\right) \qquad \frac{\partial\theta_1}{\partial\eta} = -\tfrac{1}{2}\left(1-3a_\eta\eta\right)\cdot\left(\Xi_- - \Xi^*\right)\cdot\left(\Psi_- - \Psi^*\right) \qquad \frac{\partial\theta_1}{\partial\zeta} = -\tfrac{1}{2}\left(1-3a_\zeta\zeta\right)\cdot\left(\Xi_- - \Xi^*\right)\cdot\left(\mathrm{H}_- - \mathrm{H}^*\right) \tag{4.58}$$

$$\frac{\partial\theta_2}{\partial\xi} = +\tfrac{1}{2}\left(1-3a_\xi\xi\right)\cdot\left(\mathrm{H}_- - \mathrm{H}^*\right)\cdot\left(\Psi_- - \Psi^*\right) \qquad \frac{\partial\theta_2}{\partial\eta} = -\tfrac{1}{2}\left(1-3a_\eta\eta\right)\cdot\left(\Xi_- + \Xi^*\right)\cdot\left(\Psi_- - \Psi^*\right) \qquad \frac{\partial\theta_2}{\partial\zeta} = -\tfrac{1}{2}\left(1-3a_\zeta\zeta\right)\cdot\left(\Xi_- + \Xi^*\right)\cdot\left(\mathrm{H}_- - \mathrm{H}^*\right) \tag{4.59}$$

$$\frac{\partial\theta_3}{\partial\xi} = +\tfrac{1}{2}\left(1-3a_\xi\xi\right)\cdot\left(H_- + H^*\right)\cdot\left(\Psi_- - \Psi^*\right) \qquad \frac{\partial\theta_3}{\partial\eta} = +\tfrac{1}{2}\left(1-3a_\eta\eta\right)\cdot\left(\Xi_- + \Xi^*\right)\cdot\left(\Psi_- - \Psi^*\right) \qquad \frac{\partial\theta_3}{\partial\zeta} = -\tfrac{1}{2}\left(1-3a_\zeta\zeta\right)\cdot\left(\Xi_- + \Xi^*\right)\cdot\left(H_- + H^*\right) \tag{4.60}$$

$$\frac{\partial\theta_4}{\partial\xi} = -\tfrac{1}{2}\left(1-3a_\xi\xi\right)\cdot\left(H_- + H^*\right)\cdot\left(\Psi_- - \Psi^*\right) \qquad \frac{\partial\theta_4}{\partial\eta} = +\tfrac{1}{2}\left(1-3a_\eta\eta\right)\cdot\left(\Xi_- - \Xi^*\right)\cdot\left(\Psi_- - \Psi^*\right) \qquad \frac{\partial\theta_4}{\partial\zeta} = -\tfrac{1}{2}\left(1-3a_\zeta\zeta\right)\cdot\left(\Xi_- - \Xi^*\right)\cdot\left(H_- + H^*\right) \tag{4.61}$$

$$\frac{\partial\theta_5}{\partial\xi} = -\tfrac{1}{2}\left(1-3a_\xi\xi\right)\cdot\left(H_- - H^*\right)\cdot\left(\Psi_- + \Psi^*\right) \qquad \frac{\partial\theta_5}{\partial\eta} = -\tfrac{1}{2}\left(1-3a_\eta\eta\right)\cdot\left(\Xi_- - \Xi^*\right)\cdot\left(\Psi_- + \Psi^*\right) \qquad \frac{\partial\theta_5}{\partial\zeta} = +\tfrac{1}{2}\left(1-3a_\zeta\zeta\right)\cdot\left(\Xi_- - \Xi^*\right)\cdot\left(H_- - H^*\right) \tag{4.62}$$

$$\frac{\partial\theta_6}{\partial\xi} = +\tfrac{1}{2}\left(1-3a_\xi\xi\right)\cdot\left(H_- - H^*\right)\cdot\left(\Psi_- + \Psi^*\right) \qquad \frac{\partial\theta_6}{\partial\eta} = -\tfrac{1}{2}\left(1-3a_\eta\eta\right)\cdot\left(\Xi_- + \Xi^*\right)\cdot\left(\Psi_- + \Psi^*\right) \qquad \frac{\partial\theta_6}{\partial\zeta} = +\tfrac{1}{2}\left(1-3a_\zeta\zeta\right)\cdot\left(\Xi_- + \Xi^*\right)\cdot\left(H_- - H^*\right) \tag{4.63}$$

$$\frac{\partial\theta_7}{\partial\xi} = +\tfrac{1}{2}\left(1-3a_\xi\xi\right)\cdot\left(H_- + H^*\right)\cdot\left(\Psi_- + \Psi^*\right) \qquad \frac{\partial\theta_7}{\partial\eta} = +\tfrac{1}{2}\left(1-3a_\eta\eta\right)\cdot\left(\Xi_- + \Xi^*\right)\cdot\left(\Psi_- + \Psi^*\right) \qquad \frac{\partial\theta_7}{\partial\zeta} = +\tfrac{1}{2}\left(1-3a_\zeta\zeta\right)\cdot\left(\Xi_- + \Xi^*\right)\cdot\left(H_- + H^*\right) \tag{4.64}$$

$$\frac{\partial\theta_8}{\partial\xi} = -\tfrac{1}{2}\left(1-3a_\xi\xi\right)\cdot\left(H_- + H^*\right)\cdot\left(\Psi_- + \Psi^*\right) \qquad \frac{\partial\theta_8}{\partial\eta} = +\tfrac{1}{2}\left(1-3a_\eta\eta\right)\cdot\left(\Xi_- - \Xi^*\right)\cdot\left(\Psi_- + \Psi^*\right) \qquad \frac{\partial\theta_8}{\partial\zeta} = +\tfrac{1}{2}\left(1-3a_\zeta\zeta\right)\cdot\left(\Xi_- - \Xi^*\right)\cdot\left(H_- + H^*\right) \tag{4.65}$$

The parameters a_ξ, a_η, and a_ζ determine the amount of asymmetry (or upstream weight) in each coordinate direction. When these parameters have a value of zero, then (4.47) through (4.65) reduce to the symmetric 3D basis functions and their derivatives, (4.31) through (4.46). The values of a_ξ, a_η, and a_ζ depend on location in the element:

$$a_{\xi}(\xi,\eta,\zeta) = (\mathrm{UP})\left(\frac{v_{\xi}}{|\underline{v}_{\mathrm{local}}|}\right) \tag{4.66}$$

$$a_{\eta}(\xi,\eta,\zeta) = (\mathrm{UP})\left(\frac{v_{\eta}}{|\underline{v}_{\mathrm{local}}|}\right) \tag{4.67}$$

$$a_{\zeta}(\xi,\eta,\zeta) = (\mathrm{UP})\left(\frac{v_{\zeta}}{|\underline{v}_{\mathrm{local}}|}\right) \tag{4.68}$$

where UP is the fractional strength of upstream weighting desired (chosen by the model user), $v_{\xi}(\xi, \eta, \zeta)$, $v_{\eta}(\xi, \eta, \zeta)$, and $v_{\zeta}(\xi, \eta, \zeta)$ are the components of fluid velocity given in terms of local element coordinates, and $|\underline{v}_{\mathrm{local}}(\xi, \eta, \zeta)|$ is the magnitude of fluid velocity given in terms of local coordinates.

Note that the 3D basis functions, weighting functions and their derivatives are calculated by the SUTRA subroutine "BASIS3".

4.2 Coordinate Transformations

During calculations for the finite-element mesh and during integral evaluations, transformations are required between the global (x,y[,z]) coordinate system, in which an element may have an arbitrary size and quadrilateral (2D) or generalized hexahedral (3D) shape, and the local (ξ,η[,ζ]) coordinate system in which each element is a two-by-two square (2D) or two-by-two-by-two cube (3D). Transformations are required in each direction. The transformation employs the basis functions to provide a linear remapping in each coordinate direction. The Jacobian matrix [J] is calculated separately for each element that requires transformation and may vary from point to point in an element. In 3D, the Jacobian matrix is

$$[J] = \begin{bmatrix} \frac{\partial\Omega_1}{\partial\xi} & \frac{\partial\Omega_2}{\partial\xi} & \frac{\partial\Omega_3}{\partial\xi} & \frac{\partial\Omega_4}{\partial\xi} & \frac{\partial\Omega_5}{\partial\xi} & \frac{\partial\Omega_6}{\partial\xi} & \frac{\partial\Omega_7}{\partial\xi} & \frac{\partial\Omega_8}{\partial\xi} \\ \frac{\partial\Omega_1}{\partial\eta} & \frac{\partial\Omega_2}{\partial\eta} & \frac{\partial\Omega_3}{\partial\eta} & \frac{\partial\Omega_4}{\partial\eta} & \frac{\partial\Omega_5}{\partial\eta} & \frac{\partial\Omega_6}{\partial\eta} & \frac{\partial\Omega_7}{\partial\eta} & \frac{\partial\Omega_8}{\partial\eta} \\ \frac{\partial\Omega_1}{\partial\zeta} & \frac{\partial\Omega_2}{\partial\zeta} & \frac{\partial\Omega_3}{\partial\zeta} & \frac{\partial\Omega_4}{\partial\zeta} & \frac{\partial\Omega_5}{\partial\zeta} & \frac{\partial\Omega_6}{\partial\zeta} & \frac{\partial\Omega_7}{\partial\zeta} & \frac{\partial\Omega_8}{\partial\zeta} \end{bmatrix} \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \\ x_5 & y_5 & z_5 \\ x_6 & y_6 & z_6 \\ x_7 & y_7 & z_7 \\ x_8 & y_8 & z_8 \end{bmatrix} \tag{4.69}$$

The numbered subscripts refer to the local element numbering (which is shown in Figure 4.1).

The Jacobian matrix is used to transform derivatives of basis functions from the global to the local coordinate systems and the reverse:

$$\begin{Bmatrix} \frac{\partial \Omega_j}{\partial \xi} \\ \frac{\partial \Omega_j}{\partial \eta} \\ \frac{\partial \Omega_j}{\partial \zeta} \end{Bmatrix} = [J] \begin{Bmatrix} \frac{\partial \phi_j}{\partial x} \\ \frac{\partial \phi_j}{\partial y} \\ \frac{\partial \phi_j}{\partial z} \end{Bmatrix} \tag{4.70}$$

$$\begin{Bmatrix} \frac{\partial \phi_j}{\partial x} \\ \frac{\partial \phi_j}{\partial y} \\ \frac{\partial \phi_j}{\partial z} \end{Bmatrix} = [J^{-1}] \begin{Bmatrix} \frac{\partial \Omega_j}{\partial \xi} \\ \frac{\partial \Omega_j}{\partial \eta} \\ \frac{\partial \Omega_j}{\partial \zeta} \end{Bmatrix} \tag{4.71}$$

where:

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \tag{4.72}$$

The subscript j refers to any one of the eight nodes in a 3D element (four nodes in a 2D element), and ϕ_j refers to the global basis function as defined for the j^{th} node in an element. The same transformations apply to derivatives of the asymmetric weighting functions, which are denoted ω_j in global coordinates.

The equations above are presented in 3D. For 2D simulations, only terms involving x, y, ξ, η, and node subscripts 1 through 4 are relevant; the remaining terms should be left out to obtain the 2D forms.

In (4.71), $[J^{-1}]$ is the inverse Jacobian matrix, defined such that

$$[\ J^{-1}][\ J\ ] = [\ I\ ] \tag{4.73a}$$

where [I] is the identity matrix, whose diagonal elements are all equal to one and whose off-diagonal elements are all equal to zero. In 2D, $[J^{-1}]$ takes the form

$$[\ J^{-1}] = \left(\frac{1}{\det J} \right) \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix} \tag{4.73b}$$

where det J is the determinant of the Jacobian, given in 2D by

$$\det J = J_{11}\, J_{22} - J_{12}\, J_{21} \tag{4.74}$$

The determinant may vary bilinearly over a 2D element. A detailed discussion of inverse matrices and matrix determinants in 2D and 3D can be found in most linear algebra texts.

In 2D, differential elements of area, dA, are transformed between local and global coordinate systems as

$$dA = dx\, dy = \left(\det J\right) d\xi\, d\eta \tag{4.75}$$

In 3D, differential elements of volume, dV, are transformed as

$$dV = dx\, dy\, dz = \left(\det J\right) d\xi\, d\eta\, d\zeta \tag{4.76}$$

Note that the Jacobian matrix, the determinant of the Jacobian and the derivatives of the basis functions in local and global coordinates are calculated in SUTRA subroutine "BASIS2" for 2D simulations and in subroutine "BASIS3" for 3D simulations.

4.3 Gaussian Integration

Gaussian integration is a method by which exact integration of polynomials may be carried out through a simple summation of point values of the integrand. The method is:

$$\int_{\tau=-1}^{\tau=+1} f(\tau)\, d\tau = \sum_{KG=1}^{NP} G_{KG}\, f(\tau_{KG}) \tag{4.77}$$

where $f(\tau)$ is the function to be integrated between $\tau = -1$ and $\tau = +1$. KG is the Gauss point number, NP is the total number of Gauss points, G_{KG} is a constant, and τ_{KG} is the location of the KG^{th} Gauss point. An exact integration is guaranteed by the sum in (4.77) if n Gauss points are used for a polynomial $f(\tau)$ of order (2n–1). For evaluation of integrals that arise in the SUTRA methodology, only two Gauss points are used in a given coordinate direction, as the integrands encountered are usually of order three or less. In this case, the constants, G_{KG} have a value of one and (4.77) simplifies to:

$$\int_{\tau=-1}^{\tau=+1} f(\tau)\, d\tau = \sum_{KG=1}^{2} f(\tau_{KG}) \tag{4.78}$$

The values of τ_{KG} for the two Gauss points are $\pm 3^{-1/2}$ ($\approx \pm 0.577350269189626$).

In 2D, the need to define a two-by-two element in local coordinates is apparent here. Gaussian integration is done over a range of two, from –1 to +1. In order to integrate a term of the differential governing equation over an arbitrary quadrilateral element in the mesh, the limits of the integral must first be transformed to values of –1 and +1, that is, to local coordinates. When integrating a double integral over x and y, both integrals must be transformed to have limits of –1

and +1, and two Gauss points are needed in each coordinate direction. These are defined as shown in Figure 4.3.

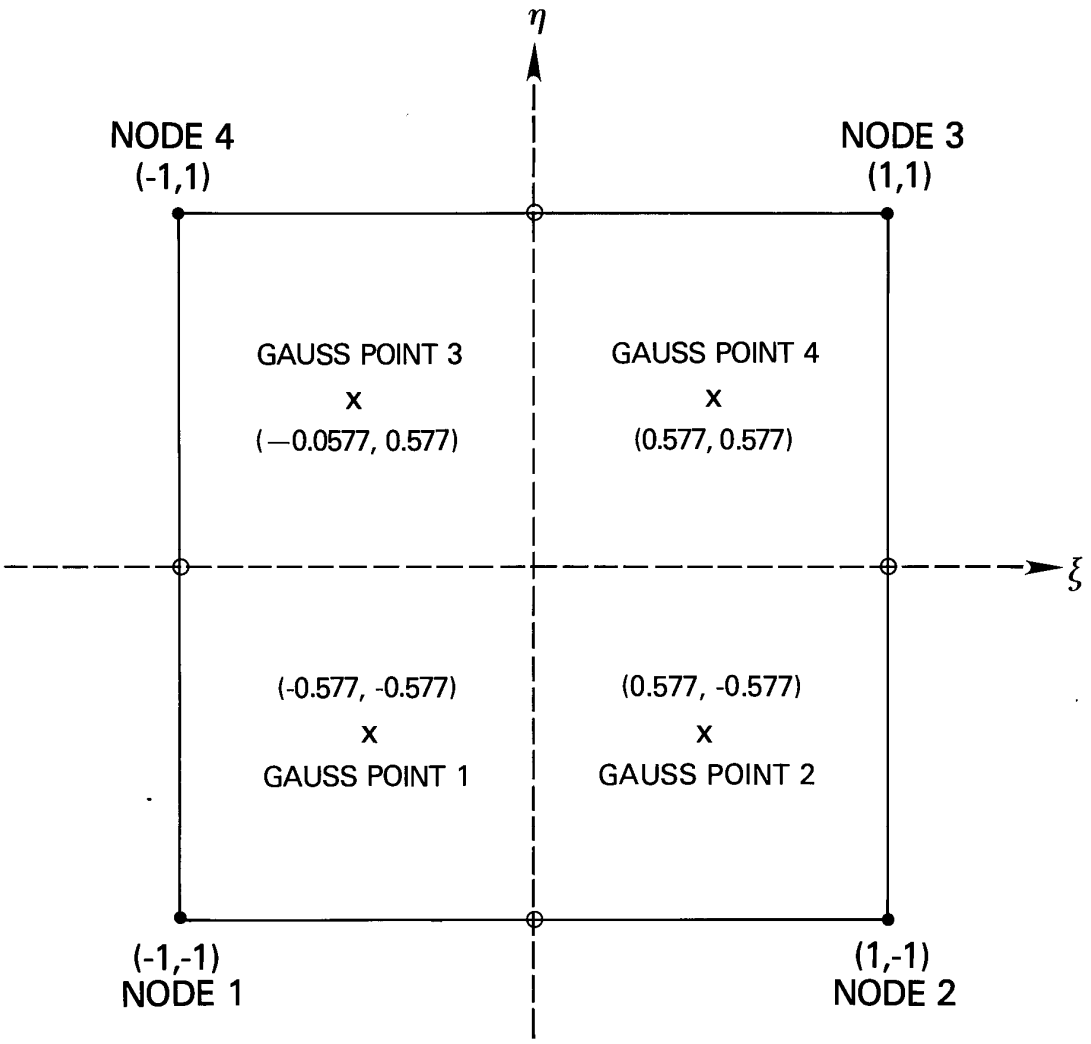


Figure 4.3. 2D finite element in local coordinate system with Gauss points.

A 2D example, evaluating the integral of (3.24), follows. The integral to evaluate is

$$A_{ij} = \int_{x_L} \int_{y_L} \left(\underline{\nabla}\phi_j \cdot \underline{\nabla}\phi_i \right) B_i \, dy \, dx \tag{4.79}$$

where x_L and y_L indicate that the integral is over the area of an element L in global coordinates. First, the (x,y) integral is converted to an integral in local coordinates (ξ, η) through use of the Jacobian:

$$A_{ij} = \int_{\xi=-1}^{+1} \int_{\eta=-1}^{+1} \left(\underline{\nabla}\phi_j \cdot \underline{\nabla}\phi_i \right) B_i \left(\det J \right) d\eta \, d\xi \tag{4.80}$$

The values of $\underline{\nabla}\phi$ are in global coordinates and are obtained by transformation of derivatives calculated in local coordinates.

Gaussian integration is applied independently to each integral:

$$A_{ij} = \sum_{K_\xi=1}^{2}\sum_{K_\eta=1}^{2}\left[\left(\underline{\nabla}\phi_j \cdot \underline{\nabla}\phi_i\right) B_i \left(\det J\right)\right] \Big|_{\left(\xi_{K_\xi}, \eta_{K_\eta}\right)} \tag{4.81}$$

or equivalently as a single summation:

$$A_{ij} = \sum_{KG=1}^{4}\left[\left(\underline{\nabla}\phi_j \cdot \underline{\nabla}\phi_i\right) B_i \left(\det J\right)\right] \Big|_{\left(\xi_{KG}, \eta_{KG}\right)} \tag{4.82}$$

where K_ξ and K_η refer to Gauss point locations in the ξ and η directions, and where ξ_{KG} and η_{KG} refer to the four Gauss points arising in (4.81) as depicted in Figure 4.3. Thus, in order to evaluate the integral (4.79) over a given element, only four values of the integrand need to be summed as given in (4.82), with one value determined at each of the four Gauss points.

In the case where a 2D element is a nonrectangular quadrilateral with variable thickness B, the polynomial to be integrated in (4.80) is of fourth order as each of the terms may vary linearly in the same direction. Otherwise, it is always of third order or less, and two-point Gauss integration provides exact results.

The procedure for integration in 3D is analogous to that in 2D. When integrating a triple integral over x, y, and z, all three integrals must be transformed to have limits of –1 and +1, and two Gauss points are needed in each of the three coordinate directions, making a total of 8 Gauss points.

Note that the summation over the Gauss points (as in (4.82)) is carried out by SUTRA subroutine "ELEMN2" in 2D and subroutine "ELEMN3" in 3D for each element in the mesh and for each integral that requires evaluation.

4.4 Numerical Approximation of SUTRA Fluid Mass Balance

The governing equation representing the SUTRA fluid mass balance, (2.24), is modified by the addition of a point source term which is used to insert points at which pressure is specified. This is done as described in text referring to relation (3.38).

$$O_p(p,U) = \left(S_w \rho S_{op} + \varepsilon\rho\frac{\partial S_w}{\partial p}\right)\frac{\partial p}{\partial t} + \left(\varepsilon S_w \frac{\partial \rho}{\partial U}\right)\frac{\partial U}{\partial t} - \underline{\nabla}\cdot\left[\left(\frac{\underline{\underline{k}}\, k_r \rho}{\mu}\right)\cdot\left(\underline{\nabla}p - \rho\underline{g}\right)\right] - Q_p - \nu_p\left(p_{BC} - p\right) = 0 \tag{4.83}$$

The last term is the source term arising from a specified pressure condition, wherein ν_p is a "conductance" and $p_{BC}(t)$ is the externally specified pressure boundary condition value. When ν_p is set to a sufficiently large value, the last term becomes much larger than the others in (4.83),

and $p \approx p_{BC}$, which is the desired boundary condition. Relation (4.83) is numerically approximated in the following sections.

Spatial integration

When the expression for $O_p(p,U)$ is approximated through nodewise, elementwise and cellwise discretizations, it no longer exactly equals zero. The approximate expression, $\langle\langle O_p(p,U)\rangle\rangle$, equals a spatially varying residual, $R_p(x,y[,z],t)$, as shown for a 2D example in (3.8). A weighted residual formulation may be written as:

$$\int_V \left\langle\left\langle O_p(p,U)\right\rangle\right\rangle W_i(x,y[,z])\,dV = 0 \qquad\qquad i = \overline{1,NN} \tag{4.84}$$

where $W_i(x,y[,z])$ is the weighting function in global coordinates chosen to be either the basis function, $\phi_i(x,y[,z])$ or the asymmetric weighting function, $\omega_i(x,y[,z])$, depending on the term of the equation. Relation (4.83) is approximated discretely and substituted for $\langle\langle O_p(p,U)\rangle\rangle$ in (4.84). The resulting set of integral terms is evaluated, one term at a time, in the following paragraphs.

The first term is an integral of the pressure derivative:

$$\int_V \left[\left\langle\left\langle S_w \rho S_{op} + \varepsilon\rho \frac{\partial S_w}{\partial p}\right\rangle\right\rangle \frac{\partial p}{\partial t}\right] \phi_i(x,y[,z])\,dV \tag{4.85}$$

where the term in square brackets is discretized cellwise, with one value of the term for each of the NN cells in the mesh, and the weighting function is chosen to be the basis function (written in global coordinates). The double angle brackets $\langle\langle\ \rangle\rangle$ surrounding a term indicate that it has been approximated in one of the three ways. Because the cellwise-approximated term is constant for a node i, it is removed from the integral, leaving only the basis function to be integrated. The volume integral of $\phi_i(x,y[,z])$ gives the volume V_i of cell i, which reduces to (3.15) for 2D simulations. The term (4.85) becomes:

$$\left(S_w \rho S_{op} + \varepsilon\rho \frac{\partial S_w}{\partial p}\right)_i \frac{\partial p_i}{\partial t} V_i \tag{4.86}$$

The second term of the expanded form of (4.84) is also a time derivative, which is approximated cellwise:

$$\int_V \left[\left\langle\left\langle \varepsilon S_w \frac{\partial \rho}{\partial U}\right\rangle\right\rangle \frac{\partial U}{\partial t}\right] \phi_i(x,y[,z])\,dV = \left(\varepsilon S_w \frac{\partial \rho}{\partial U}\right)_i \frac{\partial U_i}{\partial t} V_i \tag{4.87}$$

The third term of expanded relation (4.84), which involves the divergence of fluid flux, is weighted with the asymmetric function. The asymmetry is intended for use only in unsaturated flow problems to maintain solution stability when the mesh has not been designed fine enough to represent sharp saturation fronts. In general, the usual symmetric function is used for weighting this flux term even for unsaturated flow, but the term is developed with the asymmetric function in order to provide generality. Green's Theorem (3.17) is applied, yielding:

$$
\begin{aligned}
&-\int_V \left[\underline{\nabla} \cdot \left\langle\!\left\langle \left(\frac{\underline{\underline{k}}\, k_r \rho}{\mu} \right) \cdot \left(\underline{\nabla} p - \rho \underline{g} \right) \right\rangle\!\right\rangle \right] \omega_i \left(x, y[,z] \right) dV \\
&= -\int_\Gamma \left\langle\!\left\langle \left(\frac{\underline{\underline{k}}\, k_r \rho}{\mu} \right) \cdot \left(\underline{\nabla} p - \rho \underline{g} \right) \right\rangle\!\right\rangle \cdot \underline{n}\, \omega_i \left(x, y[,z] \right) d\Gamma + \int_V \left\langle\!\left\langle \left(\frac{\underline{\underline{k}}\, k_r \rho}{\mu} \right) \cdot \left(\underline{\nabla} p - \rho \underline{g} \right) \right\rangle\!\right\rangle \cdot \underline{\nabla} \omega_i \, dV
\end{aligned}
\tag{4.88}
$$

wherein the terms in double angle brackets are approximated discretely as described below, $\underline{n}$ is the unit outward normal to the 3D surface bounding the region to be simulated, and Γ is the surface of the region. The asymmetric weighting function in global (rather than local) coordinates is denoted, $\omega_i(x,y[,z])$. The first term on the right of (4.88) is exactly the fluid mass flux (see Darcy's law, relation (2.19)) out across the region's boundary at node i, $q_{OUT_i}(t)$ [M/s]:

$$
q_{OUT_i}(t) = \int_\Gamma \left\langle\!\left\langle \varepsilon \rho \underline{v} \cdot \underline{n} \right\rangle\!\right\rangle \omega_i \, d\Gamma = -\int_\Gamma \left[\left\langle\!\left\langle \left(\frac{\underline{\underline{k}}\, k_r \rho}{\mu} \right) \cdot \left(\underline{\nabla} p - \rho \underline{g} \right) \right\rangle\!\right\rangle \cdot \underline{n}\, \omega_i \, d\Gamma \right. \tag{4.89}
$$

This term is used to specify fluid flows across boundaries in SUTRA. Note that an inflow, $q_{IN_i}(t)$, is $q_{IN_i} = -q_{OUT_i}$.

The second term on the right of (4.88) is approximated using a combination of elementwise and nodewise discretizations. The approximation of $(\underline{\nabla} p - \rho \underline{g})$ requires particular attention and is discussed in section 4.6, "Consistent Evaluation of Fluid Velocity." The permeability tensor appearing in (4.88) has nine components in 3D:

$$
\left\langle\!\left\langle \underline{\underline{k}}^L \right\rangle\!\right\rangle = \begin{bmatrix} k_{xx}^L & k_{xy}^L & k_{xz}^L \\ k_{yx}^L & k_{yy}^L & k_{yz}^L \\ k_{zx}^L & k_{zy}^L & k_{zz}^L \end{bmatrix} \tag{4.90}
$$

wherein $\underline{\underline{k}}$ is discretized elementwise as indicated by $\left\langle\!\left\langle \underline{\underline{k}}^L \right\rangle\!\right\rangle$. In 2D simulations, $(\underline{\nabla} p - \rho \underline{g})$ is always zero in the third spatial direction, and only the four components of the permeability tensor that do not involve the z-coordinate are relevant. The pressure is discretized nodewise:

$$
p(x, y[,z], t) \cong \sum_{i=1}^{NN} p_i(t) \; \phi_i(x, y[,z]) \tag{4.91}
$$

Relative permeability, k_r, depends on saturation, which in turn depends on pressure. Relative permeabilities are evaluated at each Gauss point during numerical integration depending on the saturation (and pressure) at the Gauss point. Viscosity is evaluated at each Gauss point for energy transport as a function of nodewise discretized temperature, and is constant for solute transport.

Density, ρ, when it appears in the permeability term, is also evaluated at each Gauss point depending on the nodewise discretized value of U at the Gauss point. The density appearing in

product with the gravity term is expressly not evaluated in this usual manner. A particular discretization is used to maintain consistency with the $\underline{\nabla}p$ term, as described in section 4.6, "Consistent Evaluation of Fluid Velocity". This consistently evaluated $\rho\,\underline{g}$ term is denoted $\langle\langle \rho\,\underline{g} \rangle\rangle^*$ (see relation (4.153)).

The second term on the right of (4.88) is thus approximated as:

$$\sum_{j=1}^{NN} p_j(t) \int_V \left\{ \left[\left\langle\left\langle \underline{\underline{k}}^L \right\rangle\right\rangle \left\langle\left\{ \frac{k_r \rho}{\mu} \right\}\right\rangle \right] \cdot \underline{\nabla}\phi_j \right\} \cdot \underline{\nabla}\omega_i \; dV - \int_V \left\{ \left[\left\langle\left\langle \underline{\underline{k}}^L \right\rangle\right\rangle \left\langle\left\{ \frac{k_r \rho}{\mu} \right\}\right\rangle \right] \cdot \left\langle\left\langle \rho \underline{g} \right\rangle\right\rangle^* \right\} \cdot \underline{\nabla}\omega_i \; dV \tag{4.92}$$

where $\langle\langle \underline{\underline{k}}^L \rangle\rangle$ indicates an elementwise discretized permeability tensor, $\left\langle\left\{ \frac{k_r \rho}{\mu} \right\}\right\rangle$ indicates the value of the term based on nodewise discretized values of p and U (which is not the same as discretizing the term in a nodewise manner), and $\langle\langle \rho\,\underline{g} \rangle\rangle^*$ indicates a discretization of $(\rho\,\underline{g})$ consistent with the discretization of $\underline{\nabla}p$.

The last two terms of (4.83) are approximated cellwise with a basis function for weighting:

$$-\int_V \left\langle\left\langle Q_p \right\rangle\right\rangle \phi_i(x, y[, z])\, dV - \int \left\langle\left\langle \nu_p (p_{BC} - p) \right\rangle\right\rangle \phi_i(x, y[, z])\, dV = -Q_i - \nu_{p_i}\left(p_{BC_i} - p_i\right) \tag{4.93}$$

The cellwise discretizations that are employed in the above evaluations are

$$\left\langle\left\langle Q_p \right\rangle\right\rangle = \sum_{i=1}^{NN} \left(\frac{Q_i}{V_i} \right) \tag{4.94}$$

$$\left\langle\left\langle Q_{PBC} \right\rangle\right\rangle = \left\langle\left\langle \nu_p (p_{BC} - p) \right\rangle\right\rangle = \sum_{i=1}^{NN} \left[\left(\frac{\nu_{p_i}}{V_i} \right) \left(p_{BC_i} - p_i\right) \right] \tag{4.95}$$

where V_i is the volume of cell i, $Q_i(t)$ [M/s] is the total mass source to cell i, Q_{PBC} ($M/L^3 \cdot s$) is the fluid mass source rate due to the specified pressure, and ν_{p_i} [L·s] is the pressure-based conductance for the specified pressure source in cell i. The conductance is set to zero for nodes at which pressure is not specified (so that $Q_{PBC}=0$) and to a high value at nodes where pressure is specified.

By combining and rearranging the evaluations of approximate terms of (4.84), the following weighted residual relation is obtained in 3D:

$$AF_i \frac{dp_i}{dt} + CF_i \frac{dU_i}{dt} + \sum_{j=1}^{NN} p_j(t)\, BF_{ij} + \nu_{p_i} p_i = Q_i + \nu_{p_i} p_{BC_i} + q_{IN_i} + DF_i \quad i = \overline{1, NN} \tag{4.96}$$

where:

$$AF_i = \left(S_w \rho S_{op} + \varepsilon \rho \frac{\partial S_w}{\partial p} \right)_i V_i \tag{4.97}$$

$$CF_i = \left(\varepsilon S_w \frac{\partial \rho}{\partial U} \right)_i V_i \tag{4.98}$$

$$BF_{ij} = \int_x \int_y \int_z \left\{ \left[\left\langle \left\langle \underline{\underline{k}}^L \right\rangle \right\rangle \left\langle \left\langle \left\{ \frac{k_r \rho}{\mu} \right\} \right\rangle \right\rangle \right] \cdot \underline{\nabla} \phi_j \right\} \cdot \underline{\nabla} \omega_i \, dz \, dy \, dx \tag{4.99}$$

$$DF_i = \int_x \int_y \int_z \left\{ \left[\left\langle \left\langle \underline{\underline{k}}^L \right\rangle \right\rangle \left\langle \left\langle \left\{ \frac{k_r \rho}{\mu} \right\} \right\rangle \right\rangle \right] \cdot \left\langle \left\langle \rho \underline{g} \right\rangle \right\rangle^* \right\} \cdot \underline{\nabla} \omega_i \, dz \, dy \, dx \tag{4.100}$$

For 2D simulations, (4.99) and (4.100) are written as

$$BF_{ij} = \int_x \int_y \left\{ \left[\left\langle \left\langle \underline{\underline{k}}^L \right\rangle \right\rangle \left\langle \left\langle \left\{ \frac{k_r \rho}{\mu} \right\} \right\rangle \right\rangle \right] \cdot \underline{\nabla} \phi_j \right\} \cdot \underline{\nabla} \omega_i \, B \, dy \, dx \tag{4.101}$$

$$DF_i = \int_x \int_y \left\{ \left[\left\langle \left\langle \underline{\underline{k}}^L \right\rangle \right\rangle \left\langle \left\langle \left\{ \frac{k_r \rho}{\mu} \right\} \right\rangle \right\rangle \right] \cdot \left\langle \left\langle \rho \underline{g} \right\rangle \right\rangle^* \right\} \cdot \underline{\nabla} \omega_i \, B \, dy \, dx \tag{4.102}$$

The thickness of the mesh, B(x,y), is evaluated at each Gauss point according to a nodewise discretization:

$$B(x,y) \cong \sum_{i=1}^{NN} B_i \, \phi_i(x,y) \tag{4.103}$$

where B_i is the mesh thickness at node i. Note that mesh thickness is fixed and may not vary in time as was allowed for illustrative purposes in Chapter 3, "Fundamentals of Numerical Algorithms."

The only integrals requiring Gaussian integration are BF_{ij} and DF_i. Note that these are evaluated in SUTRA subroutine ELEMN2 (for 2D) or ELEMN3 (for 3D) in an element-by-element manner. The other terms, except for those involving ν_{p_i}, are evaluated cellwise (one for each node) by subroutine NODAL. The specified pressure terms are evaluated by subroutine BC.

Temporal discretization and iteration

The time derivatives in the spatially discretized and integrated equation are approximated by finite differences. The pressure term is approximated as:

$$\frac{dp_i}{dt} \cong \frac{p_i^{n+1} - p_i^n}{\Delta t_{n+1}} \tag{4.104}$$

where

$$p_i^n = p_i\left(t^n\right) \tag{4.105a}$$

$$p_i^{n+1} = p_i\left(t^n + \Delta t_{n+1}\right) = p_i\left(t^{n+1}\right) \tag{4.105b}$$

and

$$\Delta t_{n+1} = t^{n+1} - t^n \tag{4.106}$$

The new or current time step, Δt_{n+1}, begins at time t^n and ends at time t^{n+1}. The previous time step, for which a solution has already been obtained at time t^n, is denoted at Δt_n.

The term in (4.96) involving the time derivative of concentration or temperature, dU/dt, makes only a very small contribution to the fluid mass balance. For solution over the present time step, Δt_{n+1}, this derivative is evaluated using information from the previous time step, as these values are already known:

$$\frac{dU_i}{dt} \cong \left(\frac{dU_i}{dt}\right)^n = \frac{U_i^n - U_i^{n-1}}{\Delta t_n} \tag{4.107}$$

This approximation gives a simple method of accounting for this small contribution to the fluid mass balance.

All other terms in (4.96) are evaluated at the new time level t^{n+1} for solution of the present time step, Δt_{n+1}, except for the density in the consistently discretized $\langle\langle \rho \underline{g} \rangle\rangle^*$ term. The density is evaluated based on $U(t^n)$, the value of U at the beginning of the present time step. Because coefficients depend on the yet unknown values of p and U at the end of the time step, one or more iterations may be used to solve this nonlinear problem. On the first iteration, or when only one iteration per time step is used, coefficients are based on a projected value of p and U.

$$p_i^{proj} = p_i^n + \left(\frac{\Delta t_{n+1}}{\Delta t_n}\right)\left(p_i^n - p_i^{n-1}\right) \tag{4.108}$$

$$U_i^{proj} = U_i^n + \left(\frac{\Delta t_{n+1}}{\Delta t_n}\right)\left(U_i^n - U_i^{n-1}\right) \tag{4.109}$$

These projections estimate the p and U values at a node i, p_i^{proj} and U_i^{proj}, at the end of the present time step, Δt_{n+1}, based on linear extrapolation of the two previous values of p and U. All p and U dependent coefficients (except $\langle\langle \rho \underline{g} \rangle\rangle^*$) in (4.96) through (4.102) are estimated at time level t^{n+1}. These coefficient values are based on the most recent values of p and U, be they projections or solutions to the previous iteration. Iterations end when the maximum change in p and U at any node in the mesh falls below user-specified criteria of absolute change in p and U.

The weighted residual relations (4.96) may thus be written in a form which allows for solution of pressures at nodes, p_i^{n+1}, at the end of the present time step:

$$\left(\frac{AF_i^{n+1}}{\Delta t_{n+1}}\right) p_i^{n+1} + \sum_{j=1}^{NN} p_i^{n+1}\ BF_{ij}^{n+1} + \nu_{p_i}\ p_i^{n+1}$$
$$= Q_i^{n+1} + \nu_{p_i}\ p_{BC_i}^{n+1} + q_{IN_i}^{n+1} + DF_i^{(n+1)*} + \left(\frac{AF_i^{n+1}}{\Delta t_{n+1}}\right) p_i^n + \left(CF_i^{n+1}\right)\left(\frac{dU_i}{dt}\right)^n \quad i = \overline{1, NN} \tag{4.110}$$

where the superscript involving (n) or (n+l) indicates level of time evaluation. The term with level $(n+l)^*$ indicates that the $\langle\langle \rho \underline{g} \rangle\rangle^*$ term is evaluated at the (n) time level on the first iteration, and at the most recent level on subsequent iterations. The other coefficients are evaluated at the (n+l) time level by projection on the first iteration, and at the most recent level on subsequent iterations.

Boundary conditions, fluid sources and sinks

Specified pressures are obtained through the cellwise addition of a fluid flux (see Figure 3.5), Q_{BC_i} [M/s] with reference to (4.93):

$$Q_{BC_i}^{n+1} = \nu_{p_i}\left(p_{BC_i}^{n+1} - p_i^{n+1}\right) \tag{4.111}$$

For a cell in which ν_{p_i} is specified as a large number, this flux term dominates the fluid mass balance and $p_{BC_i}^{n+1} \cong p_i^{n+1}$, achieving a specified pressure at the node representing cell i. Note that specified pressure may change each time step. For cells in which pressure is not specified, ν_{p_i} is set to zero, and no fluid is added to the cell by (4.111).

Both fluid sources, Q_i^{n+1}, and fluid inflows across region boundaries, $q_{IN_i}^{n+1}$, are specified cellwise. They directly add fluid mass to the node in cell i. Thus, fluid sources and boundary inflows are indistinguishable in the model. Fluid sources and flows across boundaries are both accounted for by the vector Q_i^{n+1} in SUTRA, and are referred to as fluid sources. Thus the term $q_{IN_i}^{n+1}$ in (4.110) may be dropped, and the definition of Q_i^{n+1} may be generalized to include the boundary flows.

The form of the discretized fluid mass balance implemented in SUTRA is as follows:

$$\sum_{j=1}^{NN}\left[\left(\frac{AF_i\delta_{ij}}{\Delta t_{n+1}}\right)+BF_{ij}^{n+1}+\nu_{p_i}\delta_{ij}\right]p_j^{n+1}$$
$$=Q_i^{n+1}+\nu_{p_i}p_{BC_i}^{n+1}+DF_i^{(n+1)*}+\left(\frac{AF_i^{n+1}}{\Delta t_{n+1}}\right)p_i^n+\left(CF_i^{n+1}\right)\left(\frac{dU_i}{dt}\right)^n \qquad i=\overline{1,NN} \tag{4.112}$$

wherein δ_{ij} is the Kronecker delta:

$$\delta_{ij}=\begin{cases}0 \text{ if } i\neq j\\ 1 \text{ if } i=j\end{cases} \tag{4.112a}$$

4.5 Numerical Approximation of SUTRA Unified Solute Mass and Energy Balance

The governing equation representing the SUTRA unified energy and solute mass balance (2.52) is modified by the addition of a point source term that arises due to fluid inflows and outflows at points of specified pressure:

$$\begin{aligned}O_u(U)=&\left[\varepsilon S_w\rho c_w+(1-\varepsilon)\rho_s c_s\right]\frac{\partial U}{\partial t}+\varepsilon S_w\rho c_w\,\underline{v}\cdot\underline{\nabla}U\\&-\underline{\nabla}\cdot\left\{\rho c_w\left[\varepsilon S_w\left(\sigma_w\underline{\underline{I}}+\underline{\underline{D}}\right)+(1-\varepsilon)\sigma_s\underline{\underline{I}}\right]\cdot\underline{\nabla}U\right\}\\&-Q_p c_w\left(U^*-U\right)-\varepsilon S_w\rho\gamma_l^w\,U-(1-\varepsilon)\rho_s\gamma_l^s\,U_s-\varepsilon S_w\rho\gamma_o^w-(1-\varepsilon)\rho_s\gamma_o^s\\&-Q_{PBC}c_w\left(U_{BC}-U\right)\;=\;0\end{aligned} \tag{4.113}$$

The last term is the solute mass or energy source due to fluid inflow at a point of specified pressure, Q_{PBC} [$M/L^3\cdot s$] is the fluid mass source rate given by (4.95), and U_{BC} is the concentration or temperature of the flow. For outflow, $U_{BC} = U$, and the term goes to zero. Relation (4.113) is numerically approximated in the following sections.

Spatial integration

When the expression for $O_u(U)$ in (4.113) is approximated through nodewise, elementwise and cellwise discretizations, it no longer exactly equals zero. The approximate expression , $\langle\langle O_u(U)\rangle\rangle$, equals a spatially varying residual, $R_u(x,y[,z],t)$, as shown for a 2D example in (3.8). A weighted residual formulation may be written as:

$$\int_V\left\langle\left\langle O_u(U)\right\rangle\right\rangle W_i(x,y)\,dV=0 \qquad i=\overline{1,NN} \tag{4.114}$$

where $W_i(x,y[,z])$ is the weighting function, chosen to be either the basis function, $\phi_i(x,y[,z])$ or the asymmetric weighting function, $\omega_i(x,y[,z])$, depending on the term of the equation. Relation

(4.113) is discretized and the approximation is substituted for $\langle\langle O_u(U)\rangle\rangle$ in (4.114). The resulting set of integral terms is evaluated, one term at a time, in the following paragraphs.

The first term is an integral of the temperature or concentration time derivative:

$$\int_V \left[\left\langle\left\langle \varepsilon S_w \rho c_w + (1-\varepsilon)\rho_s c_s \right\rangle\right\rangle \frac{\partial U}{\partial t} \right] \phi_i(x, y[,z])\, dV \tag{4.115}$$

where the term in square brackets is discretized cellwise, and the weighting function is the basis function (written in global coordinates). As the term in double angle brackets has constant value over a cell, i, the integral contains only the basis function and equals the cell volume, V_i, which reduces to (3.15) for 2D simulations. Thus the term is

$$\left[\varepsilon S_w \rho c_w + (1-\varepsilon)\ \rho_s c_s\right]_i \frac{\partial U_i}{\partial t} V_i \tag{4.116}$$

The second integral is

$$\int_V \left\langle\left\langle \varepsilon S_w \rho c_w \underline{v} \cdot \underline{\nabla} U \right\rangle\right\rangle \omega_i(x, y[,z])\, dV \tag{4.117}$$

where the asymmetric weighting function is chosen to allow the use of "upstream weighting" for this term representing advective transport. "Upstream weighting" is intended for use only when the finite-element mesh has been designed too coarse for a particular level of dispersive and advective transport. The asymmetric function adds dispersion in an amount dependent on element length in the flow direction. As a result, it changes the effective dispersion and thus changes the physics of the problem being solved. This term is written in general to allow upstream weighting, but simplifies to weighting with a basis function when the upstream weight (UP in (4.23) and (4.24) for 2D; (4.66) through (4.68) for 3D) is set to zero. Thus, in order not to alter the physics for most simulations, this term will have symmetric weighting.

The coefficients in this term (except velocity) are evaluated at each Gauss point and depend on nodewise discretization of p and U, as indicated by the notation $\langle\{\ \}\rangle$. Porosity is discretized nodewise. Nodewise discretizations of ε and U are written:

$$\varepsilon(x, y[,z]) \cong \left\langle\left\langle \varepsilon \right\rangle\right\rangle = \sum_{i=1}^{NN} \varepsilon_i\ \phi_i(x, y[,z]) \tag{4.118}$$

$$U(x, y[,z], t) \cong \sum_{i=1}^{NN} U_i(t)\, \phi_i(x, y[,z]) \tag{4.119}$$

The velocity is evaluated at each Gauss point during numerical integration in a particular way that depends on consistent discretization of $\underline{\nabla}p$ and $\rho\underline{g}$ terms in Darcy's law. This consistent approximated velocity is denoted $\left\langle\left\langle \underline{v} \right\rangle\right\rangle^*$. Thus, the term (4.117) is evaluated as

$$\sum_{j=1}^{NN} U_j(t) \int_V \left[\left\langle \left\langle \varepsilon \right\rangle \right\rangle \left\langle \left\{ S_w \rho \right\} \right\rangle c_w \left\langle \left\langle \underline{v} \right\rangle \right\rangle^* \cdot \underline{\nabla} \phi_j \right] \omega_i(x, y[,z])\, dV \tag{4.120}$$

Specific heat, c_w, is a constant.

The third term of expanded relation (4.114) is

$$-\int_V \left\langle \left\langle \underline{\nabla} \cdot \left\{ \rho c_w \left[\varepsilon S_w \left(\sigma_w \underline{\underline{I}} + \underline{\underline{D}} \right) + (1-\varepsilon) \sigma_s \underline{\underline{I}} \right] \cdot \underline{\nabla} U \right\} \right\rangle \right\rangle \phi_i(x, y[,z])\, dV \tag{4.121}$$

where the basis function weights the integral. Green's Theorem (3.17) is applied to (4.121) resulting in

$$\begin{aligned} &-\int_\Gamma \left\langle \left\langle \rho c_w \left[\varepsilon S_w \left(\sigma_w \underline{\underline{I}} + \underline{\underline{D}} \right) + (1-\varepsilon) \sigma_s \underline{\underline{I}} \right] \cdot \underline{\nabla} U \right\rangle \right\rangle \cdot \underline{n}\, \phi_i(x, y[,z])\, d\Gamma \\ &\qquad + \int_V \left\langle \left\langle \rho c_w \left[\varepsilon S_w \left(\sigma_w \underline{\underline{I}} + \underline{\underline{D}} \right) + (1-\varepsilon) \sigma_s \underline{\underline{I}} \right] \cdot \underline{\nabla} U \right\rangle \right\rangle \cdot \underline{\nabla} \phi_i\, dV \end{aligned} \tag{4.122}$$

The first term represents the diffusive/dispersive flux of solute mass or energy out across a system boundary in the region of node i. This term is denoted, ψ_{OUT_i}. An influx would be $-\psi_{OUT_i}$ or ψ_{IN_i}. The second term is based on nodewise discretization of U. The coefficients ρ and S_w are evaluated at Gauss points based on nodewise discretization of U and p, as indicated by the notation $\langle\langle \{ \} \rangle\rangle$. Porosity, ε, is discretized nodewise as in (4.118), and c_w, σ_w and σ_s are constants. The dispersion tensor, $\underline{\underline{D}}$, is evaluated at each Gauss point according to equations (2.41) and (2.42). Velocities used in this evaluation are the consistent values, $\left\langle \left\langle \underline{v} \right\rangle \right\rangle^*$, and dispersivities, α_L and α_{T1} [and α_{T2}], are discretized elementwise except that α_L and α_{T1} [and α_{T2}] are evaluated depending on the velocity direction at each Gauss point for the anisotropic media model. The approximated $\underline{\underline{D}}$ is denoted $\left\langle \left\langle \underline{\underline{D}} \right\rangle \right\rangle$. Thus, the term (4.121) is evaluated as:

$$-\psi_{IN_i} + \sum_{j=1}^{NN} U_j(t) \int_V \left\{ \left\langle \left\langle \{ \rho \} \right\rangle \right\rangle c_w \left[\left\langle \left\langle \varepsilon \right\rangle \right\rangle \left\langle \left\langle \{ S_w \} \right\rangle \right\rangle \left(\sigma_w \underline{\underline{I}} + \left\langle \left\langle \underline{\underline{D}} \right\rangle \right\rangle \right) + \left(1 - \left\langle \left\langle \varepsilon \right\rangle \right\rangle \right) \sigma_s \underline{\underline{I}} \right] \cdot \underline{\nabla} \phi_j \right\} \cdot \underline{\nabla} \phi_i\, dV \tag{4.123}$$

The remaining terms in (4.114) are discretized cellwise with the basis function as the weighting function and with coefficients U_s, S_w and ρ depending on the nodal values of p and U:

$$-\int_V \left\langle \left\langle Q_p c_w \left(U^* - U \right) \right\rangle \right\rangle \phi_i(x, y[,z])\, dV = -Q_i c_w \left(U_i^* - U_i \right) \tag{4.124}$$

$$-\int_V \left\langle \left\langle \varepsilon S_w \rho \gamma_1^w\, U \right\rangle \right\rangle \phi_i(x, y[,z])\, dV = -\left[\varepsilon S_w \rho \gamma_1^w \right]_i U_i V_i \tag{4.125}$$

$$-\int_V \left\langle\left\langle (1-\varepsilon)\rho_s \gamma_1^s\, U_s \right\rangle\right\rangle \phi_i(x, y[, z])\, dV = -\left[(1-\varepsilon)\rho_s \gamma_1^s\, U_s\right]_i V_i \tag{4.126}$$

$$-\int_V \left\langle\left\langle \varepsilon S_w \rho \gamma_o^w + (1-\varepsilon)\rho_s \gamma_o^s \right\rangle\right\rangle \phi_i(x, y[, z])\, dV = -\left[\varepsilon S_w \rho \gamma_o^w + (1-\varepsilon)\rho_s \gamma_o^s\right]_i V_i \tag{4.127}$$

$$-\int_V \left\langle\left\langle Q_{PBC} c_w (U_{BC} - U) \right\rangle\right\rangle \phi_i(x, y[, z])\, dV = -Q_{BC_i} c_w \left(U_{BC_i} - U_i\right) \tag{4.128}$$

where

$$Q_{BC_i} = \nu_{p_i} \left(p_{BC_i} - p_i\right) \tag{4.129}$$

and

$$\left\langle\left\langle Q_{PBC} \right\rangle\right\rangle = \sum_{i=1}^{NN} \left(\frac{Q_{BC_i}}{V_i} \right) \tag{4.130}$$

The term evaluated in (4.126) is nonzero only for solute transport, and the value of U_S is given for solute transport by the adsorption isotherms in the form:

$$U_s = C_s = s_L C + s_R \tag{4.131}$$

where s_L and s_R are defined in section 4.7, "Temporal Evaluation of Adsorbate Mass Balance." In the above cellwise relations, c_w, ρ_s, γ_1^w, and γ_1^s, are constant, and γ_o^w, γ_o^s, s_L, and s_R may vary cellwise and with time.

By combining and rearranging the evaluations of integrals in expanded relation (4.114) and the definition (4.131), the following NN spatially discretized weighted residual relations are obtained in 3D:

$$\begin{aligned} &AT_i \frac{dU_i}{dt} + \sum_{j=1}^{NN} U_j(t)\, DT_{ij} + \sum_{j=1}^{NN} U_j(t)\, BT_{ij} - \left(GT_i + G_s TL_i\right) U_i(t) \\ &\quad + Q_i c_w U_i(t) + Q_{BC_i} c_w U_i(t) \\ &= Q_i c_w U_i^* + Q_{BC_i} c_w U_{BC_i} + \psi_{IN_i} + ET_i + G_s TR_i \qquad i = \overline{1, NN} \end{aligned} \tag{4.132}$$

where

$$AT_i = \left[\varepsilon S_w \rho c_w + (1-\varepsilon)\rho_s c_s\right]_i V_i \tag{4.133}$$

$$DT_{ij} = \int_x \int_y \int_z \left[\langle\langle \varepsilon \rangle\rangle \langle\langle \{S_w \rho\} \rangle\rangle c_w \langle\langle \underline{v} \rangle\rangle^* \cdot \underline{\nabla} \phi_j \right] \omega_i \, dz \, dy \, dx \tag{4.134}$$

$$BT_{ij} = \int_x \int_y \int_z \left\{ \langle\langle \{\rho\} \rangle\rangle c_w \left[\langle\langle \varepsilon \rangle\rangle \langle\langle \{S_w\} \rangle\rangle \left(\sigma_w \underline{\underline{I}} + \langle\langle \underline{\underline{D}} \rangle\rangle \right) + \left(1 - \langle\langle \varepsilon \rangle\rangle\right) \sigma_s \underline{\underline{I}} \right] \cdot \underline{\nabla} \phi_j \right\} \cdot \underline{\nabla} \phi_i \, dz \, dy \, dx \tag{4.135}$$

$$GT_i = \left(\varepsilon S_w \, \rho \, \gamma_1^w \right)_i \, V_i \tag{4.136a}$$

$$G_s TL_i = \left[(1 - \varepsilon) \rho_s \gamma_1^s s_L \right]_i \, V_i \tag{4.136b}$$

$$G_s TR_i = \left[(1 - \varepsilon) \rho_s \gamma_1^s s_R \right]_i \, V_i \tag{4.136c}$$

$$ET_i = \left[\varepsilon S_w \rho \gamma_o^w + (1 - \varepsilon) \rho_s \gamma_o^s \right]_i \, V_i \tag{4.137}$$

For 2D simulations, (4.134) and (4.135) reduce to

$$DT_{ij} = \int_x \int_y \left[\langle\langle \varepsilon \rangle\rangle \langle\langle \{S_w \rho\} \rangle\rangle c_w \langle\langle \underline{v} \rangle\rangle^* \cdot \underline{\nabla} \phi_j \right] \omega_i \, B \, dy \, dx \tag{4.138}$$

$$BT_{ij} = \int_x \int_y \left\{ \langle\langle \{\rho\} \rangle\rangle c_w \left[\langle\langle \varepsilon \rangle\rangle \langle\langle \{S_w\} \rangle\rangle \left(\sigma_w \underline{\underline{I}} + \langle\langle \underline{\underline{D}} \rangle\rangle \right) + \left(1 - \langle\langle \varepsilon \rangle\rangle\right) \sigma_s \underline{\underline{I}} \right] \cdot \underline{\nabla} \phi_j \right\} \cdot \underline{\nabla} \phi_i \, B \, dy \, dx \tag{4.139}$$

The thickness of the mesh, B(x,y), is evaluated at each Gauss point according to (4.103).

The only integrals requiring Gaussian integration are DT_{ij} and BT_{ij}. Note that these are evaluated in SUTRA subroutine ELEMN2 (for 2D) or ELEMN3 (for 3D) in an element-by-element manner. The remaining terms that do not involve Q_{BC} are evaluated cellwise by SUTRA subroutine NODAL. The flux terms arising from specified pressure (those with Q_{BC}) are evaluated by subroutine BC.

Temporal discretization and iteration

The time derivative in the spatially discretized and integrated equation is approximated by finite differences:

$$\frac{dU_i}{dt} \cong \frac{U_i^{n+1} - U_i^n}{\Delta t_{n+1}} \tag{4.140}$$

where

$$U_t^n = U_i\left(t^n\right) \tag{4.141a}$$

$$U_t^{n+1} = U_i\left(t^n + \Delta t_n\right) = U_i\left(t^{n+1}\right) \tag{4.141b}$$

All terms in (4.132) are evaluated at the new time level, t^{n+1}, except the velocity in (4.134) or (4.138) and the dispersion tensor (which involves velocity) in (4.135) or (4.139), which are lagged (based on values from previous time steps, as described just below) on the first iteration. Because coefficients depend on the yet unknown values of p and U at the end of the time step, one or more iterations may be used to solve this nonlinear problem. On the first iteration, and when only one iteration per time step is used, coefficients are based on a projected value of p and U as given by (4.108) and (4.109). On subsequent iterations, coefficients are based on the most recent value of p and U. Iterations end when the convergence criteria are satisfied.

On the first iteration, and when only one iteration per time step is used, the velocities are evaluated based on p_i^n, U_i^{n-1} and ρ_i^{n-1}. This is because the pressure gradient in the velocity calculation, $\underline{\nabla} p^n$, is based on pressures calculated when the fluid density was ρ^{n-1}. On subsequent iterations, velocities are calculated using the pressure solution for the most recent iteration together with the densities resulting from the previous iteration upon which the most recent pressure solution was based. No spurious velocities, which arise from mismatched p and ρ, are generated this way. The flux term, Q_{BC}, arising from the specified pressures is evaluated on the first iteration at the beginning of the time step in terms of p_i^n and $p_{BC_i}^n$. On subsequent iterations, it is based on the most recent pressure solution and $p_{BC_i}^{n+1}$.

The relations (4.132) may thus be written in a form which allows for solution of concentration or temperature at nodes, U_i^{n+1}, at the end of the present time step:

$$\left(\frac{AT_i^{n+1}}{\Delta t_{n+1}}\right)U_i^{n+1} + \sum_{j=1}^{NN} U_j^{n+1} DT_{ij}^{(n+1)*} + \sum_{j=1}^{NN} U_j^{n+1} BT_{ij}^{n+1}$$
$$+\left(GT_i^{n+1} + G_s TL_i^{n+1}\right)U_i^{n+1} + Q_i^{n+1} c_w\, U_i^{n+1} + Q_{BC_i}^{(n+1)*} c_w\, U_i^{n+1} \tag{4.142}$$
$$= Q_i^{n+1} c_w\, U_i^{*n+1} + Q_{BC_i}^{(n+1)*} c_w\, U_{BC_i}^{n+1} + \psi_{IN_i}^{n+1} + ET_i^{n+1} + G_s TR_i^{n+1} + \left(\frac{AT_i^{n+1}}{\Delta t_{n+1}}\right)U_i^n \qquad i = \overline{1, NN}$$

The $(n+1)^*$ level indicates that velocity and Q_{BC} are evaluated on the first iteration at the previous time step level (n) and on subsequent iterations, at the most recent iteration level of the present time step. Other coefficients are evaluated at the (n+1) time level by projection on the first iteration, and then at the most recent iteration level on subsequent iterations.

Boundary conditions, energy or solute mass sources and sinks

Specified temperatures or concentrations at nodes are obtained numerically by adding the following source term to the right side of (4.142) for all nodes:

$$\psi_{BC_i}^{n+1} = \nu_{U_i}\left(U_{UBC_i}^{n+1} - U_i^{n+1}\right) \tag{4.143}$$

where $\psi_{BC_i}^{n+1}$ is a source of energy [E/s] or solute mass [M_S/M·s], ν_{U_i} is the user-specified conductance value (for energy or solute flux through a hypothetical core - see Figure 3.5) which is nonzero only for nodes, k, at which temperature or concentration is specified, and $U_{UBC_k}^{n+1}$ is the user-specified value of temperature or concentration at time, t^{n+1} and at node k.

Source boundary conditions for U arise whenever a fluid source Q_i is specified. These may be either point sources of fluid or fluid flows across the boundaries. These fluid inflows must be assigned concentration or temperature values, U_i^{*n+1} which may change with each time step. Note that these sources are evaluated in SUTRA subroutine NODAL. Outflows of fluid result in the disappearance of the source term from the transport equation because the sink and aquifer have the same U-value ($U_i^{*n+1} = U_i^{n+1}$).

Source boundary conditions for U may arise at points of specified pressure when an inflow Q_{BC_i} occurs at such a point. A value of U must be specified for such fluid inflows as $U_{UBC_i}^{n+1}$. These values may change with each time step. This source term for U disappears for outflow at a point of specified pressure. Note that specified pressure sources are evaluated in SUTRA subroutine BC.

A source or sink at a boundary due to diffusion or dispersion appears in (4.122):

$$\psi_{IN_i}^{n+1} = \int_\Gamma \left\langle \left\langle \rho c_w \left[\varepsilon S_w \left(\sigma_w \underline{\underline{I}} + \underline{\underline{D}} \right) + (1-\varepsilon)\sigma_s \underline{\underline{I}} \right] \cdot \underline{\nabla} U \right\rangle \right\rangle^{n+1} \cdot \underline{n}\, \phi_i \, d\Gamma \qquad (4.144)$$

For solute transport, this term may represent molecular diffusion and dispersion of solute mass across a boundary. For energy transport, this term represents heat conduction and thermal dispersion across a boundary. This heat or solute flux is a user-specified value, which may change each time step. If the term is set to zero, it implies no diffusion and no dispersion across a boundary for solute transport, or for energy transport, it implies perfect thermal insulation and no dispersion across a boundary. For an open boundary across which fluid flows, this term is not automatically evaluated by SUTRA. If no user-specified value exists at an open boundary, then this term is set to zero. This implicitly assumes that the largest part of solute or energy flux across an open boundary is advectively transported rather than diffusively or dispersively transported. In cases where this assumption is inappropriate, the code may be modified to evaluate this term at the new time level depending on the value of U^{n+1}.

The form of the discretized unified energy and solute mass balance equation, which is implemented in SUTRA, is as follows:

$$\begin{aligned} &\sum_{j=1}^{NN} \left\{ \left(\frac{AT_i^{n+1}\delta_{ij}}{\Delta t_{n+1}} \right) + DT_{ij}^{(n+1)*} + BT_{ij}^{n+1} + \left[\nu_{U_i} + GT_i^{n+1} + G_s TL_i^{n+1} + \left(Q_i^{n+1} + Q_{BC_i}^{n+1} \right) c_w \right] \delta_{ij} \right\} U_j^{n+1} \\ &= c_w \left(Q_i^{n+1}\, U_i^{(n+1)*} + Q_{BC_i}^{n+1}\, U_{BC_i}^{n+1} \right) + \nu_{U_i} U_{UBC_i}^{n+1} + \psi_{IN_i}^{n+1} + ET_i^{n+1} \qquad (4.145) \\ &\quad + G_s TR_i^{n+1} + \left(\frac{AT_i^{n+1}}{\Delta t_{n+1}} \right) U_i^n \qquad\qquad i = \overline{1, NN} \end{aligned}$$

wherein δ_{ij} is the Kronecker delta.

4.6 Consistent Evaluation of Fluid Velocity

Fluid velocity is defined by equation (2.19) as

$$\underline{v} = -\left(\frac{\underline{\underline{k}} k_r}{\varepsilon S_w \mu} \right) \cdot \left(\underline{\nabla} p - \rho \underline{g} \right) \tag{4.146}$$

This relation strictly holds true at a point in space. In order for the relation to hold true when discretized, the terms $\underline{\nabla} p$ and $\rho \underline{g}$ must be given the same type of spatial variability. This avoids generation of spurious velocities, which would be caused by local mismatching of the discretized pressure gradient term and density-gravity term. For example, in a hydrostatic system where densities vary spatially, $\underline{\nabla} p$ must equal $\rho \underline{g}$ to yield a zero vertical velocity. However, if $\underline{\nabla} p$ and $\rho \underline{g}$ do not locally cancel because of the discretizations chosen, then erroneous vertical velocities are generated.

Such an error would occur over an element where $\underline{\nabla} p$ is allowed only a single constant value in a vertical section of the element and ρ is allowed to vary linearly in the vertical direction. This is the case in a standard finite-element approximation wherein both p and U vary linearly in the vertical direction across an element. Linear change in p implies a constant value $\underline{\nabla} p$, while linear change in U implies a linear change in the value of ρ according to (2.3) or (2.4). Thus, a standard finite-element approximation over a bilinear element results in inconsistent approximation in the vertical direction for $\underline{\nabla} p$ and $\rho \underline{g}$: constant $\underline{\nabla} p$ and linearly varying ρ. This inconsistency generates spurious vertical velocities, especially in regions of sharp vertical changes in U. A consistent approximation of velocity is one in which $\underline{\nabla} p$ and $\rho \underline{g}$ are allowed the same type of spatial variability, and further, are evaluated at the same time level.

A consistent evaluation of velocity is required by the transport solution in (4.134) or (4.138) and in the evaluation of the dispersion tensor in (4.135) or (4.139), where velocity is required in each element, in particular, at the Gauss points for numerical integration. In addition, a consistent evaluation of the $\rho \underline{g}$ term is required at the Gauss points in each element for the fluid mass balance solution in the integral shown in (4.100) or (4.102).

The coefficients for calculation of velocity in (4.146) are discretized as follows: *Permeability*, $\underline{\underline{k}}$, is discretized elementwise; *porosity*, ε, is discretized nodewise. *Unsaturated flow parameters*, k_r and S_w, are given values depending on the nodewise-discretized pressure according to relations (2.8) and (2.21). *Viscosity* is either constant for solute transport or is given values depending on nodewise-discretized temperature according to (2.5).

To complete the discretization of velocity, values in global coordinates at the Gauss points are required for the term $(\underline{\nabla} p - \rho \underline{g})$. Whenever this term is discretized consistently in local element coordinates (ξ, η [,ζ]), a consistent approximation is obtained in global coordinates for any arbitrarily oriented quadrilateral (2D) or hexahedral (3D) element. The remainder of this section presents a consistent approximation for this term.

Consistent discretization in local coordinates is obtained when the spatial dependence of $\partial p/\partial \xi$ and $\partial p/\partial \eta$ [and $\partial p/\partial \zeta$] is of the same type as that of ρg_ξ and ρg_η [and ρg_ζ]. Because the

discretization for p(ξ, η [,ζ]) has already been chosen to be bilinear (2D) or trilinear (3D), it is the discretization of the $\rho \underline{g}$ term, in particular, that must be adjusted. First, in the following, a discretization of the $\rho \underline{g}$ term is presented which is consistent with the discretization of $\underline{\nabla} p$ in local coordinates, and then both $\underline{\nabla} p$ and $\rho \underline{g}$ are transformed to global coordinates while maintaining consistency. The development is presented for the 3D case. In 2D, the summations are performed over four (instead of eight) nodes and only terms not involving z or ζ are relevant.

The pressure gradient within a 3D element in local coordinates is defined in terms of the derivatives with respect to the local coordinates:

$$\frac{\partial p}{\partial \xi}(\xi, \eta, \zeta) = \sum_{i=1}^{8} p_i \frac{\partial \Omega_i}{\partial \xi} \tag{4.147a}$$

$$\frac{\partial p}{\partial \eta}(\xi, \eta, \zeta) = \sum_{i=1}^{8} p_i \frac{\partial \Omega_i}{\partial \eta} \tag{4.147b}$$

$$\frac{\partial p}{\partial \zeta}(\xi, \eta, \zeta) = \sum_{i=1}^{8} p_i \frac{\partial \Omega_i}{\partial \zeta} \tag{4.147c}$$

The summations may be expanded and written in detail by reference to relations (4.39) - (4.46) and (4.25) - (4.30) (in 2D, (4.9) - (4.12) and (4.1) - (4.4)).

A local discretization of $\rho \underline{g}$ with a spatial functionality that is consistent with the local pressure derivatives (4.147a-c) is

$$(\rho g)_\xi(\xi, \eta, \zeta) = \sum_{i=1}^{8} \rho_i g_{\xi_i} \left| \frac{\partial \Omega_i}{\partial \xi} \right| \tag{4.148}$$

$$(\rho g)_\eta(\xi, \eta, \zeta) = \sum_{i=1}^{8} \rho_i g_{\eta_i} \left| \frac{\partial \Omega_i}{\partial \eta} \right| \tag{4.149}$$

$$(\rho g)_\zeta(\xi, \eta, \zeta) = \sum_{i=1}^{8} \rho_i g_{\zeta_i} \left| \frac{\partial \Omega_i}{\partial \zeta} \right| \tag{4.150}$$

where the vertical bars indicate absolute value, ρ_i is the value of ρ at node i in the element based on the value of U at the node through relation (2.3) or (2.4), and g_{ξ_i}, g_{η_i}, and g_{ζ_i} are the ξ-, η-, and ζ-components of $\underline{g}$ at node i, respectively. The 24 (in 2D, eight) gravity vector components at the nodes in each element need be calculated only once for a given mesh and may be saved. This discretization is robust in that it allows both the density and the direction and magnitude of the gravity vector components to vary over an element. No particular significance should be attached to the absolute values of basis function derivatives, except that these happen to give the desired consistent approximations, as is shown shortly.

The gravity vector components in local coordinates at a point in the 3D element are obtained from the global gravity components as:

$$\begin{Bmatrix} g_\xi \\ g_\eta \\ g_\zeta \end{Bmatrix} = [J] \begin{Bmatrix} g_x \\ g_y \\ g_z \end{Bmatrix} \tag{4.151}$$

where [J] is the Jacobian matrix defined by (4.69).

The derivatives of pressure in local coordinates, (4.147a-c), and the consistent density-gravity term components in local coordinates, (4.148) - (4.150), are transformed to global coordinates for use in the evaluation of the integrals in which they appear by

$$\begin{Bmatrix} \frac{\partial p}{\partial x} \\ \frac{\partial p}{\partial y} \\ \frac{\partial p}{\partial z} \end{Bmatrix} = \left[J^{-1}\right] \begin{Bmatrix} \frac{\partial p}{\partial \xi} \\ \frac{\partial p}{\partial \eta} \\ \frac{\partial p}{\partial \zeta} \end{Bmatrix} \tag{4.152}$$

$$\begin{Bmatrix} \left(\left\langle\left\langle \rho \underline{g} \right\rangle\right\rangle^*\right)_x \\ \left(\left\langle\left\langle \rho \underline{g} \right\rangle\right\rangle^*\right)_y \\ \left(\left\langle\left\langle \rho \underline{g} \right\rangle\right\rangle^*\right)_z \end{Bmatrix} = \left[J^{-1}\right] \begin{Bmatrix} (\rho g)_\xi \\ (\rho g)_\eta \\ (\rho g)_\zeta \end{Bmatrix} \tag{4.153}$$

where $(\langle\langle \rho \underline{g} \rangle\rangle^*)_x$, $(\langle\langle \rho \underline{g} \rangle\rangle^*)_y$, and $(\langle\langle \rho \underline{g} \rangle\rangle^*)_z$ are the consistently discretized density-gravity term components in global coordinates, and $[J]^{-1}$ is the inverse Jacobian matrix defined by (4.73a). The 2D forms of relations (4.147) to (4.153) are obtained by leaving out all terms involving z and ζ.

The spatial consistency of these approximations may be seen by inspecting their expansions in local coordinates. For example, in 2D, the ξ-components are

$$\frac{\partial p}{\partial \xi} = \frac{1}{4}\left[(p_2 - p_1)(1-\eta) + (p_3 - p_4)(1+\eta)\right] \tag{4.154}$$

$$(\rho g)_\xi = \frac{1}{4}\left[\left(\rho_1 g_{\xi_1} + \rho_2 g_{\xi_2}\right)(1-\eta) + \left(\rho_3 g_{\xi_3} + \rho_4 g_{\xi_4}\right)(1+\eta)\right] \tag{4.155}$$

The terms in parentheses preceding the terms containing η all have a constant value for the element, and thus the approximations have consistent spatial dependences.

4.7 Temporal Evaluation of Adsorbate Mass Balance

The terms in the unified energy and solute mass balance equation that stem from the adsorbate mass balance require particular temporal evaluation because some are nonlinear. The following terms of relation (4.142) are evaluated here: AT_i^{n+1}, GT_i^{n+1}, and ET_i^{n+1}. For solute transport, the coefficient c_{s_i} in AT_i^{n+1} (4.133) becomes $\kappa_{l_i}^{n+1}$ according to (2.52b). The relation that defines κ_l is given by either (2.34c), (2.35c), or (2.36c) depending on the sorption isotherm. The variable $U_{s_i}^{n+1}$ is expressed in terms of the concentration of adsorbate, $C_{s_i}^{n+1}$, in a form given by (4.131). The parameters in (4.131), s_L and s_R, are defined in this section and are based on either (2.34a), (2.35a) and (2.36a) depending again on the sorption isotherm. The temporal approximations of these parameters are described below for each isotherm.

For linear sorption, all terms and coefficients related to the adsorbate mass are linear and are evaluated at the new time level and strictly solved for at this level:

$$U_{s_i}^{n+1} = C_{s_i}^{n+1} = \chi_1 \, \rho_o \, C_i^{n+1} \tag{4.156a}$$

$$C_{s_i}^{n+1} = \kappa_{l_i}^{n+1} = \chi_1 \rho_o \tag{4.156b}$$

$$s_L = \chi_1 \rho_o \tag{4.156c}$$

$$s_R = 0 \tag{4.156d}$$

For Freundlich sorption, the adsorbate concentration is split into a product of two parts for temporal evaluation. One part is treated as a first-order term as is linear sorption. This part is evaluated strictly at the new time level and solved for on each iteration or time step. The remaining part is evaluated as a known quantity, either based on the projected value of C_i at the end of the time step on the first iteration, or based on the most recent C_i solution on any subsequent iteration.

$$U_{S_i}^{n+1} = C_{S_i}^{n+1} = \left[\chi_1 (\rho_o)^{\left(\frac{1}{\chi_2}\right)} \left(C_i^{proj}\right)^{\left(\frac{1-\chi_2}{\chi_2}\right)} \right] C_i^{n+1} \tag{4.157a}$$

Also:

$$C_{S_i}^{n+1} = \kappa_{l_i}^{n+1} = \left(\frac{\chi_1}{\chi_2}\right) \rho_o^{\left(\frac{1}{\chi_2}\right)} \left(C_i^{proj}\right)^{\left(\frac{1-\chi_2}{\chi_2}\right)} \tag{4.157b}$$

$$s_L = \chi_1 (\rho_o)^{\left(\frac{1}{\chi_2}\right)} \left(C_i^{proj}\right)^{\left(\frac{1-\chi_2}{\chi_2}\right)} \qquad s_R = 0 \tag{4.157c}$$

where the coefficient $\kappa_{l_i}^{n+1}$ is evaluated from the projected or most recent value of C_i, depending on the iteration.

Finally, for Langmuir sorption the form used for the temporal evaluation preserves dependence on a linear relation to C_i. However, the linear relation is appropriate only at low solute concentrations. At high concentrations, the adsorbate concentration approaches (χ_1/χ_2). Therefore, two temporal approximations are combined, (one for low C, and one for high C) in a manner depending on the magnitude of concentration. When $(\chi_2 \rho_o c) << 1$, the following temporal approximation for low values of C, referred to as C_s^o, is employed:

$$C_s^o = \left(\chi_1 \rho_o C^{n+1}\right)\left[1 - \frac{\chi_2 \rho_o C^{proj}}{\left(1 + \chi_2 \rho_o C^{proj}\right)}\right] \tag{4.158}$$

When $(\chi_2 \rho_o c) >> 1$, the following temporal approximation for high C, C_s^∞ is employed:

$$C_s^\infty = \left(\frac{\chi_1}{\chi_2}\right)\left[1 - \frac{1}{\left(1 + \chi_2 \rho_o C^{proj}\right)}\right] \tag{4.159}$$

Thus $C_{s_i}^{n+1}$ may be defined

$$U_{s_i}^{n+1} = C_{s_i}^{n+1} = W_o C_{s_i}^o + W_\infty C_{s_i}^\infty \tag{4.160}$$

where the weights W_o and W_∞, are

$$W_\infty = \frac{\chi_2 \rho_o C^{proj}}{\left(1 + \chi_2 \rho_o C^{proj}\right)} \tag{4.161a}$$

$$W_o = 1 - W_\infty \tag{4.161b}$$

By substituting (4.158), (4.159), (4.161a), and (4.161b) into (4.160), the following temporal evaluation of $C_{s_i}^{n+1}$ is obtained after algebraic manipulation:

$$C_{s_i}^{n+1} = \frac{\chi_1 \rho_o C_i^{n+1}}{\left(1 + \chi_2 \rho_o C_i^{proj}\right)^2} + \frac{\left(\chi_1 \rho_o C_i^{proj}\right)\left(\chi_2 \rho_o C_i^{proj}\right)}{\left(1 + \chi_2 \rho_o C_i^{proj}\right)^2} \tag{4.162a}$$

The coefficient $\kappa_{l_i}^{n+1}$ is defined as

$$C_{s_i}^{n+1} = \kappa_{l_i}^{n+1} = \frac{\chi_1 \rho_o}{\left(1 + \chi_2 \rho_o C_i^{proj}\right)^2} \tag{4.162b}$$

$$s_L = \frac{\chi_1 \rho_o}{\left(1 + \chi_2 \rho_o C_i^{proj}\right)^2} \tag{4.162c}$$

$$s_R = \frac{(\chi_1 \chi_2)\left(\rho_o C_i^{proj}\right)^2}{\left(1 + \chi_2 \rho_o C_i^{proj}\right)^2} \tag{4.162d}$$

The first term in (4.162a) is solved for on each iteration, and the second term is treated as a known. In the above four relations, C_i^{proj} is based on a projection for the first iteration on a time step, and is the most recent value of C_i on subsequent iterations for the time step.

Chapter 5: Other Methods and Algorithms

5.1 Rotation of Permeability Tensor

The aquifer permeability may be anisotropic (as discussed in section 2.2 under the heading "Fluid flow and flow properties") and may vary in magnitude and direction from element to element (as shown in (4.90)). In 2D, the permeability in each element is completely described by input data values for the principal permeability values k_{max} and k_{min}, and for θ, the direction in degrees from the global +x direction to the direction of maximum permeability. In 3D, it is described completely by input data values for k_{max}, k_{mid}, and k_{min}, and for the angles θ_1, θ_2, and θ_3, which describe how the principal permeability directions are related to the (x,y,z)-coordinate directions (see section 2.2). The evaluation of integrals (4.99) and (4.100) (or (4.101) and (4.102)) as well as the velocity evaluation (4.146) require the permeability tensor components in global coordinates as given by (4.90). Thus, a rotation of the tensor is required from the principal permeability directions to the global coordinate directions, as shown for 2D in Figure 2.2.

The rotation is given by

$$\underline{\underline{k}}^{L} = \underline{\underline{H}}\, \underline{\underline{k}}_{p}^{L}\, \underline{\underline{H}}^{T} \tag{5.1}$$

where in 2D,

$$\underline{\underline{k}}_{p}^{L} = \begin{bmatrix} k_{max}^{L} & 0 \\ 0 & k_{min}^{L} \end{bmatrix} \tag{5.2a}$$

$$\underline{\underline{H}} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{5.2b}$$

and in 3D,

$$\underline{\underline{k}}_{p}^{L} = \begin{bmatrix} k_{max}^{L} & 0 & 0 \\ 0 & k_{mid}^{L} & 0 \\ 0 & 0 & k_{min}^{L} \end{bmatrix} \tag{5.3a}$$

$$\underline{\underline{H}} = \begin{bmatrix} \cos\theta_1 \cos\theta_2 & \begin{matrix} -\sin\theta_1 \cos\theta_3 \\ -\cos\theta_1 \sin\theta_2 \sin\theta_3 \end{matrix} & \begin{matrix} +\sin\theta_1 \sin\theta_3 \\ -\cos\theta_1 \sin\theta_2 \cos\theta_3 \end{matrix} \\ \sin\theta_1 \cos\theta_2 & \begin{matrix} +\cos\theta_1 \cos\theta_3 \\ -\sin\theta_1 \sin\theta_2 \sin\theta_3 \end{matrix} & \begin{matrix} -\cos\theta_1 \sin\theta_3 \\ -\sin\theta_1 \sin\theta_2 \cos\theta_3 \end{matrix} \\ \sin\theta_2 & \cos\theta_2 \sin\theta_3 & \cos\theta_2 \cos\theta_3 \end{bmatrix} \tag{5.3b}$$

The matrix $\underline{\underline{k}}^L$ is given by the relevant portion of (4.90), depending on whether it is 2D or 3D.

The matrix $\underline{\underline{H}}^T$ is the transpose of $\underline{\underline{H}}$, defined by

$$H_{ij}^T = H_{ji} \text{ for all i and j} \tag{5.4}$$

For example, the result in 2D is

$$k_{xx}^L = k_{max}^L \cos^2\theta + k_{min}^L \sin^2\theta \tag{5.5a}$$

$$k_{yy}^L = k_{max}^L \sin^2\theta + k_{min}^L \cos^2\theta \tag{5.5b}$$

$$k_{xy}^L = k_{yx}^L = \left(k_{max}^L - k_{min}^L\right)\sin\theta \ \cos\theta \tag{5.5c}$$

5.2 Radial Coordinates

For 2D simulations, SUTRA is written in terms of 2D Cartesian coordinates x and y. In general, the 2D numerical methods are applied to Cartesian forms of the governing equations; however, radial coordinates (cylindrical coordinates) r and z can be exactly represented by allowing the mesh thickness, B_i, to vary from node to node in an appropriate manner.

A function, f(r,z), of radius r, and vertical coordinate z, is integrated over a cylindrical volume as follows:

$$R = \int_z \int_r \int_\theta f(r,z)\, r\, d\theta\, dr\, dz \tag{5.6}$$

Assuming symmetry with respect to angular coordinate θ (f(r,z) does not depend on θ), the integral becomes

$$R_r = \int_z \int_r f(r,z)(2\pi r)\, dr\, dz \tag{5.7}$$

This integration may be compared with a general integration of a function g(x,y) in Cartesian coordinates as it is carried out in SUTRA methodology:

$$R_c = \int_y \int_x g(y,z)\, B(x,y)\, dx\, dy \tag{5.8}$$

Integrals R_r and R_c are exactly analogous if x≡r, y≡z, and

$$B(x,y) = 2\pi r \tag{5.9}$$

Thus, by a simple redefinition of coordinate names and by setting the mesh thickness, B, at each node equal to the circumference of the circle it would sweep out when rotated about the r=0 axis of the cylinder ($B_i = 2\pi r_i$), the SUTRA simulation is converted exactly to radial coordinates. Figure 5.1 shows a mesh and the volume it sweeps out when in radial coordinates. Each element becomes a 3D ring when used in radial coordinates.

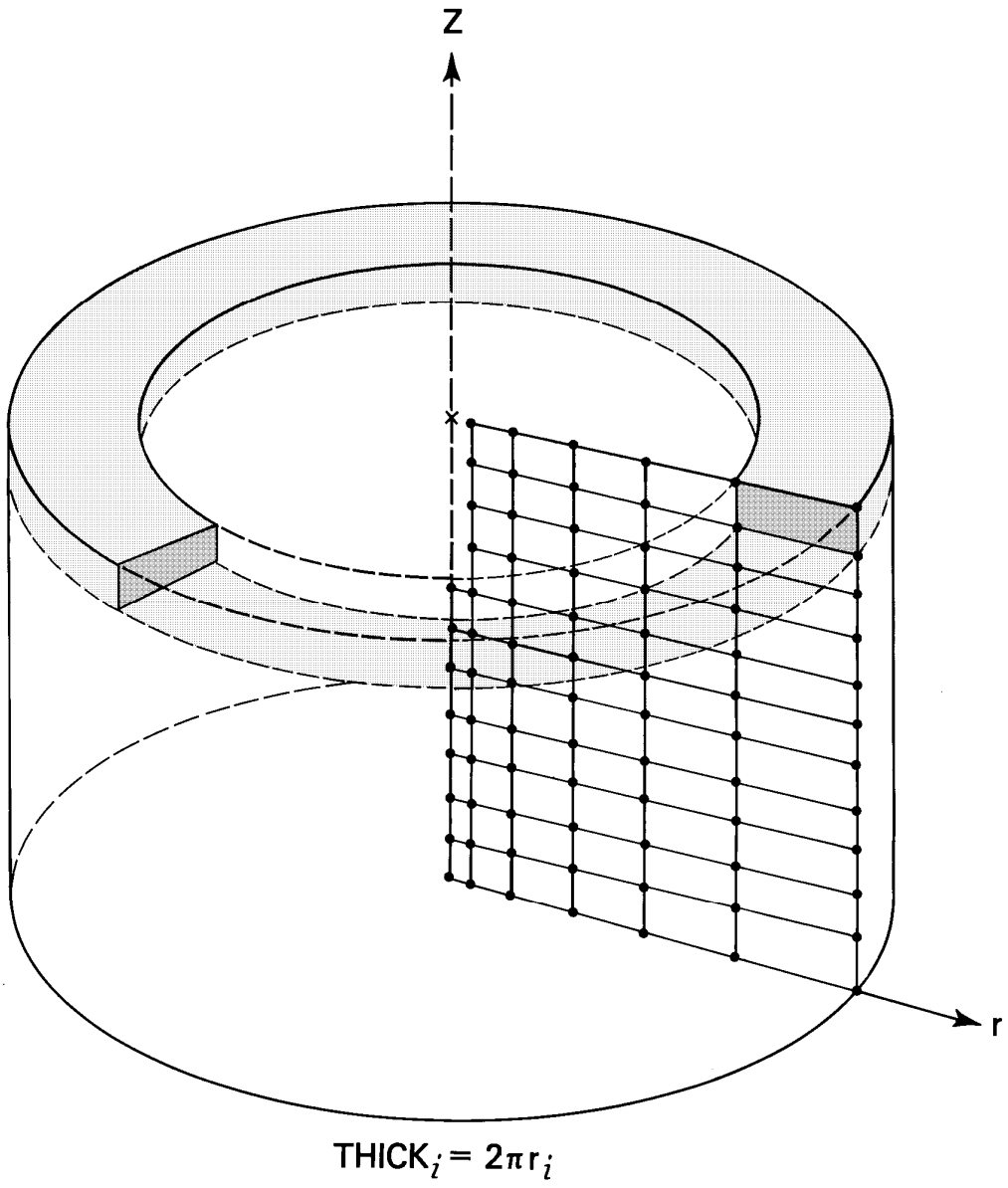


Figure 5.1. Finite-element mesh in radial coordinates. One element is shaded.

5.3 User-defined Schedules

Beginning with this version of SUTRA (2.1), time stepping and the timing of observation output are controlled by user-defined "schedules" – sequences of times or time steps. See sections 5.4 and 5.6 for details on how schedules are used to control time stepping and observation output, respectively.

There are four ways to define a schedule:
1. as a time list – an explicitly listed sequence of times;
2. as a time cycle – a sequence of times defined by an initial time, a limiting time, a time increment, and additional parameters that determine how the time increment changes throughout the sequence;
3. as a step list – an explicitly listed sequence of time steps;
4. as a step cycle – a sequence of time steps defined by an initial time step, a limiting time step, and a time step increment.

Time-based schedules (time lists and time cycles) can include times that fall within time steps (i.e., between the beginning and end of a time step). Step-based schedules (step lists and step cycles) must consist only of integer time step numbers.

Time-based schedules may be defined in terms of either absolute or elapsed times. Absolute times are expressed relative to a time datum of the user's choice. Elapsed times are expressed relative to the starting time of the simulation.

5.4 Control of Time Stepping

In previous versions of SUTRA (2.0 and earlier), user control over time stepping was limited. In this version of SUTRA (2.1), time stepping in a transient simulation is controlled by a user-defined schedule named "TIME_STEPS" that is specified in dataset 6 of the **".inp"** file. The first time value in the "TIME_STEPS" schedule determines the starting time of the simulation. If the schedule is defined in terms of absolute times, then the first time value in the schedule is the (absolute) starting time for the simulation. If the "TIME_STEPS" schedule is defined in terms of elapsed times (times relative to the starting time of the simulation), then first time value in the schedule must be zero, and the (absolute) starting time of the simulation is TICS (from dataset 1 of the **".ics"** file). Subsequent time values define the ends of time steps in terms of either absolute or elapsed times. For a purely steady-state simulation, user-defined schedules are ignored by SUTRA and need not be specified.

5.5 Solution Sequencing

On any given time step, the matrix equations are created and solved in the following order: (1) the matrix equation for the fluid mass balance is set up, (2) the transport balance matrix equation is set up, (3) pressure is solved for, and (4) concentration or temperature is solved for. Both balances are set up on each pass such that the elementwise calculations need be done only once per pass. However, SUTRA allows the p or U equation to be set up and solved only every few time steps in a cyclic manner based on parameters NPCYC and NUCYC. These values represent the solution cycle in time steps. For example, for transient flow and transient transport, setting up and solving for both p and U each time step (NPCYC = NUCYC = l) gives the following cycle:

time step:	1	2	3	4	5	6	7	…
solve for:	p	p	p	p	p	p	p	…
	U	U	U	U	U	U	U	…

or solving for p every third time step and for U each time step (NPCYC=3 and NUCYC=1) gives:

time step:	1	2	3	4	5	6	7	8	9	10	11	12	13	…
solve for:	p	•	p	•	•	p	•	•	p	•	•	p	•	…
	U	U	U	U	U	U	U	U	U	U	U	U	U	…

However, either of p or U must be solved for on each time step and therefore either NPCYC or NUCYC must be set to one.

For a simulation with steady-state flow and transient transport, the sequencing is

time step:	0	1	2	3	4	5	…
solve for:	p	•	•	•	•	•	…
	•	U	U	U	U	U	…

For steady-state flow and steady-state transport, the sequencing is

time step:	0	1
solve for:	p	•
	•	U

The only exception to the cycling is that for nonsteady cases, both unknowns are solved for on the first time step, as shown in the case for NPCYC=3, NUCYC=1, above, and on the last time step, irrespective of the values of NPCYC and NUCYC.

It is computationally advantageous to avoid unnecessarily reconstructing the U equation and, when the direct solver is used, to avoid the U matrix decomposition steps by allowing solution by back-substitution only. This is begun on the second time step solving for U only after the step on which a solution is obtained for both p and U. To do this, the matrix coefficients (including the time step) must remain constant. Thus, nonlinear variables and fluid velocity are held constant with values used on the first time step for U after the step for p and U. For example, when NPCYC=1, NUCYC=6:

time step:	1	2	3	4	5	6	7	8	9	10	11	12	…
solve for:	p	•	•	•	•	p	•	•	•	•	•	p	…
	U	U	U	U	U	U	U	U	U	U	U	U	…
	|	constant values			|		|	constant values			|	
		|	back-substitute		|			|	back-substitute		|	

Note that p and U solutions must be set by the user to occur on time steps when relevant boundary conditions, sources or sinks are set to change in value.

5.6 Observation Output

"Observations" are simulation results that are reported at particular times and locations of interest to the user. In previous versions of SUTRA (2.0 and earlier), observations could be made only at nodes, and only at the beginning/end of time steps. In this version of SUTRA (2.1), observations can be made at any point within the model domain and at any time within the simulated period. SUTRA interpolates results in space using finite-element basis functions to compute pressure (p) and concentration or temperature (U), and the unsaturated functions in subroutine UNSAT to compute saturation (S_w) from the interpolated value of p. Results are interpolated linearly in time between the beginning and end of a time step.

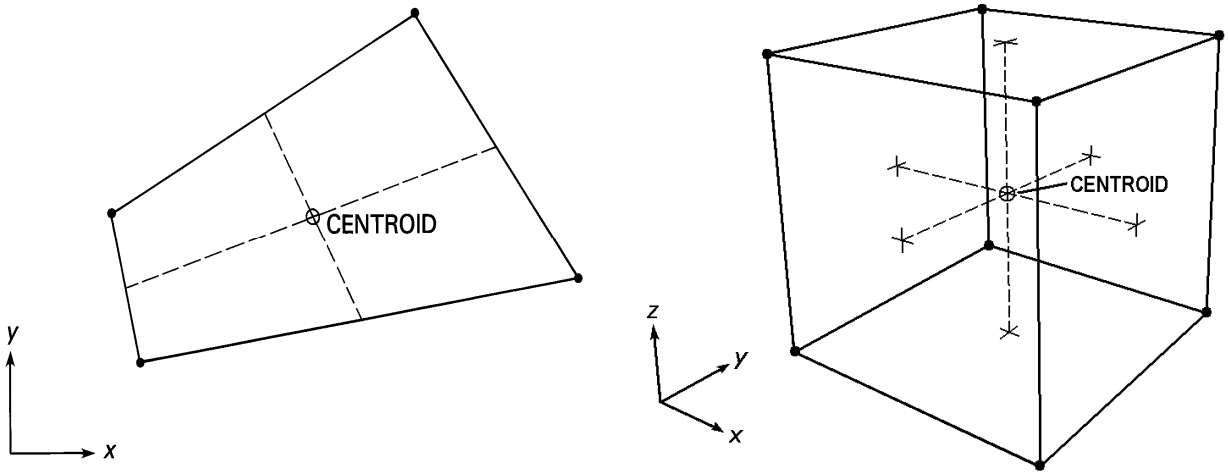
Observation results are written to **".obs"** and/or **".obc"** output files. Both report the same information, but they are organized differently. In **".obs"** files, all results computed at a given time are listed across the page on one line, unless a line wrapping option is invoked to avoid generating extremely long lines of output. The organization of **".obc"** files is analogous to that of **".nod"** and **".ele"** files: results computed at a given time are preceded by a header and are listed one observation point per line.

The timing of observation output is controlled by schedules defined in dataset 6 of the **".inp"** file. Observation points are defined in dataset 8D of the **".inp"** file. The definition of each observation point includes its name, its spatial coordinates, the name of the schedule that controls its output, and an output format (OBS or OBC). Observations that have the same schedule and output format are written to the same output file, so the number of **".obs"** and/or **".obc"** output files depends on the number of combinations of schedule and output format encountered in dataset 8D. SUTRA automatically generates the required number of output files, assigning filenames derived from base filenames specified by the user in file "SUTRA.FIL".

5.7 Velocity Calculation for Output

The velocities employed in the numerical solution of the fluid mass, and the solute mass or energy balances are those calculated at the Gauss points in each element (as described in section 4.6, "Consistent Evaluation of Fluid Velocity.") For purposes of output, however, only one velocity value per element is made available. This is the velocity at the element centroid, which is defined as the point whose coordinates are the arithmetic average of the coordinates of the four (2D) or eight (3D) nodes at the corners of the element; for example, the x-coordinate of the centroid is the average of the x-coordinates of the corner nodes. The centroid can equivalently be defined as the point in the element where the lines connecting the midpoints of opposite sides intersect, as shown in Figure 5.2 for both 2D and 3D elements.

The velocity at the centroid of an element is calculated by taking the average of the velocities (in global coordinates) at the four (2D) or eight (3D) Gauss points; for example, the x-component of velocity at the centroid is the average of the x-components of the velocity at the Gauss points. This process gives the "true" velocity at the centroid that would be calculated employing the consistent velocity approximation evaluated at this point in the element. In 2D, this may be seen by setting $\xi=\eta=0$ in (4.154) and (4.155). **Note that this velocity calculation is based on previous, not current, pressures and concentrations or temperatures (i.e., from the previous iteration or time step).**



(a) (b)

Figure 5.2. (a) 2D finite element in global coordinates (x,y) with element centroid. The dashed lines connect the midpoints of the element sides. (b) 3D finite element in global coordinates (x,y,z) with element centroid. Crosses (+) indicate centroids of element faces.

5.8 Budget Calculations

A fluid mass and solute mass or energy budget provides information on the quantities of fluid mass and either solute mass or energy entering or exiting the simulated region. Generally, it is not intended as a check on numerical accuracy, but rather as an aid in interpreting simulation results. When an iterative matrix solver is used, the discrepancy in the budget may be used to judge iterative convergence.

The fluid budget is calculated based on the terms of the integrated-discretized fluid mass balance, (4.96), as approximated in time according to (4.112). After the solution to a time step makes available p_i^{n+1} and U_i^{n+1}, the time derivatives of these, dp_i/dt and dU_i/dt, are calculated according to (4.104) and (4.140).

The total rate of change in stored fluid mass in the region due to pressure changes over the recent time step is:

$$\sum_{i=1}^{NN} AF_i^{n+1} \frac{dp_i}{dt} \qquad [M/s] \tag{5.11}$$

where AF_i is defined in (4.97), and the total rate of change in stored fluid due to changes in concentration or temperature is:

$$\sum_{i=1}^{NN} CF_i^{n+1} \frac{dU_i}{dt} \qquad [M/s] \tag{5.12}$$

where CF_i is defined in (4.98). The sum of (5.11) and (5.12) gives the total rate of change of fluid mass in the entire region.

Fluid sources, Q_i^{n+1}, may vary with time, and those that do vary are reported by the budget at each source node. The sum of Q_i^{n+1}

$$\sum_{i=1}^{NN} Q_i^{n+1} \qquad [M/s] \tag{5.13}$$

gives the total rate of fluid mass change due to all sources and sinks of fluid mass, as well as to specified fluxes across boundaries. Fluid sources due to specified pressure conditions, $Q_{BC_i}^{n+1}$, usually vary with time and are also reported by the budget at each node. This source is calculated at each node from (4.111). The sum of $Q_{BC_i}^{n+1}$

$$\sum_{i=1}^{NN} Q_{BC_i}^{n+1} \qquad [M/s] \tag{5.14}$$

gives the total rate of fluid mass change in the entire region due to inflows and outflows at all specified pressure nodes.

The sum of (5.13) and (5.14) should be close to the value given by the sum of (5.11) and (5.12). These may be expected to match better when iterations for nonlinearity have been used and convergence was achieved, because the budget is calculated for a time step using only one iteration with the (n+1) time level values of nonlinear coefficients, and the solution is obtained with coefficients based on projected values of p and U.

The solute mass or energy budget is calculated based on the terms of the total balance (2.47) (prior to subtraction of the terms that appear in the fluid mass balance done in equations (2.48) through (2.52)). This allows simpler physical interpretation of the terms (particularly the fluid source terms) in the balance than would the similar terms of the fluid-conservative balance form (2.52). An integrated-discretized balance, similar to that given in (4.132) for the fluid-conservative form, and as approximated in the time according to (4.142), is presented in SUTRA's solute mass or energy budget.

The total rate of change in stored solute mass or energy in the region due to change in concentration or temperature over the recently computed time step is

$$\sum_{i=1}^{NN} AT_i^{n+1} \frac{dU_i}{dt} \qquad [M_s/s \text{ or } E/s] \tag{5.15a}$$

where AT_i^{n+1} is calculated from (4.133) using U_i^{n+1} in all coefficients requiring a value (including adsorption isotherms for $c_s = \kappa_1$). In reporting this portion of the budget, a separate value is given for the sum of the portion stemming from $(\varepsilon s_w \rho c_w)$ and for $(1-\varepsilon)\rho_s c_s$. The former sum relates to the rate of solute mass or energy change in the fluid, and the latter relates to the change in the solid-immobile portion. The total rate of change in stored solute mass or energy in the region due to change in stored fluid mass (see first term of (2.48)) is

$$\sum_{i=1}^{NN} c_w U_i^{n+1} \left(AF_i^{n+1} \frac{dp_i^{n+1}}{dt} + CF_i^{n+1} \frac{dU_i^n}{dt} \right) \qquad [M_s/s \text{ or } E/s] \tag{5.15b}$$

The total rate of first-order solute mass production in the fluid is calculated as

$$\sum_{i=1}^{NN} GT_i^{n+1}\, U_i^{n+1} \qquad [M/s] \tag{5.16a}$$

and at the rate of first-order adsorbate production is calculated as

$$\sum_{i=1}^{NN} G_sTL_i^{n+1}\, U_i^{n+1} + G_sTR_i^{n+1} \qquad [M/s] \tag{5.16b}$$

where GT_i and G_sTL_i and G_sTR_i are defined by (4.136a-c), and all isotherms are based on U_i^{n+1}. Fluid and adsorbate rates are reported separately by the budget. These terms have no analogy for energy transport. The terms of zero-order production of solute and adsorbate mass or energy production in the fluid and solid matrix are

$$\sum_{i=1}^{NN} ET_i^{n+1} \qquad [M_s/s \text{ of } E/s] \tag{5.17}$$

where ET_i is defined by (4.137) and the fluid and immobile phase production rates are reported separately by the budget.

Solute mass and energy sources and sinks due to inflowing or outflowing fluid mass may vary with time and are reported by the budget at each fluid source node and at each specified pressure node. These are summed separately for the entire region:

$$\sum_{i=1}^{NN} Q_i^{n+1} c_w\, U_i^{*n+1} \qquad [M_s/s \text{ or } E/s] \tag{5.18}$$

$$\sum_{i=1}^{NN} Q_{BC_i}^{(n+1)^*} c_w\, U_{BC_i}^{n+1} \qquad [M_s/s \text{ or } E/s] \tag{5.19}$$

where U_i^{*n+1} and $U_{BC_i}^{n+1}$ take on the user-specified values of U for fluid inflows, and the U value of the ambient system fluid for outflows. The $(n+1)^*$ level indicates that Q_{BC} is evaluated on the first iteration at the previous time level (n) and on subsequent iterations, at the most recent iteration level for the present time step. These sums give the total rate of change of solute mass or energy in the entire system due to these fluid sources and sinks.

The diffusive-dispersive sources of solute mass or energy (4.144) are summed for the entire system and are also reported by node as they may vary with time:

$$\sum_{i=1}^{NN} \psi_{IN_i}^{n+1} \qquad [M_s/s \text{ or } E/s] \tag{5.20}$$

Finally, solute mass or energy sources due to specified concentration or temperature conditions, $\psi_{BC_i}^{n+1}$, usually vary with time and are calculated at each node from (4.143). These are additionally reported by the budget at each node. The sum of these sources:

$$\sum_{i=1}^{NN} \psi_{BC_i}^{n+1} \qquad \left[M_s/s \text{ or } E/s\right] \qquad (5.21)$$

gives the total rate of solute mass or energy change in the entire region due to fluxes of solute or energy at all specified concentration or temperatue nodes.

The sum of (5.16a), (5.16b), (5.17), (5.18), (5.19), (5.20) and (5.21) should be close to the value given by (5.15a) and (5.15b). These values may be expected to match best when nonlinearity iterations have been used and convergence achieved, because the budget is calculated for a time step with only one iteration with all information at the (n+1) time level, whereas the solution is obtained using nonlinear coefficients based on projections of p and U.

In addition to providing the net changes in storage, net sources and inflows, and net production of fluid and energy or solute, the budget also separately provides the total of all positive contributions and the total of all negative contributions for each of these quantities. For example, the net inflow of fluid at specified pressure nodes is provided as the sum over all nodes at which there is an inflow (a positive flow) and the sum over all nodes at which there is an outflow (a negative flow).

Based on the sums described above, SUTRA computes and reports absolute and relative balance errors for fluid mass and for energy or solute mass. For fluid mass, the absolute error is defined as

$$\text{absolute fluid mass balance error} = S_f - F_f \qquad (5.22)$$

where

$$S_f = \text{net rate of change in stored fluid} \qquad (5.23)$$

and

$$F_f = \text{net rate of gain/loss of fluid through inflows/outflows} \qquad (5.24)$$

The relative error is then defined as

$$\text{relative fluid mass balance error} = 100 * (S_f - F_f)/A_f \qquad (5.25)$$

where

$$A_f = \left(S_f^+ - S_f^- + F_f^+ - F_f^-\right)/2 \qquad (5.26)$$

The superscripts "+" and "-" indicate the summed positive and negative contributions, respectively, to S_f and F_f. By definition, S_f^+ and F_f^+ are positive, and S_f^- and F_f^- are negative, so A_f is always positive and is a measure of the overall fluid mass balance "activity" (storage, release, inflow, and outflow of fluid) in the system. The relative error is expressed as a percentage of this activity.

For energy or solute mass (including adsorbate), the absolute error is defined as

$$\text{absolute energy or solute mass balance error} = S_t - P_t - F_t \qquad (5.27)$$

where

S_t = net rate of change in stored energy or solute mass (5.28)

P_t = net rate of production/decay of energy or solute mass (5.29)

and

F_t = net rate of gain/loss of energy or solute mass (5.30)

through inflows/outflows and sources/sinks

The relative error is then defined as

$$\text{relative energy or solute mass balance error} = 100 * (S_t - P_t - F_t)/A_t \quad (5.31)$$

where

$$A_t = \left(S_t^+ - S_t^- + P_t^+ - P_t^- + F_t^+ - F_t^-\right)/2 \quad (5.32)$$

The superscripts "+" and "-" indicate the summed positive and negative contributions, respectively, to S_t, P_t, and F_t. By definition, S_t^+, P_t^+, and F_t^+ are positive, and S_t^-, P_t^-, and F_t^- are negative, so A_t is always positive and is a measure of the overall energy or solute mass balance "activity" (storage, release, production, decay, inflows, outflows, sources, and sinks of energy or solute mass) in the system. The relative error is expressed as a percentage of this activity.

5.9 Program Structure and Program Unit Descriptions

SUTRA is structured in a modular, top-down programming style that allows for code readability, ease in tracing logic, and hopefully, ease in eventual modifications. Each subroutine carries out a primary function that is clearly distinguished from all other program functions. User-required program changes are limited to coding portions of a subroutine that is used to control time-dependent sources and boundary conditions (when they are used) and a subroutine that sets the unsaturated flow functions when unsaturated flow is simulated. The program is commented to aid in tracing logic.

SUTRA is written in FORTRAN-90 and takes advantage of dynamic array allocation; however, few structures are used that are not compatible with FORTRAN-77. The code runs accurately when it employs "double-precision" real variables (64 bit words with 47 bit mantissa) with a precision of about 15 significant figures, and 32 bit word integer variables. Should the code require modification to run on machines with other word lengths or other bit to byte ratios, the number of significant figures in a real variable should be preserved, if not increased.

Input and output are also somewhat modularized. Input is through three data files consisting of list-directed records. The input first file, called "**SUTRA.FIL**", contains a list of the names of the remaining input and output files (and, optionally, their corresponding FORTRAN unit numbers). The second input file, typically given the filename extension "**.ics**", contains only data on initial conditions for p and U at the nodes. The third input file, typically given the filename extension "**.inp**", contains all other input data required for a simulation.

Output is to as many as seven types of data files:

- A single file, typically given the filename extension "**.rst**", that receives the result of the final time step in a format equivalent to that of the "**.ics**" file (with some additional information), for later use as the initial conditions file if the simulation is to be restarted.
- A single file, typically given the filename extension "**.nod**", that receives nodewise results (node coordinates, pressures, concentrations, and saturations) at each node for a user-specified sequence of time steps.
- A single file, typically given the filename extension "**.ele**", that receives elementwise results (element centroid coordinates and velocity components) at each element centroid for a user-specified sequence of time steps. In the "**.nod**" and "**.ele**" files, output is arranged in columns to facilitate importing the results into postprocessing software.
- A file or series of files, typically given the filename extension "**.obs**", that receives results, called observations, for a set of points in space and a sequence of times or time steps specified by the user.
- A file or series of files, typically given the filename extension "**.obc**", that receives results, called observations, for a set of points in space and a sequence of times or time steps specified by the user. Offers an alternative output format to the **".obs"** file.
- A single file, typically given the filename extension "**.lst**", that can list (at the user's option) a variety of information, including a summary of the input parameters and fluid and solute mass budget calculations, as shown in Figure 5.3.
- A single file, typically given the filename extension "**.smy**", that summarizes simulation progress, receives convergence and error information; its default name is "**SUTRA.SMY**".

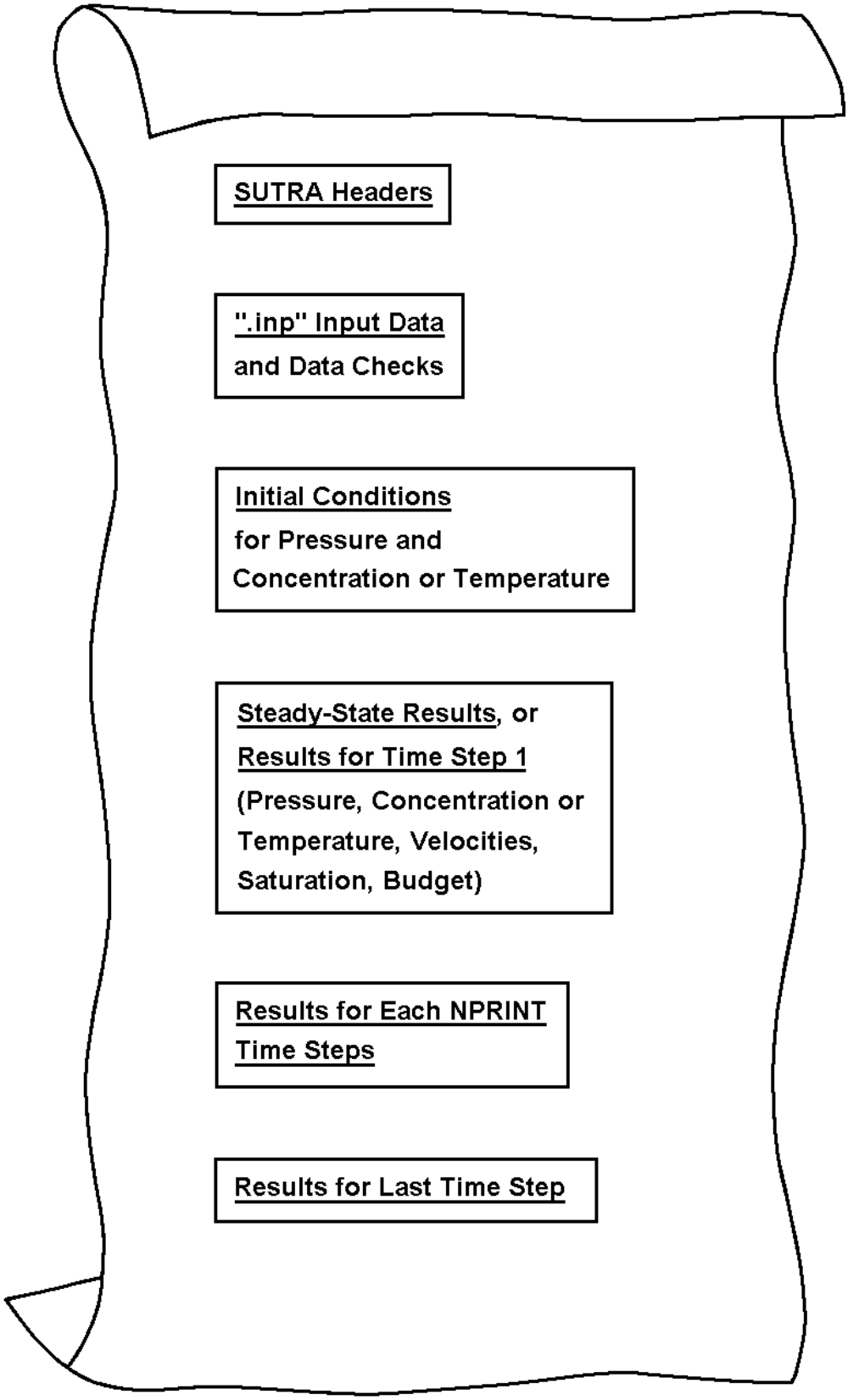


Figure 5.3. Schematic of SUTRA output to the ".lst" file.

The input and output files are summarized below:

	Filename or extension	Contents
Input files	SUTRA.FIL	file assignments
	.inp	main input
	.ics	initial conditions
Output files	.smy (SUTRA.SMY)	simulation summary
	.lst	main results listing
	.rst	restart file (same format as .ics file)
	.nod	nodewise results
	.ele	elementwise results
	.obs	observation results
	.obc	observation results (alternative format)

The main logic flow of the program is straightforward. A schematic diagram of the code is shown in Figure 5.4. The main program sets up dimensions and calls the main control routine, SUTRA, which cycles the program tasks by calling most of the remaining subroutines in sequence. Subroutines are named to describe their main function. The remainder of this section describes the SUTRA main program and each of the subroutines and subprograms in alphabetical order.

Main Program SUTRA_MAIN

- Purpose:

  1. To read in input data, including the problem dimensions.

  2. To compute the dimensions of and dynamically allocate memory for the various arrays used by SUTRA.

  3. To start and stop the simulation.

- Calls to:

  BANWID, BOUND, CONNEC, DIMWRK, FINDL2, FINDL3, FOPEN, INDAT0, INDAT1, INDAT2, PRSWDS, PTRSET, READIF, SOURCE, SUTERR, SUTRA, ZERO

- Uses:

  ALLARR, EXPINT, PTRDEF, SCHDEF

- Description:

  The main program reads data from the ".inp" input file and initial conditions from the ".ics" input file. It uses the input data to compute (in part, by way of a call to subroutine DIMWRK) the dimensions of the various arrays that are declared in subroutine SUTRA; these dimensions are passed through COMMON blocks. It then passes control to subroutine SUTRA.

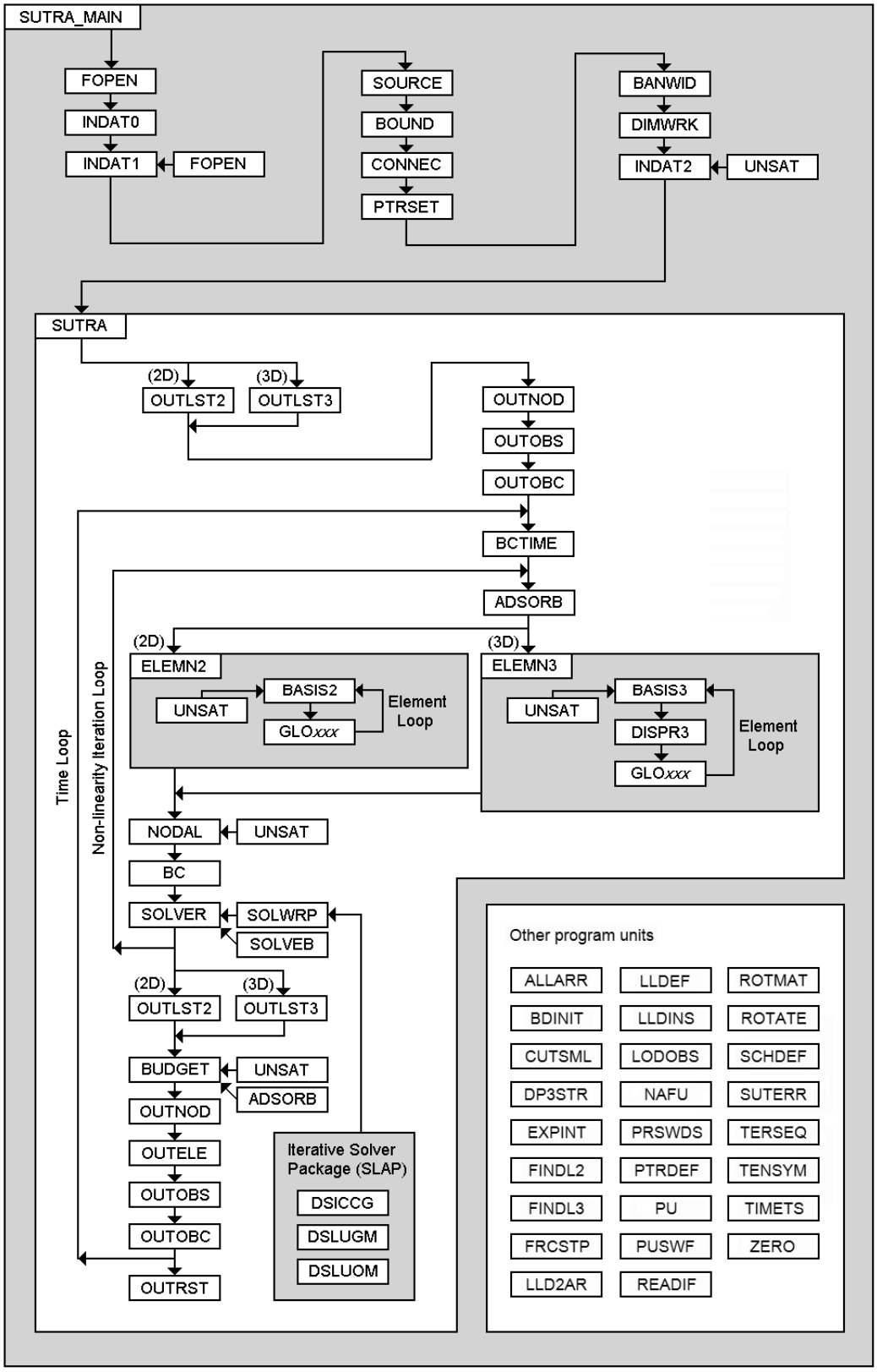


Figure 5.4. SUTRA logic flow. “Other program units” are subroutines, functions, modules, and block subprograms that are not integral to the main logic flow; they are listed separately for the sake of clarity and completeness. GLO*xxx* refers to either GLOBAN or GLOCOL.

Subroutine ADSORB

- Purpose:

  To calculate and supply values from adsorption isotherms to the simulation.

- Called by:

  BUDGET, SUTRA

- Description:

  ADSORB calculates the sorption coefficient, $\kappa_{l_i}^{n+1}$ (called CS1), and s_L (called SL) and s_R (called SR), which are used in calculating adsorbate concentrations, U_s, depending on the particular isotherm chosen: linear, Freundlich or Langmuir. The calculations are based on the description given in section 4.7, "Temporal Evaluation of Adsorbate Mass Balance." ADSORB is called once per time step for U, when sorption is employed in the simulation.

Module ALLARR

- Purpose:

  1. To declare the main allocatable arrays.
  2. To define the derived type OBSDAT.

- Used by:

  Main program, INDAT1, LODOBS, OUTOBC, OUTOBS, PTRSET, SUTRA, TERSEQ

- Description:

  ALLARR declares the main allocatable arrays used by SUTRA and defines the derived type OBSDAT, which is used in the declaration of array OBSPTS, which holds observation point information.

Subroutine BANWID

- Purpose:

  To calculate the bandwidth of the mesh.

- Called by:

  Main program

- Description:

  BANWID checks the incidence array, IN, in all elements for the maximum difference in node numbers contained in an element. This value, NDIFF, is used to calculate the bandwidth, NBL.

Subroutine BASIS2

- Purpose:

  To calculate values of basis functions, weighting functions, their derivatives, Jacobians, and coefficients at a point in a quadrilateral element for 2D meshes.

- Called by:

  ELEMN2

- Calls to:

  UNSAT

- Description:

  BASIS2 receives the coordinates of a point in an element in local coordinates (ξ, η), denoted (XLOC,YLOC) in the routine. At this point, BASIS2 determines the following: values of the four basis functions and their derivatives in each local coordinate direction, elements of the Jacobian matrix, the determinant of the Jacobian matrix, elements of the inverse Jacobian matrix, and if required, four values of the asymmetric weighting function (one for each node) and their derivatives. In addition, the derivatives are transformed to global coordinates and passed out to ELEMN2. Values of nodewise-discretized parameters, the local and global velocity, and parameters dependent on p or U are calculated at this location in the element. Unsaturated parameters are obtained by a call to UNSAT. The calculations are based on section 4.1 “Basis and Weighting Functions”, 4.2 “Coordinate Transformations,” and 4.6 “Consistent Evaluation of Fluid Velocity.”

Subroutine BASIS3

- Purpose:

  To calculate values of basis functions, weighting functions, their derivatives, Jacobians, and coefficients at a point in a hexahedral element for 3D meshes.

- Called by:

  ELEMN3

- Calls to:

  UNSAT

- Description:

  BASIS3 receives the coordinates of a point in an element in local coordinates (ξ, η, ζ), denoted (XLOC,YLOC,ZLOC) in the routine. At this point, BASIS3 determines the following: values of the eight basis functions and their derivatives in each local coordinate direction, elements of the Jacobian matrix, the determinant of the Jacobian matrix, elements of the inverse Jacobian matrix, and if required, eight values of the asymmetric weighting function (one for each node) and their derivatives. In addition, the derivatives are

transformed to global coordinates and passed out to ELEMN3. Values of nodewise-discretized parameters, the local and global velocity, and parameters dependent on p or U are calculated at this location in the element. Values of parameters dependent on p or U are calculated at this location. Unsaturated parameters are obtained by a call to UNSAT. The calculations are based on section 4.1 "Basis and Weighting Functions", 4.2 "Coordinate Transformations," and 4.6 "Consistent Evaluation of Fluid Velocity."

Subroutine BC

- Purpose:

  1. To implement specified pressure node conditions in the matrix equations.

  2. To implement specified temperature or concentration node conditions in the matrix equations.

- Called by:

  SUTRA

- Description:

  The source terms involving ν_{p_i} in (4.112) are added to fluid balance matrix equation to implement specified p nodes. The unified energy-solute mass balance is modified by the addition of a source, QPL (calculated with the most recent p solution by subroutine SUTRA) with concentration or temperature, UBC. The source terms involving ν_{U_i} in (4.145) are added to the energy-solute mass balance matrix equation in order to implement specified U nodes.

Subroutine BCTIME

- Purpose:

  A user-programmed routine in which time-dependent sources and boundary conditions are specified.

- Called by:

  SUTRA

- Description:

  BCTIME is called on each time step when a time-dependent source or boundary condition is specified by the user. It allows the value of a source or boundary condition to be changed on any or all time steps.

  BCTIME is divided into four sections. The first section allows the user to specify either time-dependent pressure and concentration or temperature of an inflow, or both, at specified pressure nodes (PBC or UBC). The second section allows user specification of time-dependent U at specified concentration/temperature nodes. The third section allows user

specification of time-dependent fluid source or source concentration/temperature. The fourth section allows user-specification of time-dependent solute mass or energy sources.

The current time step number, IT, and current time (at the end of the present time step) in various units are available for use in the user-supplied programming. The user may program in any convenient way through data statements, calls to other programs, logical structures, "read" or "write" statements, or other preferred methods of specifying the time variability of sources or specified p and U conditions. More information may be found in section 7.5, "User-Supplied Programming."

Block-data Subprogram BDINIT

- Purpose:

  To initialize variables named in COMMON blocks.

- Description:

  BDINIT is a block-data subprogram that initializes certain variables named in COMMON blocks.

Subroutine BOUND

- Purpose:

  1. To read specified pressure node numbers and pressure values, check the data, and print information.

  2. To read specified concentration or temperature node numbers and the values, check the data, and print information.

  3. To set up pointer arrays that track the specified p and U nodes for the simulation.

- Called by:
  Main program

- Calls to:

  READIF, SUTERR

- Description:

  BOUND reads and organizes, checks and prints information on specified p nodes and specified U nodes. The pressure information read is node number, pressure value and U value of any inflow at this node. If there are NPBC specified pressure nodes, the above information becomes the first NPBC values in vectors IPBC, PBC and UBC. The specified U information read is node number and U value. If there are NUBC specified concentration nodes, the above information begins in the (NPBC+1) position of IUBC and UBC, and ends in the (NUPBC+NUBC) position of IUBC and UBC. This is shown below:

$$
\begin{array}{lcccccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & \\
\text{IPBC} & (& x & x & x & x & x & x & & & & &) \\
 & 1 & 2 & 3 & 4 & 5 & 6 & & & & & & \\
\text{PBC} & (& x & x & x & x & x & x & & & & &) \\
 & x & x & x & x & x & x & y & y & y & y & & \\
\text{UBC} & (& 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 &) \\
 & & & & & & & & y & y & y & y & & \\
\text{IUBC} & (& 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 &)
\end{array}
$$

where "x " refers to specified p information, and "y" refers to specified U information.

Counts are made of each type of specification and are checked against NPBC and NUBC for correctness. A zero node number ends the data set for p and then for U. One blank element is left at the end of each of these arrays in case there are no specified p or U nodes. The first NPBC elements of IUBC are blank. These arrays are used primarily by subroutines BC and BUDGET.

Subroutine BUDGET

- Purpose:
  1. To calculate and output a fluid mass budget on each time step with output.
  2. To calculate and output a solute mass or energy budget on each time step with output.

- Called by:

  SUTRA

- Calls to:

  ADSORB, UNSAT

- Description:

  BUDGET calculates and outputs a fluid mass, solute mass or energy budget on each output time step for whichever of p and/or U are solved for on the just-completed time step. The calculations are done as described in section 5.8 "Budget Calculations."

Subroutine CONNEC

- Purpose:

  To read, output, and organize node incidence data.

- Called by:

  SUTRA

- Calls to:

  READIF, SUTERR

- Description:

  CONNEC reads the nodal incidence list, which describes how nodes are connected. The data are organized as an array, IN, which contains the ordered set of node numbers (four in 2D; eight in 3D) in each element in order of element number. Thus, for a 2D simulation, the ninth through twelfth values in IN are the four nodes in element number three. Array IN is passed to a number of program units and is used in computations in BANWID, BASIS2, BASIS3, ELEMN2, ELEMN3, FINDL2, FINDL3, GLOBAN, GLOCOL, OUTELE, PTRSET, PU, and SUTRA.

Function CUTSML

- Purpose:

  To set numbers of magnitude less than 1.D-99 to zero.

- Called by:

  SUTRA

- Description:

  If the absolute value of the argument is less than 1.D-99, CUTSML returns zero; otherwise, it returns the argument unchanged. CUTSML is used to round down to zero quantities that are small enough to require a three-digit exponent on output, thereby avoiding potential problems caused by FORTRAN omitting the exponent specifier “E” or “D” when a three-digit exponent is written.

Subroutine DIMWRK

- Purpose:

  To return dimensions for the iterative solver work arrays.

- Called by:

  Main program

- Description:

  DIMWRK computes the dimensions of the integer and floating-point work arrays used by the iterative matrix solvers.

Subroutine DISPR3

- Purpose:

  1. To compute longitudinal and transverse dispersivities for 3D simulations.

  2. To return the angles of the velocity vector with respect to the principal permeability directions.

- Called by:

  ELEMN3

- Calls to:

  ROTATE, ROTMAT, TENSYM

- Description:

  DISPR3 computes longitudinal and transverse dispersivities using an ad hoc, 3D anisotropic dispersion model that is a generalization of the 2D SUTRA dispersion model (see section 2.5). Three dispersivities are computed: AL, the longitudinal dispersivity, and AT1 and AT2, the two transverse dispersivities.

Function DP3STR

- Purpose:

  To return three double-precision numbers in the form of a string.

- Called by:

  OUTOBC, OUTOBS

- Description:

  DP3STR is used by subroutines OUTOBC and OUTOBS to facilitate interpolation and output of observation data.

Subroutine ELEMN2

- Purpose:

  1. To carry out all 2D elementwise calculations required in the matrix equations.
  2. To calculate 2D element centroid velocities for output.

- Called by:

  SUTRA

- Calls to:

  BASIS2, GLOBAN, GLOCOL, SUTERR

- Description:

  ELEMN2 undertakes a loop through all the elements in a mesh. For each element, subroutine BASIS2 is called four times, once for each Gauss point. BASIS2 provides basis function information, and values of coefficients and velocities at each Gauss point, all of which is saved by ELEMN2 for use in calculations for the present element.

  Gaussian integration (two by two points) as described in section 4.3, is carried out for each integral in the fluid mass balance ((4.101) and (4.102)), and for each integral in the unified energy and solute mass balance ((4.138) and (4.139)). The portion of cell volume within the present element for node I, VOLE(I), is calculated with the fluid balance integrals. The values of the integrals are saved either as four-element vectors or as four-by-four arrays. Separate (nearly duplicate) sections of the integration code employ either basis functions for weighting or asymmetric weighting functions.

  The vectors and arrays containing the values of integrals over the present element are passed to subroutine GLOBAN or GLOCOL for addition to the global matrix equation (assembly process).

Subroutine ELEMN3

- Purpose:

  1. To carry out all 3D elementwise calculations required in the matrix equations.
  2. To calculate 3D element centroid velocities for output.

- Called by:

  SUTRA

- Calls to:

  DISPR3, BASIS3, GLOBAN, GLOCOL, SUTERR

- Description:

  ELEMN3 undertakes a loop through all the elements in a mesh. For each element, subroutine BASIS3 is called eight times, once for each Gauss point. BASIS3 provides basis function information, and values of coefficients and velocities at each Gauss point, all of which is saved by ELEMN3 for use in calculations for the present element.

  Gaussian integration (two by two by two points) as described in section 4.3, is carried out for each integral in the fluid mass balance ((4.99) and (4.100)), and for each integral in the

unified energy and solute mass balance ((4.134) and (4.135)). The portion of cell volume within the present element for node I, VOLE(I), is calculated with the fluid balance integrals. The values of the integrals are saved either as eight-element vectors or as eight-by-eight arrays. Separate (nearly duplicate) sections of the integration code employ either basis functions for weighting or asymmetric weighting functions.

The vectors and arrays containing the values of integrals over the present element are passed to subroutine GLOBAN or GLOCOL for addition to the global matrix equation (assembly process).

Module EXPINT

- Purpose:

  To provide explicit interfaces for the procedures that need them.

- Used by:

  Main program, FOPEN, INDAT0, INDAT1, OUTELE, OUTNOD, OUTOBC, OUTOBS, SUTRA

- Uses:

  LLDEF

- Description:

  EXPINT provides an explicit interface for subroutines LLD2AR and LLDINS and subroutines PUSWF and DP3STR.

Subroutine FINDL2

- Purpose:

  To determine whether a given point in 2D global coordinates is contained within a given 2D element.

- Called by:

  Main program

- Description:

  FINDL2 determines whether a given point in 2D global coordinates is contained within a given element by computing the local coordinates of the point and checking whether each coordinate lies within the range -1 to 1. The point is contained within the element if and only if both coordinates are within this range.

Subroutine FINDL3

- Purpose:

  To determine whether a given point in 3D global coordinates is contained within a given 3D element.

- Called by:

  Main program

- Description:

  FINDL3 determines whether a given point in 3D global coordinates is contained within a given element by computing the local coordinates of the point and checking whether each coordinate lies within the range -1 to 1. The point is contained within the element if and only if all three coordinates are within this range.

Subroutine FOPEN

- Purpose:

  1. To open the file "SUTRA.FIL" and read in unit numbers and file names.
  2. Assign unit numbers to SUTRA input and output files.
  3. Determine files needed for observation output.
  4. To check for the existence of and open files used in the SUTRA simulation.

- Called by:

  Main program

- Calls to:

  SUTERR

- Uses:

  EXPINT, SCHDEF

- Description:

  FOPEN is called twice by the main program. On the first call, FOPEN first opens the file "SUTRA.FIL", from which it reads the Fortran unit numbers and associated filenames to be used during the SUTRA run. It then assigns unit numbers, checks for the existence of the input files, and opens the input and output files, except for the observation output files. On the second call, FOPEN determines how many files are needed for observation output, assigns filenames and unit numbers, and opens the observation output files.

Function FRCSTP

- Purpose:

  To return the fractional time step that corresponds to a given time.

- Called by:

  INDAT0

- Uses:

  LLDEF, SCHDEF

- Description:

  FRCSTP uses the "TIME_STEPS" schedule, which controls time stepping, to determine the time step number that corresponds to a given time. If the time falls within a time step (as opposed to the beginning/end of a time step), a fractional time step number is returned. For example, the time step number that corresponds to the time half-way between the end of time step 1 and the end of time step 2 is 1.5.

Subroutine GLOBAN

- Purpose:

  To assemble elementwise integrations into the global banded matrix form used by the direct (Gaussian elimination) solver.

- Called by:

  ELEMN2, ELEMN3

- Description:

  GLOBAN carries out the sum over elements of integrals evaluated over each element by ELEMN2 or ELEMN3 (as suggested in 2D by relation (3.23)). Both the matrix and right-hand-side vector terms involving integrals in the discretized governing equations (4.112) and (4.145) are constructed.

Subroutine GLOCOL

- Purpose:

  To assemble elementwise integrations into the global "column" matrix form used by the SLAP iterative solvers.

- Called by:

  ELEMN2, ELEMN3

- Description:

  GLOCOL carries out the sum over elements of integrals evaluated over each element by ELEMN2 or ELEMN3 (as suggested in 2D by relation (3.23)). Both the matrix and right-hand-side vector terms involving integrals in the discretized governing equations (4.112) and (4.145) are constructed.

Subroutine INDAT0

- Purpose:

  To input, output, and organize a portion of the ".inp" input data.

- Called by:

  Main program

- Calls to:

  LLD2AR, LLDINS, READIF, SUTERR

- Uses:

  EXPINT, LLDEF, SCHDEF

- Description:

  INDAT0 reads data from the ".inp" file (datasets 5 – 7) that are needed by the main program to compute array dimensions before passing control to subroutine SUTRA. It also constructs the schedules that control time stepping and observation output.

Subroutine INDAT1

- Purpose:

  1. To read simulation and mesh data from the ".inp" data file, and output this information.
  2. To initialize some variables and carry out minor calculations.

- Called by:

  Main program

- Calls to:

  FOPEN, LLDINS, READIF, SUTERR, ROTMAT, TENSYM

- Uses:

  ALLARR, EXPINT, LLDEF, SCHDEF

- Description:

  INDAT1 reads a portion of the ".inp" input data file, ending with the elementwise data set. Most information is printed on the ".lst" data file after reading; the amount of output depends on the user choice of long or short output format. Scale factors are multiplied with appropriate input data. Calculations are carried out for a thermal conductivity adjustment and for determination of the components of the permeability matrix $\underline{\underline{k}}$ in each element from k_{max}, k_{min}, and θ (for 2D) or from k_{max}, k_{mid}, k_{min}, θ_1, θ_2, and θ_3 (for 3D).

Subroutine INDAT2

- Purpose:

  1. To read initial conditions from the ".ics" file.
  2. To initialize some arrays.

- Called by:

  Main program

- Calls to:

  READIF, SUTERR, UNSAT, ZERO

- Description:

  INDAT2 reads the ".ics" data file, which contains initial conditions for p and U. The warm-start section reads initial conditions and parameter values of a previous time step, all of which must have been stored by subroutine OUTRST on a previous simulation. For a cold start, INDAT2 reads only initial p and initial U. On a cold start, INDAT2 calls UNSAT for calculation of initial saturation values.

Subroutine LLD2AR

- Purpose:

  To load a linked list of pairs of double-precision numbers into two arrays.

- Called by:

  INDAT0

- Uses:

  LLDEF

- Description:

  LLD2AR is used by subroutine INDAT0 to temporarily transfer schedules, which are stored as linked lists of (time, time step) pairs, to ordinary arrays.

Module LLDEF

- Purpose:

  To define the derived type LLDEF.

- Used by:

  EXPINT, FRCSTP, INDAT0, INDAT1, LLD2AR, LLDINS, OUTOBC, OUTOBS, SCHDEF, SUTRA, TIMETS

- Description:

  LLDEF defines the derived type LLD, which is used to linked lists with three components: two double-precision numbers and a pointer to the next entry in the list.

Subroutine LLDINS

- Purpose:

  To insert a pair of double-precision values into a linked list.

- Called by:

  INDAT0, INDAT1

- Uses:

  LLDEF

- Description:

  LLDINS inserts a pair of double-precision values into a linked list in ascending order based on the first value in the pair.

Subroutine LODOBS

- Purpose:

  To load into an array the indices of observation points to be written.

- Called by:

  OUTOBS

- Uses:

  ALLARR, SCHDEF

- Description:

  LODOBS is used by subroutine OUTOBS to load into an array the indices of observation points to be written to the next line of the current ".obs" output file.

Subroutine NAFU

- Purpose:

  To find the next available FORTRAN unit.

- Called by:

  FOPEN, READIF

- Calls to:

  SUTERR

- Uses:

  SCHDEF

- Description:

  NAFU loops through existing files to find the next available FORTRAN unit number. It is used by subroutines FOPEN and READIF during automatic assignment of unit numbers to input and output files.

Subroutine NODAL

- Purpose:

  To calculate and assemble all nodewise and cellwise terms in the matrix equation.

- Called by:

  SUTRA

- Calls to:

  UNSAT

- Description:

  NODAL undertakes a loop through all nodes in the mesh and calculates values of all cellwise terms. For each node, time derivatives and a fluid source are added to the fluid mass balance matrix equation. The time derivative as well as terms due to fluid sources, production, and boundary fluxes of U are prepared and added to the solute mass/energy balance matrix equation. Subroutine UNSAT is called for unsaturated flow parameters. The terms added by NODAL may be described as the nonintegral terms of (4.96) and (4.132) (except for the specified pressure terms.)

Subroutine OUTELE

- Purpose:

  To print elementwise output, organized in columns, to the ".ele" file.

- Called by:

  SUTRA

- Uses:

  EXPINT, SCHDEF

- Description:

  OUTELE writes element numbers, element centroid coordinates, and elementwise solution data (components of velocity in global coordinates) in a columnwise format that facilitates importing SUTRA output into post-processing software. The content of each column is specified in the ".inp" input file, giving the user some control over the format of the output.

Subroutine OUTLST2

- Purpose:

  To output the following to the ".lst" file for 2D simulations:
  Initial conditions
  Pressure solutions
  Saturation values
  Concentration or temperature solutions
  Steady-state pressure solution
  Fluid velocities (magnitude and direction)

- Called by:

  SUTRA

- Description:

  For 2D simulations, OUTLST2 is the main output routine for writing to the ".lst" file and is used for printing solutions.

Subroutine OUTLST3

- Purpose:

  To output the following to the ".lst" file for 3D simulations:
  Initial conditions
  Pressure solutions
  Saturation values
  Concentration or temperature solutions
  Steady-state pressure solution
  Fluid velocities (magnitude and direction)

- Called by:

  SUTRA

- Description:

  For 3D simulations, OUTLST3 is the main output routine for writing to the ".lst" file and is used for printing solutions.

Subroutine OUTNOD

- Purpose:

  To print nodewise output, organized in columns, to the ".nod" file.

- Called by:

  SUTRA

- Uses:

  EXPINT, SCHDEF

- Description:

  OUTNOD writes node numbers, node coordinates, and nodewise solution data (pressures, concentrations or temperatures, and saturations) in a columnwise format that facilitates importing SUTRA output into postprocessing software. The content of each column is specified in the ".inp" input file, giving the user some control over the format of the output.

Subroutine OUTOBC

- Purpose:

  To print observation data to ".obc" files.

- Called by:

  SUTRA

- Calls to:

  DP3STR, PUSWF

- Uses:

  ALLARR, EXPINT, LLDEF, SCHDEF

- Description:

  OUTOBC writes pressures, concentrations or temperatures, and saturations at observation points to ".obc" files. All observations assigned the OBC output format and the same output schedule are written to the same ".obc" file.

Subroutine OUTOBS

- Purpose:

  To print observation data to ".obs" files.

- Called by:

  SUTRA

- Calls to:

  DP3STR, LODOBS, PUSWF

- Uses:

  ALLARR, EXPINT, LLDEF, SCHDEF

- Description:

  OUTOBS writes pressures, concentrations or temperatures, and saturations at observation points to ".obs" files. All observations assigned the OBS format and the same output schedule are written to the same ".obs" file.

Subroutine OUTRST

- Purpose:

  To store p and U results as well as other parameters in the ".rst" file in a format ready for use as a ".ics" initial conditions file. This file can also act as a backup for restart in case a simulation is unexpectedly terminated before completion.

- Called by:

  SUTRA

- Description:

  OUTRST is called upon completion of each ISTORE time steps of a simulation, where the value of ISTORE is set by the user. OUTRST writes the most recent solution for p and U at the nodes to the ".rst" file, in a format exactly equivalent to that of the ".ics" input data file. Information is also written which is used in a warm start (restart) of the simulation. The results of only the most recent time step are stored in the ".rst" file, as OUTRST rewinds the file each time before writing.

Subroutine PRSWDS

- Purpose:

  To parse character strings into words.

- Called by:

  Main program, SUTERR

- Description:

  PRSWDS parses the character variable STRING into an array, WORDS, that contains the individual words that make up STRING. Words are groups of characters separated by one or more of the single-character delimiter DELIM and/or blanks.

Module PTRDEF

- Purpose:

  To define pointers and arrays needed to construct the IA and JA arrays used by the SLAP solvers.

- Used by:

  Main program, PTRSET

- Description:

  PTRDEF defines two derived types and declares general-purpose linked lists and arrays HLIST and LLIST, which are used by subroutine PTRSET in constructing the IA and JA arrays used by the SLAP solvers.

Subroutine PTRSET

- Purpose:

  To set up pointer arrays needed to specify the matrix structure used by the iterative solvers.

- Called by:

  Main program

- Uses:

  ALLARR, PTRDEF

- Description:

  PTRSET sets up several pointer arrays that are used to define the “column” matrix structure used by the SLAP iterative solvers. The information is first stored in linked lists, then transferred to arrays IA and JA.

Subroutine PU

- Purpose:

  To evaluate p and U at given local coordinates within a 2D or 3D element.

- Called by:

  PUSWF

- Description:

  PU computes p and U by evaluating the basis functions at the specified local coordinates within the element. The calculations are a subset of those performed by subroutines BASIS2 and BASIS3.

Function PUSWF

- Purpose:

  To interpolate p, U, and S_w in time between the beginning and end of a time step.

- Called by:

  OUTOBC, OUTOBS

- Calls to:

  PU, UNSAT

- Description:

  PUSWF interpolates p and U linearly between the beginning and the end of a time step. For unsaturated conditions, it evaluates S_w from the interpolated p value using subroutine UNSAT.

Subroutine READIF

- Purpose:

  1. To read a line from an input file into the character variable INTFIL.
  2. To handle opening and closing of inserted files as necessary.

- Called by:

  Main program, BOUND, CONNEC, INDAT0, INDAT1, INDAT2, SOURCE

- Calls to:

  NAFU, SUTERR

- Uses:

  SCHDEF

- Description:

  READIF reads a line from an input file into the character variable INTFIL, which is used by various SUTRA subroutines as a Fortran "internal file" from which input data are read. This

allows list-directed SUTRA input files to be read and processed strictly line-by-line, which simplifies the identification of errors in the input data. Comment lines and blank lines are skipped, and inserted files are automatically opened and closed as needed.

Subroutine ROTATE

- Purpose:

  To transform a 3D vector by applying a rotation matrix.

- Called by:

  DISPR3

- Description:

  ROTATE transforms a 3D vector {x} to $\{x_p\}$ by applying the rotation matrix [G].

Subroutine ROTMAT

- Purpose:

  To compute the transformation matrix that converts a vector from one Cartesian coordinate system to another.

- Called by:

  DISPR3, INDAT1

- Description:

  ROTMAT computes the transformation matrix [G] that converts vector {v} from coordinate system (x, y, z) to coordinate system (x', y', z') according to {v'}=[G]{v}. The overall transformation is the result of three rotations applied consecutively: A1 = angle of rotation in the x,y-plane, counterclockwise looking down the +z-axis toward the origin; A2 = angle of rotation from the x,y-plane, counterclockwise looking away from the origin, up the axis that, prior to the first rotation, was aligned with the +y-axis; A3 = angle of rotation about the x'-axis (the axis that, prior to any rotations, was aligned with the x-axis), counterclockwise looking down the +x'-axis toward the origin.

Module SCHDEF

- Purpose:

  To define derived types and declare pointers and arrays associated with schedules and observation output.

- Used by:

  Main program, FOPEN, FRCSTP, INDAT0, INDAT1, LODOBS, NAFU, OUTELE, OUTNOD, OUTOBC, OUTOBS, READIF, SUTRA, TERSEQ, TIMETS

- Uses:

  LLDEF

- Description:

  SCHDEF defines derived types and declares pointers and arrays that hold information about schedules. Array SCHDLS holds basic information about each schedule. Arrays OFP, IUNIO, FNAMO, and ONCK78 are specific to observation schedules and output.

Subroutine SOLVEB

- Purpose:

  To directly solve a matrix equation with a nonsymmetric banded matrix.

- Called by:

  SOLVER

- Description:

  SOLVEB expects the matrix band as a vertical rectangular block with the main diagonal in the center column, and minor diagonals in the other columns. The upper left-hand corner and lower right-hand corner of the matrix are blank.

  The first section of the routine carries out an LU decomposition of the matrix, which is saved within the original matrix space. The second section of the routine prepares the right-hand-side for solution and carries out back-substitution with a given right-hand-side vector.

Subroutine SOLVER

- Purpose:

  To call the matrix solver routine specified by the user.

- Called by:

  SUTRA

- Calls to:

  SOLWRP, SOLVEB

- Description:

  SOLVER calls the direct (Gaussian elimination) solver SOLVEB or, by way of SOLWRP, an iterative solver.

Subroutine SOLWRP

- Purpose:

  1. To set up parameters and arrays prior to calling an iterative solver.
  2. To call a solver routine from the iterative solver package.

- Called by:

  SOLVER

- Calls to:

  DSICCG, DSLUGM, DSLUOM

- Description:

  SOLWRP is a "wrapper" for the iterative solver package. It sets up the right-hand-side vector and the solution vector (which contains the initial guess on input to the iterative solver), and calls an individual solver routine. Subroutines DSICCG, DSLUGM, and DSLUOM are called to run the CG, GMRES, and ORTHOMIN solvers, respectively.

Subroutine SOURCE

- Purpose:

  1. To read source node numbers and source values for fluid mass sources and boundary fluxes and for diffusive and productive U sources, as well as fluxes of U at boundaries; to check the data; and to print information.

  2. To set up pointer arrays that track the source nodes for the simulation.

- Called by:

  Main program

- Calls to:

  READIF, SUTERR

- Description:

  SOURCE reads and organizes, checks and prints information on source nodes for fluid mass, and for sources of solute mass or energy. The fluid mass source information read is node number, mass source rate, and U value of any inflowing fluid at this node. If there are NSOP fluid source nodes, the node numbers become the first NSOP values in vector IQSOP. The rates are entered in the element corresponding to the nodes at which they are defined in vectors QIN and UIN, which are of length NN. The source information for U read is node number and solute mass or energy source rate. If there are NSOU source nodes for U, the node numbers become the first NSOU values in IQSOU. Vector QUIN is of length NN and contains the source rates in numerical order by node. Counts are made of each type of source and are checked against NSOP and NSOU for correctness. A blank (zero) node number ends

the data set for QIN and then for QUIN. One blank element is left at the end of IQSOP and IQSOU so that a dimension of one is obtained even when no source nodes exist. These arrays are used primarily in NODAL and BUDGET.

Subroutine SUTERR

- Purpose:

  To handle SUTRA, iterative solver, and Fortran READ errors.

- Called by:

  Main program, BOUND, CONNEC, ELEMN2, ELEMN3, FOPEN, INDAT0, INDAT1, INDAT2, NAFU, READIF, SOURCE, SUTRA

- Calls to:

  PRSWDS, TERSEQ

- Description:

  SUTERR acts as the clearinghouse for errors that occur during a SUTRA run. When an input data error, nonconvergence of iterations to resolve nonlinearities, an iterative solver error, or a Fortran READ error occurs, SUTERR is called. SUTERR reports the error, along with a concise description, and calls subroutine TERSEQ to initiate the termination sequence.

Subroutine SUTRA

- Purpose:

  1. To act as primary control on SUTRA simulation, cycling both iterations and time steps.
  2. To sequence program operations by calling subroutines for output and most program calculations.
  3. To carry out minor calculations.

- Called by:

  Main program

- Calls to:

  ADSORB, BC, BCTIME, BUDGET, ELEMN2, ELEMN3, NODAL, OUTLST2, OUTLST3, OUTNOD, OUTELE, OUTOBC, OUTOBS, OUTRST, SOLVER, SUTERR, ZERO

- Uses:

  ALLARR, EXPINT, LLDEF, SCHDEF

- Description:

  Subroutine SUTRA initializes certain constants and calls OUTLST2 (for 2D) or OUTLST3 (for 3D) to print the initial conditions to the ".lst" file.

  The subroutine decides on cycling parameters if steady state pressures will be calculated, and calls ZERO to initialize arrays. For transient pressure solution steps, time-step cycling parameters are set and a decision is made as to which (or both) of p and U will be solved for on this time step. The decision depends on NPCYC and NUCYC, and subroutine SUTRA sets the switch, ML, as follows:

$$ML = \begin{cases} 0 & \text{solve for both p and U} \\ 1 & \text{solve for p only} \\ 2 & \text{solve for U only} \end{cases}$$

  The switch for steady state flow is ISSFLO, which is set as follows:

$$ISSFLO = \begin{cases} 0 & \text{steady flow not assumed} \\ 1 & \text{steady flow assumed, before pressure time step} \\ 2 & \text{steady flow assumed, after beginning of pressure time step} \end{cases}$$

  Note that time step number, IT, is set to zero for the steady p solution, and increments to one for the first transport time step.

  Subroutine SUTRA increments the simulation clock, TSEC, to the time at the end of the new time step, and shifts new vectors to previous level vectors, which begins the time step. BCTIME is called to set time-dependent sources and boundary conditions if such exist. ADSORB is called if sorption is required. The element-by-element calculations required to construct the matrix equations are carried out by a call to ELEMN2 (for 3D) or ELEMN3 (for 3D). NODAL is called to carry out nodewise and cellwise calculations for the global matrices. BC is called to modify the matrix equations for boundary conditions.

  SOLVE is called for p and or U solution (depending on the value of ML), and if iterations to resolve nonlinearities are underway, convergence is checked. If iterations are continued, control switches back to the step, which shifts new to old vectors, and the sequence of calls is repeated. If no more iterations are required, SUTRA may call OUTLST2 (for 2D) or OUTLST3 (for 3D) to print results to the ".lst" file if these are requested on the present time step. BUDGET is called if budget output is requested to the ".lst" file on this time step. OUTNOD, OUTELE, OUTOBC, and OUTOBS are called to print nodewise results to the ".nod" file, elementwise results to the ".ele" file, and observation data to ".obc" and ".obs" files, respectively, if these are requested on the present time step.

  If more time steps are to be undertaken, control switches back to the step that initializes arrays, and continues down from that point. If the simulation is complete, OUTRST is called if the store option has been selected to set up a ".rst" restart file. At this point, control returns to the main program.

Subroutine TENSYM

- Purpose:

  To transform a symmetric tensor between two Cartesian coordinates systems.

- Called by:

  DISPR3, INDAT1

- Description:

  TENSYM performs the transformation [P]=[G][T], where [P] and [T] are matrices that represent a symmetric tensor in two different Cartesian coordinate systems, and [G] is the rotation matrix that transforms the input coordinate system to the output coordinate system. [T] is a diagonal matrix. The rotation is defined in term of three angles, ANGLE1, ANGLE2, and ANGLE3, which correspond to the angles A1, A2, and A3 defined in the description of subroutine ROTMAT.

Subroutine TERSEQ

- Purpose:

  To gracefully terminate a SUTRA run by deallocating the main allocatable arrays and closing all files.

- Called by:

  Main program, SUTERR

- Uses:

  ALLARR, SCHDEF

- Description:

  TERSEQ is called to execute the termination sequence either by the main program at the successful completion of a simulation or by SUTERR in the event of an error. It deallocates the main allocatable arrays, closes all files, and stops the run.

Subroutine UNSAT

- Purpose:

  A user-programmed routine in which unsaturated flow functions are specified.

- Called by:

  BASIS2, BASIS3, BUDGET, INDAT2, NODAL, PUSWF

- Description:

  UNSAT is called by INDAT2 to calculate initial saturations at nodes, by BASIS2 (for 2D) or BASIS3 (for 3D) at each Gauss point in each element during numerical integration, by NODAL for each cell, by BUDGET for each cell, and by PUSWF at observation points. It allows the user to specify the functional dependence of relative permeability on saturation or pressure, and the dependence of saturation on pressure. UNSAT is divided into three sections. The first section requires the user to specify the saturation-pressure (or capillary pressure) function. The second section requires the user to specify the derivative or saturation with respect to pressure. The third section requires the user to specify the relative permeability dependence on saturation or capillary pressure. INDAT2 and PUSWF require only values of saturation, BASIS2 and BASIS3 require only values of saturation and relative permeability, and NODAL and BUDGET require values of saturation and its pressure derivative. These calculations are controlled in UNSAT by the parameter IUNSAT which INDAT2 and PUSWF set to a value of three, BASIS2 and BASIS3 set to a value of two, and NODAL and BUDGET set to one. For simulation of purely saturated flow, IUNSAT is set to zero by INDAT1, and UNSAT is never called. The user may program these functions in any convenient way, for example, through data statements, calls to other programs, logical structures, "read" or "write" statements, or other preferred methods. More information may be found in section 7.5, "User-Supplied Programming." Nodes and elements may be grouped by the user into REGIONS. For each REGION, a different set of unsaturated flow functions may be specified in UNSAT.

Function TIMETS

- Purpose:

  To return the time associated with a given time step.

- Called by:

  INDAT0

- Uses:

  LLDEF, SCHDEF

- Description:

  TIMETS uses the "TIME_STEPS" schedule, which controls time stepping, to determine the time that corresponds to the end of a given time step.

Subroutine ZERO

- Purpose:

  To fill a real array with a constant value.

- Called by:

  Main program, INDAT2, SUTRA

- Description:

  ZERO fills an entire array with a specified value.

5.10 Iterative Solver Package

SLAP (Sparse Linear Algebra Package; Seager (1989)) is a package of Fortran subroutines designed to implement a variety of iterative matrix solvers. The package included in SUTRA is a subset of SLAP version 2.0.2, which in turn is part of the SLATEC Common Mathematical Library (Vandevender and Haskell, 1982) version 4.1. The SLAP subroutines provided with SUTRA include only the double-precision versions of the CG, GMRES, and ORTHOMIN solvers and supporting subroutines. SLAP was not developed by the U.S. Geological Survey; disclaimers and programming credits appear in the code as comments. The code has undergone some minor modifications by the authors of SUTRA; such changes are commented in the code. General documentation for the double-precision SLAP routines appears in the listing of subroutine DLPDOC. In addition, each SLAP subroutine contains comments describing its purpose, usage, and arguments.

SUTRA accesses the SLAP package by calling the SLAP subroutines DSICCG (for CG), DSLUGM (for GMRES), and DSLUOM (for ORTHOMIN) from SUTRA subroutine SOLWRP. In preparation for calling the SLAP solvers, SUTRA stores the coefficient matrices in SLAP "column" format. If A is an array that contains the nonzeros of a coefficient matrix, then the pointer arrays IA and JA define the matrix structure by giving, respectively, the row index of each entry in A, and the index offsets into the IA and A arrays for the beginning of each column. For each column, the diagonal entry is listed first, followed by the remaining non-zero entries in order down the column. The coefficient matrices for the flow (p) and transport (U) problems have the same structure (the same pattern of nonzeros), so one pair of pointer arrays IA and JA is used for both matrices.

The CG (Conjugate Gradient) solver includes incomplete Cholesky preconditioning. The GMRES (Generalized Minimum Residual) and ORTHOMIN solvers both include incomplete LU preconditioning. Both GMRES and ORTHOMIN can handle nonsymmetric matrix problems, which arise when solving the transport equation with advection present (which is generally the case) or the flow equation with upstream weighting. Because the CG solver can handle only symmetric problems, its applicablility is, as a rule, limited to the flow equation in the absence of upstream weighting.

CG, GMRES, and ORTHOMIN belong to the Krylov-subspace-method family of iterative solvers. A discussion of the theory underlying these solvers and their particular implementation in the SLAP package is beyond the scope of this document. The methods are well established and are described in detail in texts that discuss sparse linear system solvers.

SUTRA SIMULATION EXAMPLES

Chapter 6: Simulation Examples

This chapter outlines a number of example simulations that serve to demonstrate some of the capabilities of SUTRA modeling. Some of the examples show results that are compared with analytical solutions or numerical solutions available in the literature. These results serve to verify the accuracy of SUTRA algorithms for a broad range of flow and transport problems. The other examples demonstrate physical processes that SUTRA may simulate in systems where no other solutions are available.

6.1 Pressure Solution for Radial Flow to a Well (Theis Analytical Solution)

Physical Setup:

A confined, infinite aquifer contains a fully penetrating withdrawal well. Fluid is pumped out at a rate of Q_{TOT}.

Objective:

To simulate transient drawdown in this system, which should match the Theis solution. In terms of variables used in SUTRA, the Theis solution (Lohman, 1979) is given by:

$$s^* = \frac{Q_{TOT}\mu}{4\pi\rho^2\,\Delta z\,k\,|\underline{g}|}\quad W(u) \tag{6.1a}$$

where s^* is the drawdown, W(u) is the well function of u, and

$$u = \frac{r^2\,\mu\,S_{op}}{4\,k\,t} \tag{6.1b}$$

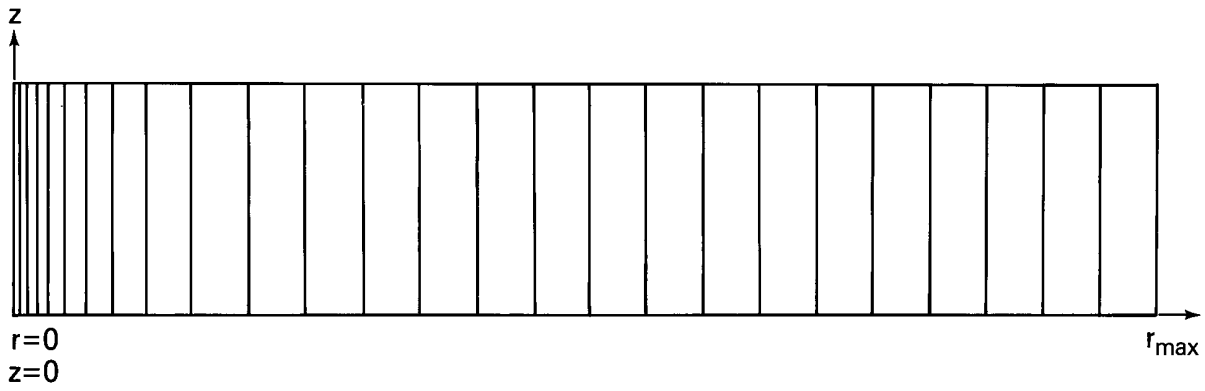


Figure 6.1. Radial finite-element mesh for Theis solution.

where r is the radial distance from the well to an observation point and t is the elapsed time since start of pumping.

Simulation Setup:

The mesh contains one row of elements with element width expanding by a constant factor, 1.2915, with increasing distance from the well; other mesh dimensions are Δr_{min}=2.5 [m], Δr_{max}=25.0 [m], r_{max}=500.0 [m], Δz=1. [m]. Mesh thickness at node i, is given by $B_i=2\pi r_i$, which provides a radial coordinate system. The number of nodes and elements in the mesh are: NN=54, NE=26, respectively. See Figure 6.1.

The initial time step is, Δt_o=l. [s], with time steps increasing by a factor of 1.5 on each subsequent step. One pressure solution is obtained per time step and solutions for concentration are ignored; the cycling parameters are NPCYC=1 and NUCYC=9999.

Parameters:

$S_{op} = 1.039 \times 10^{-6}$ [m · s^2/kg] $\varepsilon = 0.20$

$\alpha = 1.299 \times 10^{-6}$ [m · s^2/kg] $k = 2.0387 \times 10^{-10}$ [m^2]

$\beta = 4.4 \times 10^{-10}$ [m · s^2/kg] $\rho = 1000.$ [kg/m^3]

$|\underline{g}| = 9.81$ [m/s^2]

$Q_{TOT} = 0.6284$ [kg/s] (one half at each well node)

Boundary Conditions:

No flow occurs across any boundary except where hydrostatic pressure is specified at r_{max}. At the top outside corner of the mesh, r_{max}, pressure is held at zero. A sink is specified at r=0 to represent the well.

Initial Conditions:

Hydrostatic pressure with p=0.0 at the top of the aquifer is set initially.

Results:

SUTRA results are plotted for two locations in the mesh representing observation wells at r=15.2852 [m] and r=301.0867 [m]. Both locations should plot on the same Theis curve. The match of SUTRA results between 1 and 6000 minutes with the Theis analytical solution shown in Figure 6.2 is good.

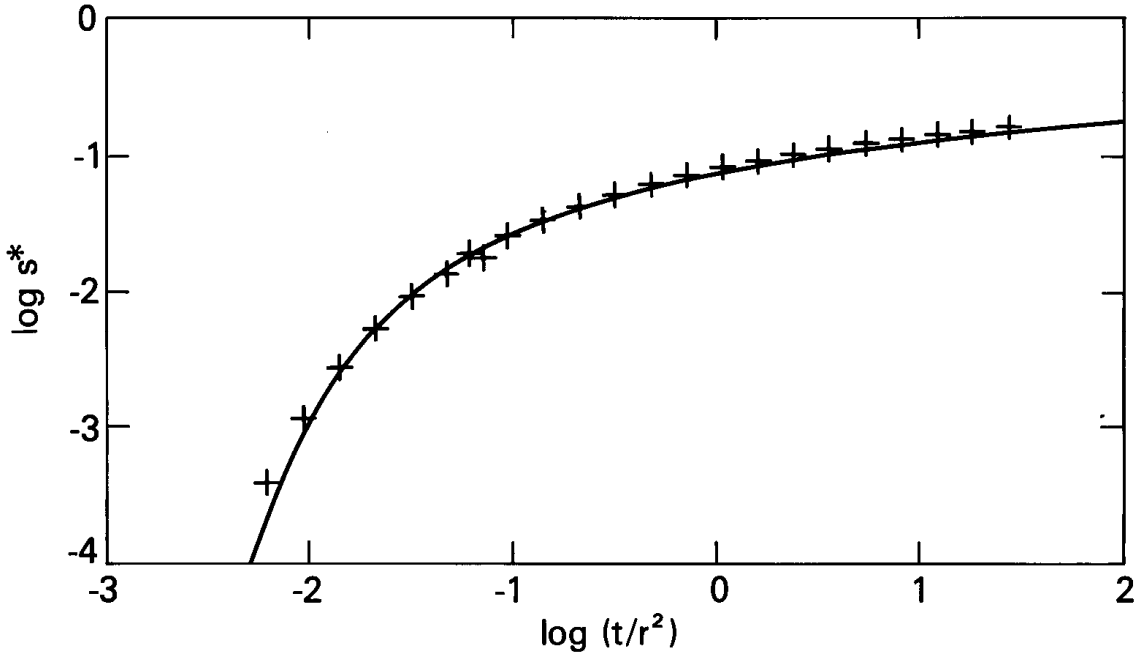


Figure 6.2. Match of Theis analytical solution (solid line) with SUTRA solution (+). Radial position, r, and drawdown, s^*, are in meters; time, t, is in minutes.

6.2 Radial Flow with Solute Transport (Analytical Solutions)

Physical Setup:

A confined infinite aquifer contains a fully penetrating injection well. Fluid is injected at a rate of Q_{TOT}, with a solute concentration of C^*. The aquifer initially contains fluid with solute concentration C_o. The fluid density does not vary with concentration.

Objective:

To use 2D SUTRA to simulate the transient propagation of the solute front as it moves radially away from the well. The concentrations should match the approximate analytical solutions of Hoopes and Harleman (1967) and Gelhar and Collins (1971).

The solution of Gelhar and Collins (1971) is:

$$\left(\frac{C - C_o}{C^* - C_o}\right) = \frac{1}{2}\,\mathrm{erfc}\left\{\frac{r^2 - (r^*)^2}{2\left[\left(\frac{4}{3}\alpha_L\right)(r^*)^3 + \left(\frac{D_m}{A}\right)(r^*)^4\right]^{\frac{1}{2}}}\right\} \tag{6.2}$$

where

$$r^* = (2At)^{\frac{1}{2}} \tag{6.3a}$$

$$A = \left(\frac{Q_{TOT}}{2\pi\varepsilon b\rho} \right) \tag{6.3b}$$

The Hoopes and Harleman (1967) solution is obtained by replacing r^* in the denominator of (6.2) with r.

Simulation Setup:

The mesh consists of one row of elements with element width expanding from Δr_{min}=2.5 [m] by a factor, 1.06, to r=395.0 [m], and then maintaining constant element width of Δr=24.2 [m] to r_{max}=1000.0 [m]. Element height, b, is 10. [m]. Mesh thickness is set for radial coordinates, $B_i=2\pi r_i$, with the number of nodes and elements given by NN=132 and NE=65, respectively. See Figure 6.3.

The time step is constant at Δt=4021.0 [s] and outputs are obtained for times steps numbered 225, 450, 900, 1800. One pressure solution is carried out to obtain a steady state, (ISSFLO=1), and one concentration solution is done per time step (NUCYC=1).

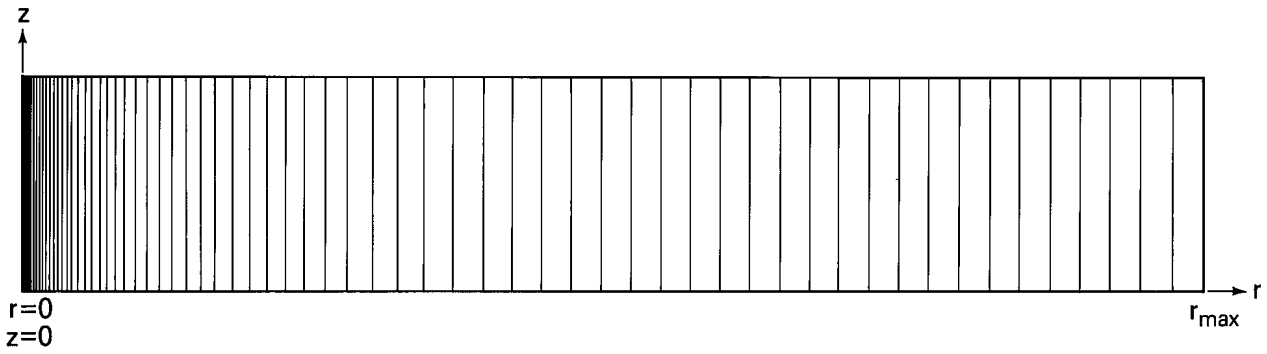


Figure 6.3. Radial finite-element mesh for constant-density solute- and energy-transport examples.

Parameters:

S_{op} = 0.0 [m · s^2/kg]

ρ = 1000. [kg/m^3]

k = 1.02x10^{-11} [m^2]

D_m = 1.x10^{-10} [m^2/s]

ε = 0.2

α_L = 10.0 [m]

μ = 1.0x10^{-3} [kg/m · s]

α_T = 0.0 [m]

$|\underline{g}|$ = 9.8 [m/s^2]

C^* = 1.0

Q_{TOT} = 62.5 [kg/s] (one half at each well node)

Boundary Conditions:

No flow occurs across any boundary except where hydrostatic pressure is specified at r_{max}. At the top outside corner of the mesh, r_{max}, pressure is held at zero. A source is specified at r = 0.0 to represent the injection well.

Initial Conditions:

Initially hydrostatic pressure is set with p = 0.0 at the aquifer top. Initial concentration, C_o, is set to zero.

Results:

SUTRA results after 225, 450, 900 and 1800 time steps are compared with the approximate analytical solutions of Gelhar and Collins (1971) and Hoopes and Harleman (1967) in Figure 6.4. The analytical solutions are approximate, and they bound the SUTRA solution at the top and bottom of the solute front. All solutions compare well with each other, and the SUTRA solution may be considered more accurate than either approximate analytic solution because it makes no simplifying assumptions to solve the governing equations and is based on a very fine spatial and temporal discretization of the governing equation.

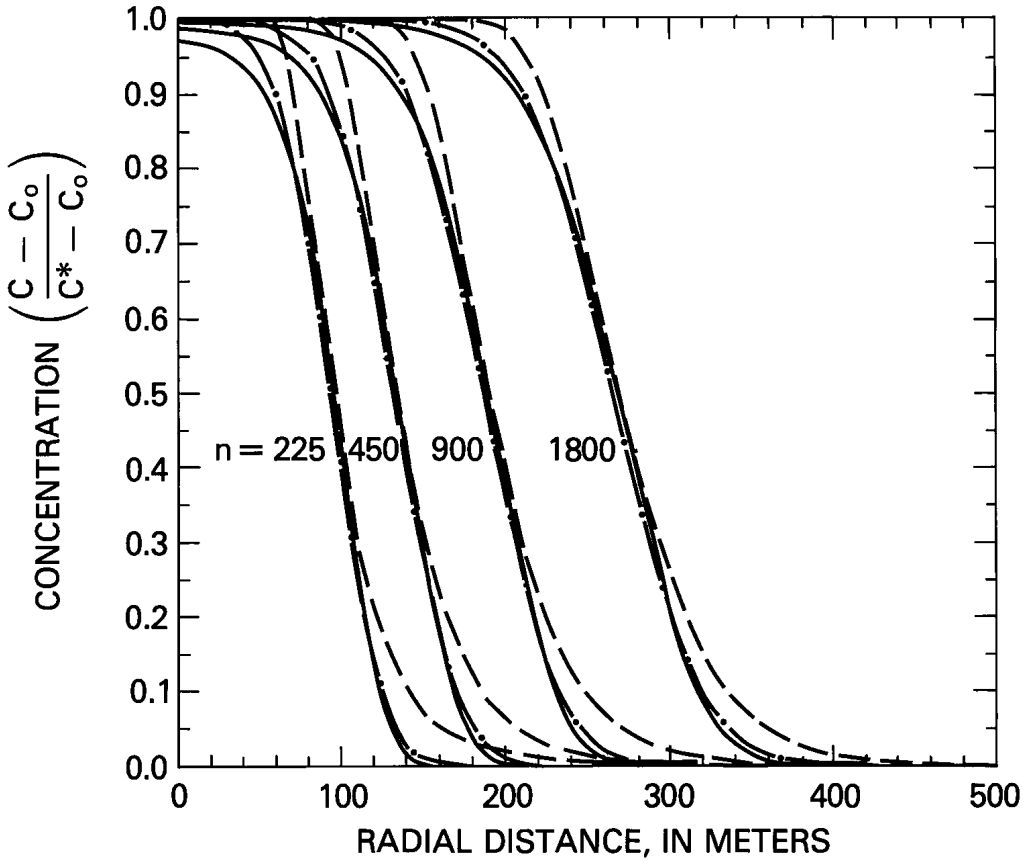


Figure 6.4. Match of analytical solutions for radial solute transport of Hoopes and Harleman (1967) (dashed), Gelhar and Collins (1971), (solid), and SUTRA solution (dash-dot). Number of elapsed time steps is n.

6.3 Radial Flow with Energy Transport (Analytical Solution)

Physical Setup:

A confined aquifer contains a fully penetrating injection well. Fluid is injected at a rate of Q_{TOT}, with a temperature of T^*, into the aquifer initially at a temperature of T_o. For this problem, density, ρ, and viscosity, μ, are kept approximately constant by injecting fluid that only slightly differs in temperature from the ambient fluid; i.e., (T^*-T_o) is small.

Objective:

To use 2D SUTRA to simulate the transient propagation of the temperature front as it radially moves away from the well. The solution should match an approximate analytical solution of Gelhar and Collins (1971) modified for energy transport. The Gelhar and Collins (1971) solution, as modified for energy transport is:

$$\left(\frac{T-T_o}{T^*-T_o}\right)=\frac{1}{2}\,\mathrm{erfc}\left\{\frac{r^2-(r^*)^2}{2\left[\left(\frac{4}{3}\alpha_L\right)(r^*)^3+\left(\frac{\lambda_{TOT}}{A_T}\right)(r^*)^4\right]^{\frac{1}{2}}}\right\} \tag{6.4}$$

$$A=\frac{Q_{TOT}}{2\pi\varepsilon b\rho} \tag{6.5}$$

$$A_T=\left(\frac{\varepsilon\rho c_w}{c_{TOT}}\right)A \tag{6.6}$$

$$c_{TOT}=\varepsilon\rho c_w+(1-\varepsilon)\rho_s c_s \tag{6.7}$$

$$\lambda_{TOT}=\varepsilon\lambda_w+(1-\varepsilon)\lambda_s \tag{6.8}$$

$$r^*=(2A_T t)^{\frac{1}{2}} \tag{6.9}$$

The energy solution above may be obtained from the solute solution by retarding the velocity of transport to represent movement of an isotherm rather than a parcel of solute. This is done by accounting for energy storage in the solid grains of the aquifer material in the storage term of the analytical solution.

Simulation Setup:

The mesh used for this example is the same as for the radial solute-transport example (section 6.2). Time steps and frequency of SUTRA outputs are the same as for the radial solute-transport example. Further, cycling of the SUTRA solution is the same as for the radial solute-transport example.

Parameters:

$c_w = 4182.$ [J/kg · °C]

$c_s = 840.$ [J/kg · °C]

$\lambda_w = 0.6$ [J/s · m · °C]

$\rho = 1000.$ [kg/m^3]

$\lambda_s = 3.5$ [J/s · m · °C]

$\rho_s = 2650.$ [kg/m^3]

$\frac{\partial \rho}{\partial T} = 0.0$

$\mu = \mu$ (T) (relation (2.5))

$Q_{TOT} = 312.5$ [kg/s] (one half at each well node)

$T^* = 1.0$ [°C]

$S_{op} = 0.$ [m · s^2/kg]

$k = 1.02 \times 10^{-11}$ [m^2]

$\varepsilon = 0.2$

$|\underline{g}| = 9.8$ [m/s^2]

$\alpha_L = 10.$ [m]

$\alpha_T = 0.$ [m]

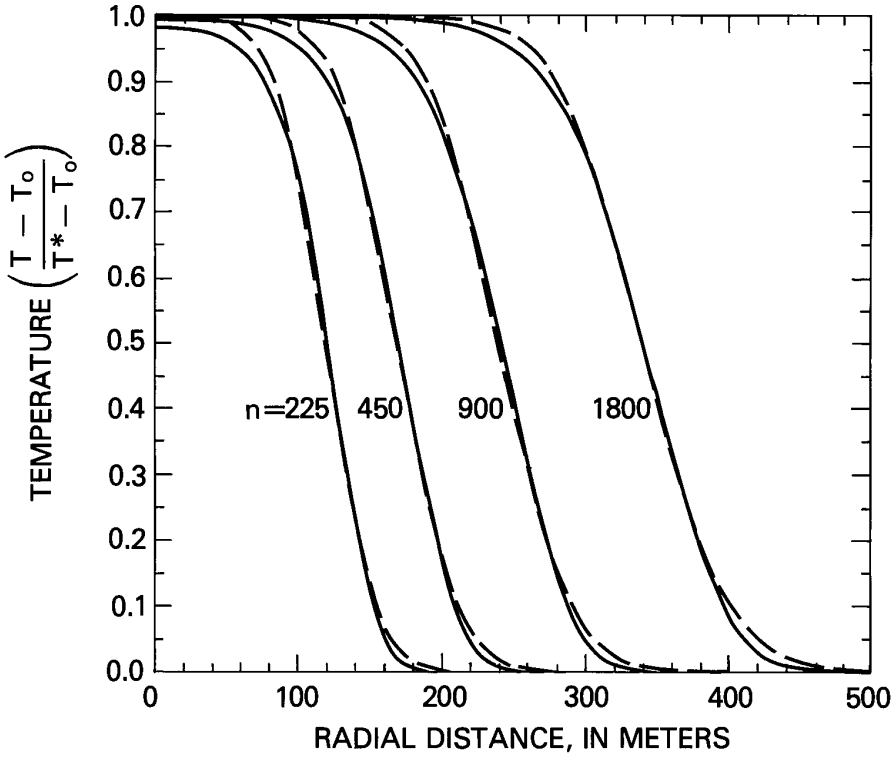


Figure 6.5. Match of analytical solution for radial energy transport modified from Gelhar and Collins (1971) (solid line) with SUTRA solution (dashed line). Number of elapsed time steps is n.

Boundary Conditions:

No flow occurs across any boundary except where hydrostatic pressure is specified at r_{max}. At the top outside corner of the mesh, pressure is held at zero. A source is specified at r = 0.0 to represent the injection well. Further, the system is thermally insulated along the top and bottom of the mesh.

Initial Conditions:

Initially, hydrostatic pressure is set with p = 0.0 at the top of the aquifer. The initial temperature is $T_o = 0.0$ [°C].

Results:

SUTRA results after 225, 450, 900 and 1800 time steps are compared with the approximate (modified) analytical solution of Gelhar and Collins (1971) in Figure 6.5. The analytical solution has the same relation to the SUTRA solution as it does in Figure 6.4 for solute transport. Thus, the match is good, and again the SUTRA result may be more accurate than the approximate analytic result.

6.4 Areal Constant-Density Solute Transport (Example at Rocky Mountain Arsenal)

Physical Setup:

This example involves a simple representation of ground-water flow and solute transport at the Rocky Mountain Arsenal, Denver, Colorado, which is based on the detailed model of the system by Konikow (1977). The simplified representation consists of an areal model of a rectangular alluvial aquifer with a constant transmissivity and two impermeable bedrock outcrops which influence groundwater flow. (See Figure 6.6.)

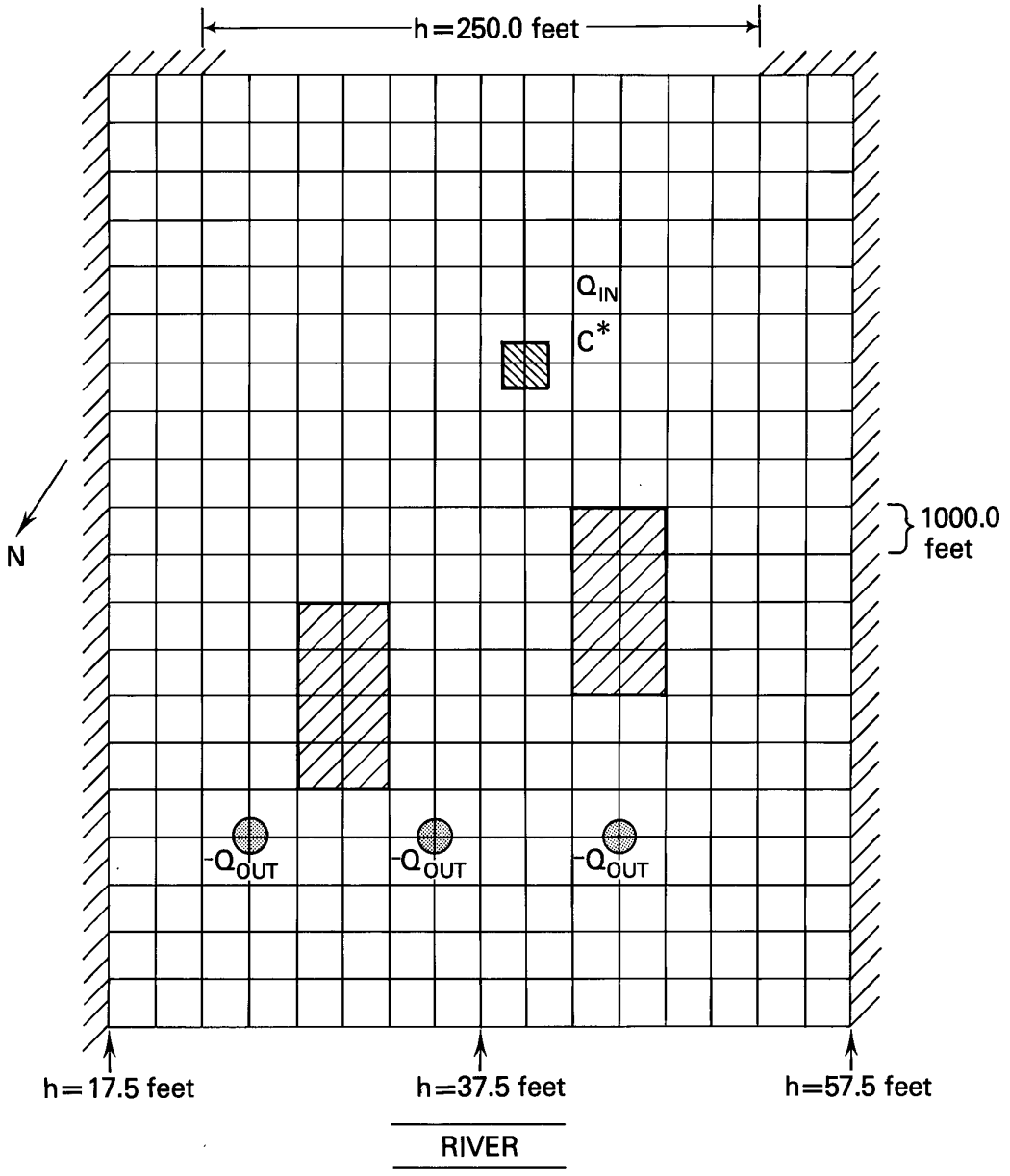


Figure 6.6. Idealized representation for example at Rocky Mountain Arsenal, and finite-element mesh. Upper shaded square is the pond, shaded rectangles are impermeable zones, and three circles are wells.

Regional flow is generally from the southeast to the northwest where some discharge occurs at the South Platte River. This is idealized as flow originating in a constant head region at the top of the rectangle in Figure 6.6, and discharging to the river at the bottom of the rectangle, which also acts as a specified head region. Three wells pump from the aquifer (at a rate of Q_{OUT} each), and contamination enters the system through a leaking waste isolation pond (at a rate of Q_{IN}, with concentration, C^*). The natural background concentration of the contaminant is C_o.

Objectives:

1) To demonstrate the applicability of SUTRA to simulate an areal (2D) constant-density solute transport problem. 2) To convert SUTRA input data values so the pressure results represent heads, and the concentration results are in [ppm]. 3) To simulate steady-state flow and hypothetical steady-state distributions of the contaminating solute, both as a conservative solute, and as a solute that undergoes first order decay, assuming that the contamination source in the idealized system is at a steady state. 4) To test the ability of SUTRA to give the same results in 3D when the 2D problem is "extruded" into the third dimension, i.e., when the problem is formulated so that the solution varies in any two of the three coordinate directions, but not in the third direction.

Simulation Setup:

The rectangular mesh consists of 16 by 20 elements each of dimension 1000.0 [ft] by 1000.0 [ft], as shown in Figure 6.6. (NN=357, NE=320). Mesh thickness, B, is the actual aquifer thickness, assumed constant for the idealized model.

One steady-state pressure solution is obtained (ISSFLO=1), and one concentration solution is obtained. The concentration solution is obtained after a single time step of 1000. years, which, for all practical purposes, brings the concentration distribution to a steady state.

The leaky pond is simulated as an injection of fluid (Q_{IN}, C^*) at a single node. Where the impermeable bedrock outcrop occurs, elements are assigned a conductivity value one millionth of the aquifer values. A single value of constant head is specified along a portion of the top boundary, and a series of head values is specified along the bottom (river) boundary to represent changing elevation of the river.

To obtain results in terms of hydraulic head and [ppm], the following must be specified: ρ=1.0, $\partial\rho/\partial c$ =0.0, $|\underline{g}|$=0.0, μ=1.0. Hydraulic conductivities are entered in the permeability input data set. Head values in [ft] are entered in data sets for pressure. Concentrations in [ppm] are entered in data sets for mass fraction concentration. Sources and sinks are entered in units of volume per time.

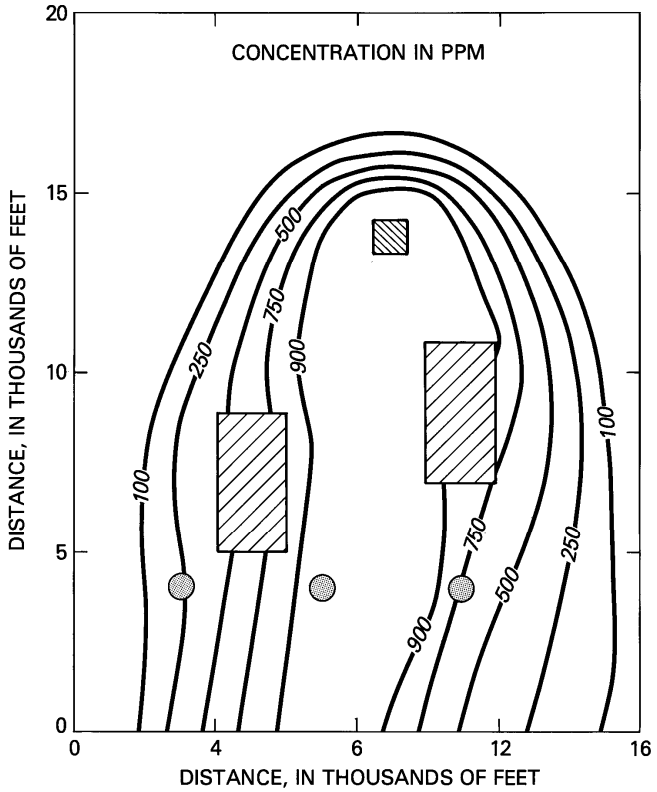


Figure 6.7. Nearly steady-state conservative solute plume as simulated for the Rocky Mountain Arsenal example by SUTRA.

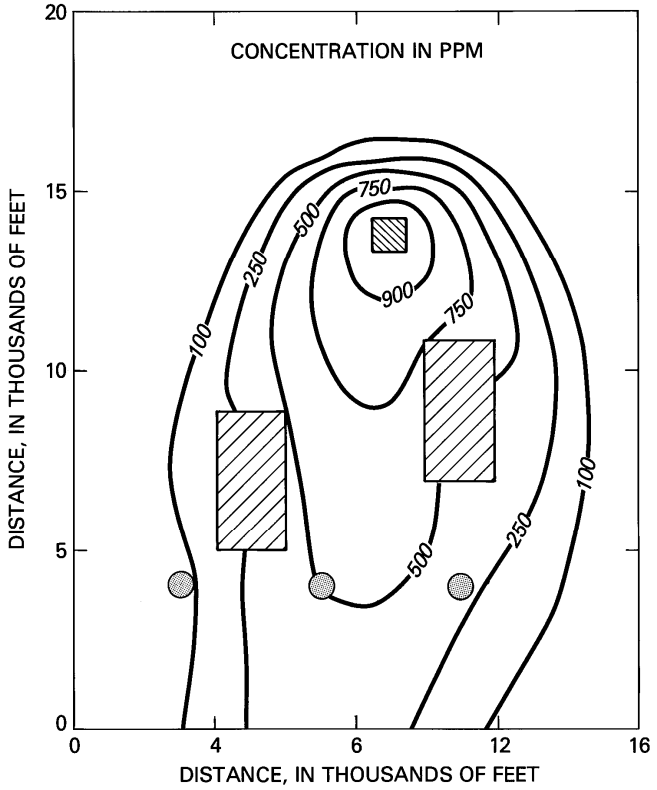


Figure 6.8. Nearly steady-state solute plume (with solute half life ~ 20. years) as simulated for the Rocky Mountain Arsenal example by SUTRA.

Parameters:

α_L = 500. [ft]

α_T = 100. [ft]

ε = 0.2

K = 2.5×10^{-4} [ft/s]
(hydraulic conductivity)

B = 40. ft

Q_{IN} = 1.0 [ft^3/s]

C^* = 1000. [ppm]

C_o = 10. [ppm]

Q_{OUT} = 0.2 [ft^3/s]
(at each of three wells)

Boundary Conditions:

No flow occurs across any boundary except where constant head is specified at 250.0 [ft] at the top of the mesh and where constant head is specified as changing linearly between 17.5 [ft] at the bottom left corner, and 57.5 [ft] at the bottom right corner of the mesh. Inflow at the top of the mesh is at background concentration, C_o=10.0 [ppm]. A source is specified at the leaky pond node, and a sink is specified at each well node.

Initial Conditions:

Initial pressures are arbitrary for steady-state simulation of pressure. Initial concentration is C_o=10.0 [ppm].

Results:

A nearly steady-state solute plume for a conservative solute is obtained after a 1000 year time step shown in Figure 6.7. For a solute which undergoes first order decay with decay coefficient, $\gamma = 1.1 \times 10^{-9}$ [s^{-1}] (approximately a 20 year half life), the nearly steady plume is shown in Figure 6.8. Just upstream of the plume envelope is a region in which concentration dips slightly below background levels. This is due to a numerical problem of insufficient spatial discretization in a region where the concentration must change sharply from fresh upstream values to contaminated plume values. Lower dispersivity values would exacerbate the problem in the upstream region, but minor upstream oscillations do not affect concentration values within the plume.

Results of three simulations using SUTRA in 3D with the areal problem formulated in the (x,y), (x,z) and (y,z) coordinates, respectively, and extruded into the third dimension are each equivalent to those obtained using SUTRA in 2D .

6.5 Density-Dependent Flow and Solute Transport (Henry (1964) Solution for Seawater Intrusion)

Physical Setup:

This problem involves seawater intrusion into a confined aquifer studied in cross section under steady conditions. Freshwater recharge inland flows over saltwater in the section and discharges at a vertical sea boundary.

The intrusion problem is nonlinear and may be solved by approaching the steady state gradually with a series of time steps. Initially there is no saltwater in the aquifer, and at time zero, saltwater begins to intrude the freshwater system by moving under the freshwater from the sea boundary. The intrusion is caused by the greater density of the saltwater.

Dimensions of the problem are selected to make for simple comparison with the steady-state dimensionless solution of Henry (1964), and with a number of other published simulation models. A total simulation time of t=100.0 [min], is selected, which is sufficient time for the problem to essentially reach steady state at the scale simulated.

Objective:

1) To compare SUTRA results with the solution of Henry (1964), and with other published simulation results. 2) To test the ability of SUTRA to give the same results in 3D when the 2D problem is "extruded" into the third dimension, i.e., when the problem is formulated so that the solution varies in any two of the three coordinate directions, but not in the third direction.

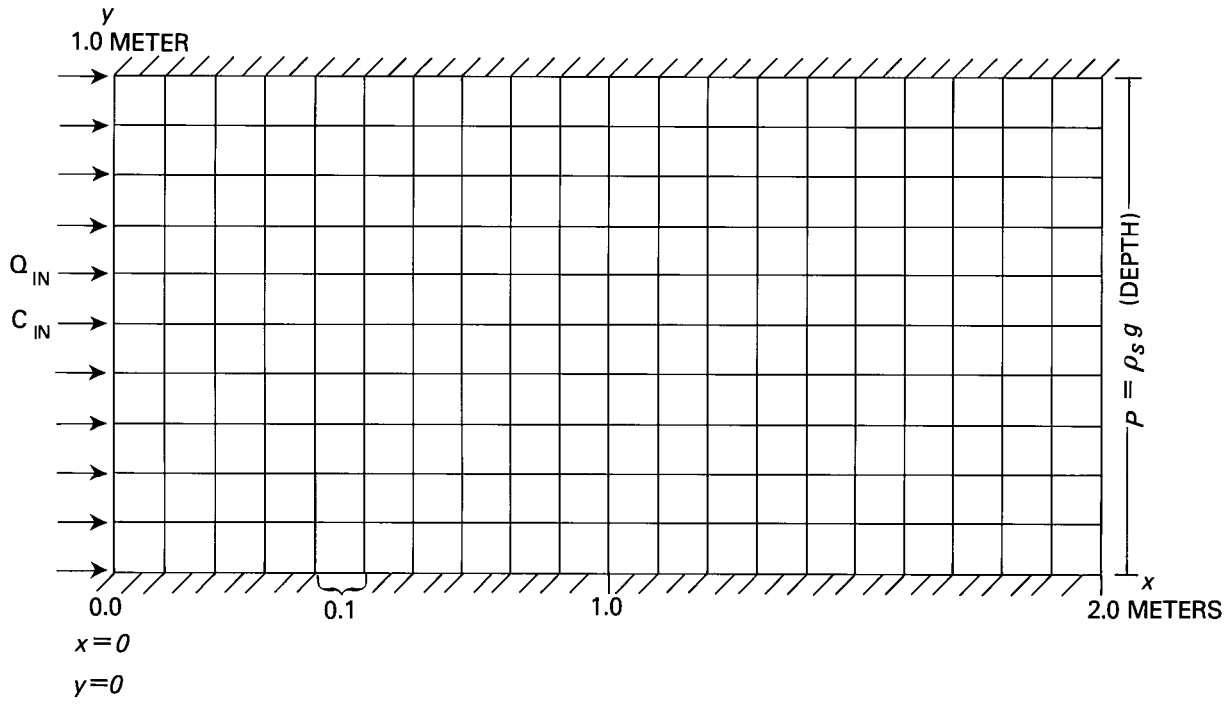


Figure 6.9. Boundary conditions and finite-element mesh for Henry (1964) solution.

Simulation Setup:

The mesh consists of twenty by ten elements, each of size 0.1 [m] by 0.1 [m], (NN=231, NE=200). Mesh thickness, B, is 1.0 [m]. See Figure 6.9. Time steps are of length 1.0 [min], and 100 time steps are taken in the simulation. Both pressure and concentration are solved for on each time step (NUCYC=NPCYC=1).

A source of freshwater is implemented by employing source nodes at the left vertical boundary, which inject freshwater at a rate of Q_{IN}, and concentration of C_{IN}. The right vertical boundary is held at hydrostatic pressure of seawater through the use of specified pressure nodes. Any water that enters the section through these nodes has the concentration of seawater ($C_{BC} = C_{sea}$).

Parameters:

$\varepsilon = 0.35$

$k = 1.020408 \times 10^{-9}$ [m^2]
(based on $K = 1.0 \times 10^{-2}$ [m/s])

$C_{sea} = 0.0357 \left[\frac{\text{kg(dissolved solids)}}{\text{kg(seawater)}} \right]$

$B = 1.0$ [m]

$\rho_{sea} = 1024.99$ [kg/m^3]

$\alpha_L = \alpha_T = 0.0$ [m]

$\frac{\partial \rho}{\partial C} = 700. \left[\frac{\text{kg(seawater)}^2}{\text{kg(dissolved solids) m}^3} \right]$

$|\underline{g}| = 9.8$ [m/s^2]

$\rho_o = 1000.$ [kg/m^2]

$D_m = \begin{cases} 6.6 \times 10^{-6} \text{ [m}^2\text{/s]} & \text{two} \\ 18.8571 \times 10^{-6} \text{ [m}^2\text{/s]} & \text{cases} \end{cases}$

$Q_{IN} = 6.6 \times 10^{-2}$ [kg/s]
(divided among 11 nodes at left boundary)

$C_{IN} = 0.0 \left[\frac{\text{kg(dissolved solids)}}{\text{kg(water)}} \right]$

Boundary Conditions:

No flow occurs across the top and bottom boundaries. A freshwater source is set along the left vertical boundary. Specified pressure is set at hydrostatic seawater pressure with (ρ_{sea}=1024.99 [kg/m^3]) along the right vertical boundary. Any inflowing fluid at this boundary has the concentration, C_{sea}=0.0357 [kg(dissolved solids)/kg(seawater)], of seawater.

Initial Conditions:

Natural steady pressures are set everywhere in the aquifer based on the freshwater inflow, zero concentration everywhere, and the specified pressures at the sea boundary. These initial conditions are obtained through a preliminary simulation that calculates steady pressures under these conditions.

Results:

Henry's solution assumes that dispersion is represented by a constant large coefficient of diffusion, rather than by velocity-dependent dispersivity. Two different values of this diffusivity have apparently been used in the literature by those testing simulators against Henry's solution. The total dispersion coefficient of Henry (1964), D, is equivalent to the product of porosity and molecular diffusivity in SUTRA, $D = \varepsilon D_m$.

Henry's results are given for his nondimensional parameters $\xi = 2.0$, $b = 0.1$, $a \cong 0.264$ (page C80—Figure 34 in Henry (1964)). To match the Henry parameters using simulation parameters as listed above, values of $D = 6.6 \times 10^{-6}$ [m^2/s] and $D_m = 18.8571 \times 10^{-6}$ [m^2/s] are required. Some authors, however, have apparently used a value equivalent to $D_m = 6.6 \times 10^{-6}$ [m^2/s] and $D = 2.31 \times 10^{-6}$ [m^2/s], which differs from the Henry parameters by a factor equal to the porosity.

In the previous model solutions compared here, only Huyakorn and Taylor (1976) have employed the higher value of diffusivity, which should match Henry's solution. A comparison of SUTRA results using the higher diffusivity value with those of Huyakorn and Taylor (1976) along the bottom of the section at t=100. [min] is shown in Figure 6.10. Huyakorn and Taylor's results are for a number of simulation models based on significantly different numerical methods. SUTRA results are also shown for the lower diffusivity value. The results of simulations using the higher diffusivity value compare favorably. Results using the higher value have also been obtained with the INTERA (1979) finite-difference code at t=100. [min] (with centered-in-space and centered-in-time approximations). These are compared with SUTRA and the Henry solution for the 0.5 isochlor in Figure 6.11. The models match well but do not compare favorably with the analytical solution, which is approximate and may not be as accurate as the numerical solutions.

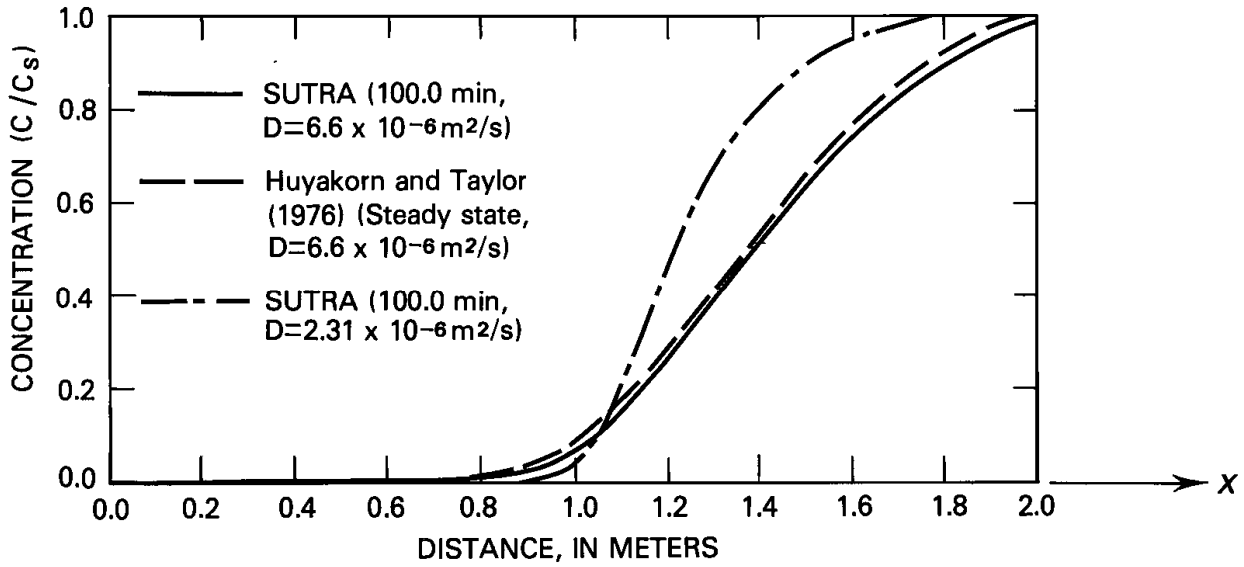


Figure 6.10. Match of isochlors along bottom of aquifer for numerical results of Huyakorn and Taylor (1976) and SUTRA.

For the lower value of diffusivity, $D_m = 6.6x10^{-6}$ [m^2/s], (which should not compare with the Henry result), the SUTRA solution at t=100. [min] is compared in Figure 6.12 with that of Pinder and Cooper (1970) (method of characteristics), Segol et. al. (1975) (finite elements), Desai and Contractor (1977) (finite elements–coarse mesh), and Frind (1982) (finite elements). The match of the numerical 0.5-isochlor solutions is remarkably good; however, it should be noted that none of these match the analytical solution.

Results obtained using SUTRA in 3D with the problem formulated in the (x,y), (x,z) and (y,z) coordinates, respectively, and extruded in the third dimension are each equivalent to those obtained using SUTRA in 2D .

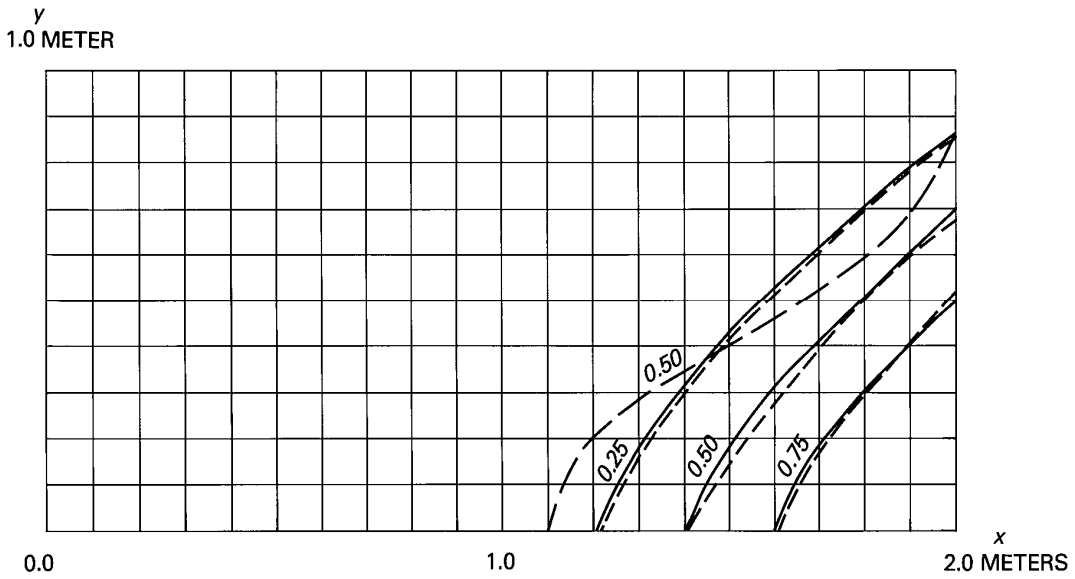


Figure 6.11. Match of isochlor contours for Henry analytical solution (for 0.50 isochlor) (long dashes), INTERA code solution (short dashes), SUTRA solution (solid line).

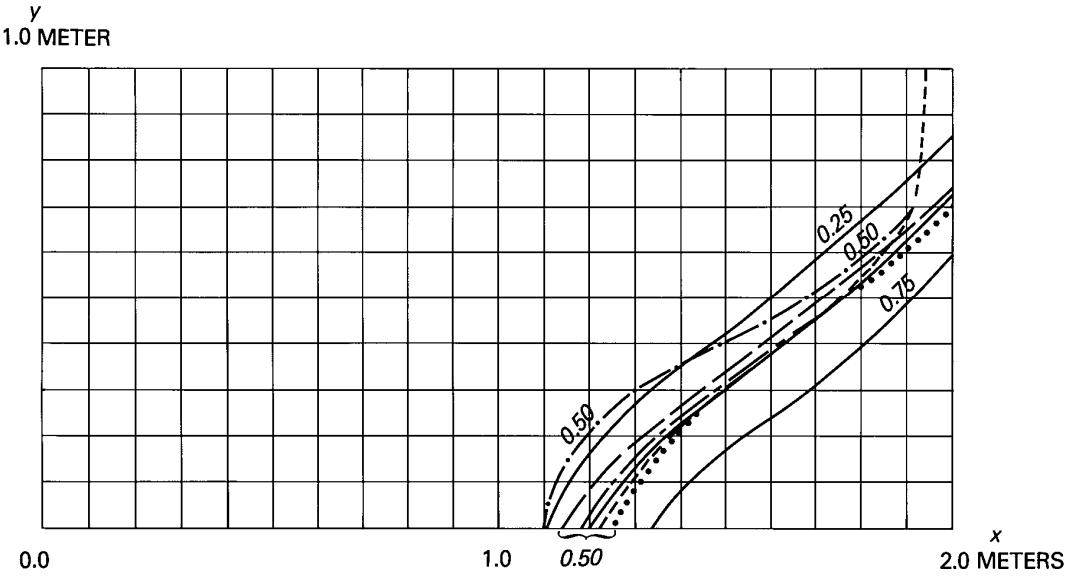


Figure 6.12. Match of 0.50 isochlor contours for Henry problem with simulated results for $D_m = 6.6 \times 10^{-9}$ [m^2/s] of Pinder and Cooper (1970), (short dashes), Segol, et. al. (1975) (dotted line), Frind (1982) (long and short dashes), Desai and Contractor (1977) (long dashes). SUTRA results at isochlors (0.25, 0.50, 0.75) (solid line). Henry (1964) solution for $D_m = 18.8571 \times 10^{-9}$ [m^2/s], (0.50 isochlor, dash-dot).

6.6 Density-Dependent Radial Flow and Energy Transport (Aquifer Thermal Energy Storage Example)

Physical Setup:

This is an example of aquifer thermal energy storage. Hot water is injected into an aquifer for storage and later withdrawn and used as an energy source. The fully penetrating injection wells are emplaced in a well field in a hexagonal packing pattern. The wells are at the vertices of contiguous equilateral triangles with sides of 500.0 [m]. This gives approximately radial symmetry to physical processes surrounding an interior well.

Objective:

To simulate the initial injection-withdrawal cycle at an interior well consisting of 90 days of injection (at Q_{IN}) of 60 [°C] water into the aquifer initially at 20 [°C], and 90 days of withdrawal (at $-Q_{IN}$) producing the stored water. Degradation of recovered fluid temperature should occur due to thermal conduction, dispersion, and tipping of the thermal front. The front should tip as less dense, less viscous hot water rises over colder, denser, and more viscous formation water.

Simulation Setup:

The mesh is 30.0 [m] high with a vertical spacing between nodes of 3.0 [m]. The first column of elements has width $\Delta r_{min} = 1.0$ [m], and element width increases with each column by a factor, 1.1593, to a final column of width, $\Delta r_{max} = 35.0$ [m]. The outside boundary of the mesh is at $r_{max} = 246.0$ [m]. See Figure 6.13. Mesh thickness, B, at any node i, is $B_i = 2\pi r_i$, giving cylindrical symmetry. The number of nodes and elements in the mesh is given by NN=286 and NE=250, respectively.

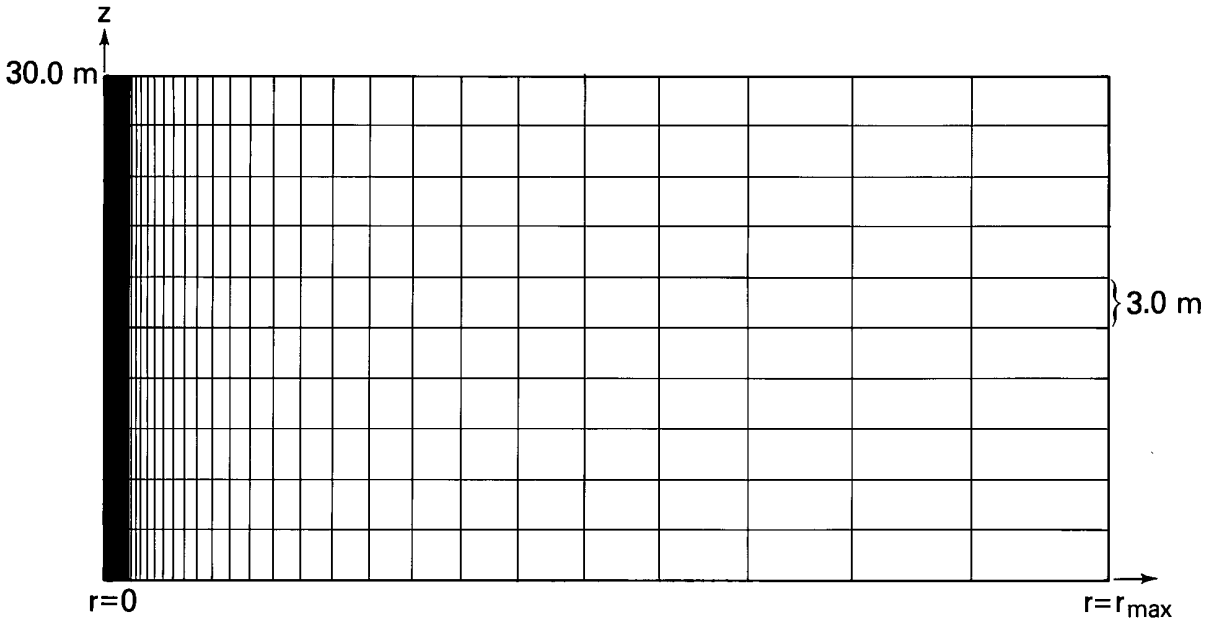


Figure 6.13. Radial two-dimensional finite-element mesh for aquifer thermal energy storage example.

The time step is constant at $\Delta t = 3.0$ [days]. One pressure solution and one temperature solution is obtained at each time step (NPCYC=NUCYC=1). The storage coefficient is assumed negligible, resulting in a steady flow field at any time step. Subroutine BCTIME is programmed to control the well rate, which changes after 90 days from fluid injection to fluid withdrawal. This may also be accomplished by stopping the simulation after 90 days and restarting with fluid withdrawal, using the restart (".rst") file as the initial conditions (".ics") file.

A time-dependent fluid source is specified at the left vertical boundary (center axis), which injects 60. [°C] water for 90 days and then withdraws ambient water for 90 days. The right vertical boundary is held at hydrostatic pressure for water at 20. [°C]. Any inflow at this boundary has a temperature of 20. [°C]. Thermally insulated and impermeable conditions are held at the top and bottom of the mesh.

Parameters:

$c_w = 4182.$ [J/kg · °C]

$c_s = 840.$ [J/kg · °C]

$\lambda_w = 0.6$ [J/s · m · °C]

$\lambda_s = 3.5$ [J/s · m · °C]

$T_o = 20.$ [°C]

$\frac{\partial \rho}{\partial T} = -0.375$ [kg/m^3 · °C]

$T^* = 60.$ [°C]

$Q_{TOT} = 200.$ [kg/s]
(distributed along well)

$S_{op} = 0$ [m · s^2/kg]

$k = 1.02 \times 10^{-10}$ [m^2]

$\varepsilon = 0.35$

$\rho_o = 1000.$ [kg/m^3]

$\rho_s = 2650.$ [kg/m^3]

$\mu = \mu(T)$ (relation (2.5))

$|\underline{g}| = 9.81$ [m/s^2]

$\alpha_L = 4.0$ [m]

$\alpha_T = 1.0$ [m]

Boundary Conditions:

Conditions of no flow and thermal insulation are held at all boundaries except where hydrostatic pressure at T = 20.0 [°C] is specified at r_{max}. At the top outside corner of the mesh the pressure is held at zero. A time-dependent source is specified at r = 0.0 to represent the injection-withdrawal well.

Initial Conditions:

Hydrostatic pressure is specified initially, with p = 0.0 at the top of the aquifer. The initial temperature is set to $T_o = 20.0$ [°C].

Results:

SUTRA results during injection after 30 days and 90 days are shown in Figure 6.14 and Figure 6.15. Simulated results during withdrawal are shown in Figure 6.16, Figure 6.17, and Figure 6.18 after 30 days, 60 days, and 90 days of withdrawal. The thermal transition zone (between hot and cold water) widens throughout the injection-production cycle, due to both dispersion and heat conduction. The top of the transition zone tips away from the well during the entire cycle, due to the buoyancy of the hotter water. These two effects combine to cause cooler water to reach the bottom of the withdrawal well much earlier than if no density differences or dispersion existed. In addition, although the same quantity of water has been removed as injected, energy is lost to the aquifer during the cycle as seen at the end of simulation.

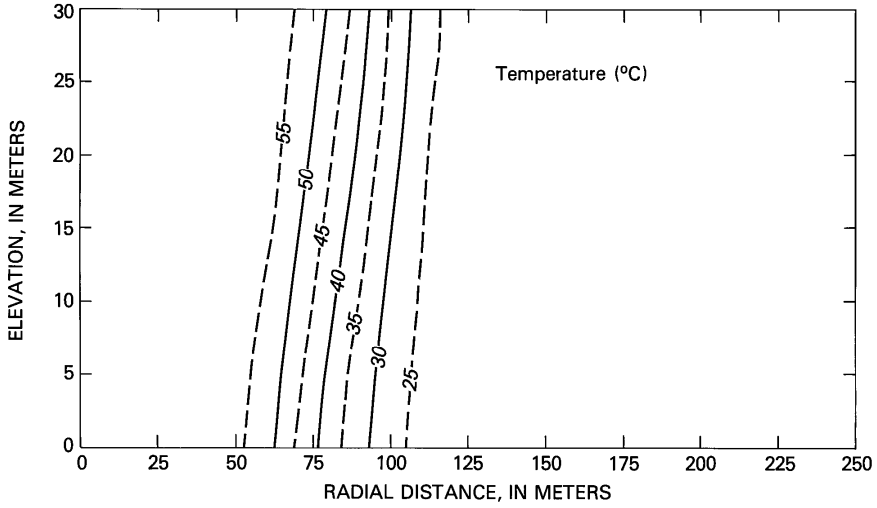


Figure 6.14. SUTRA results after 30 days of hot water injection.

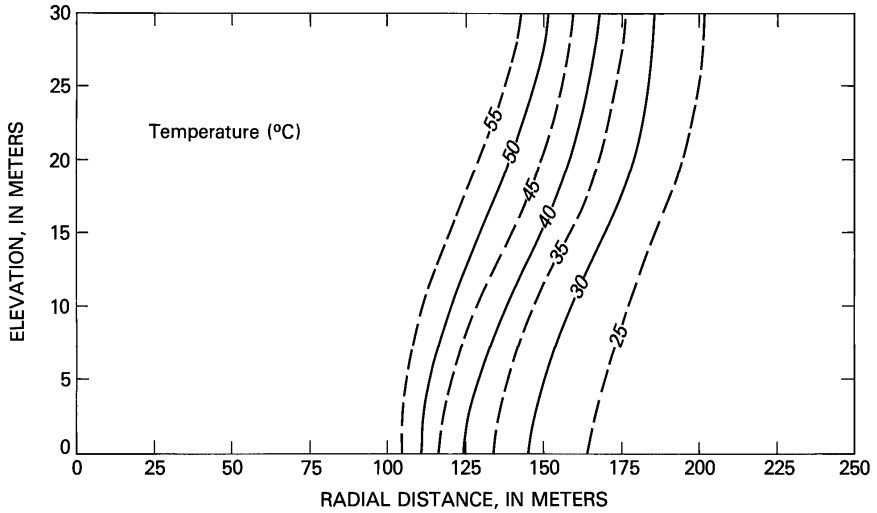


Figure 6.15. SUTRA results after 90 days of hot water injection.

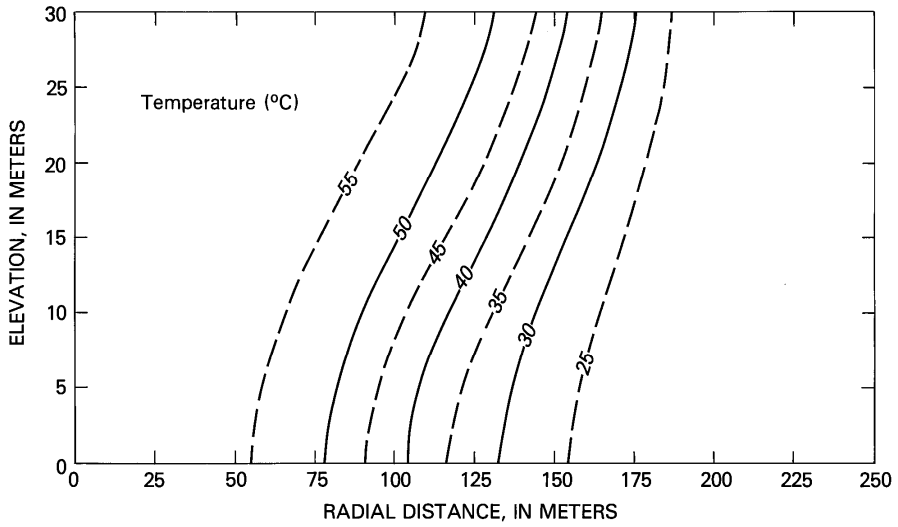


Figure 6.16. SUTRA results after 30 days of pumping (120 days total elapsed time).

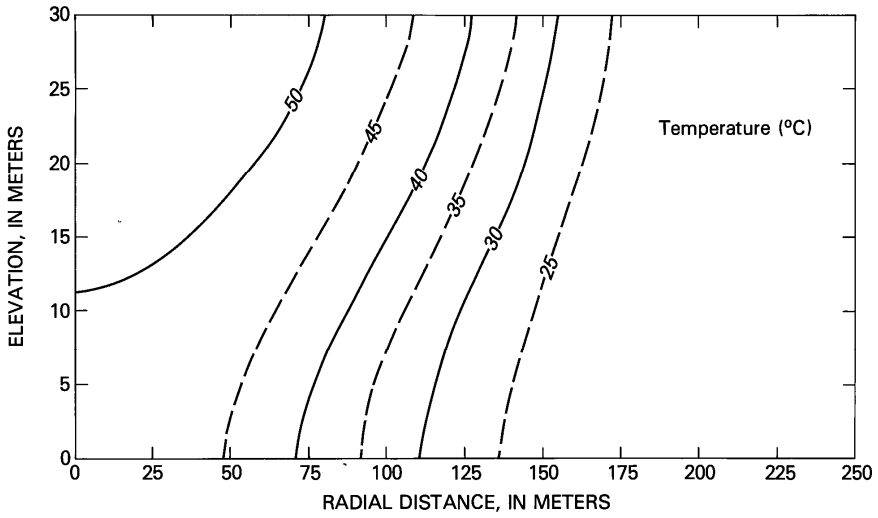


Figure 6.17. SUTRA results after 60 days of pumping (150 days total elapsed time).

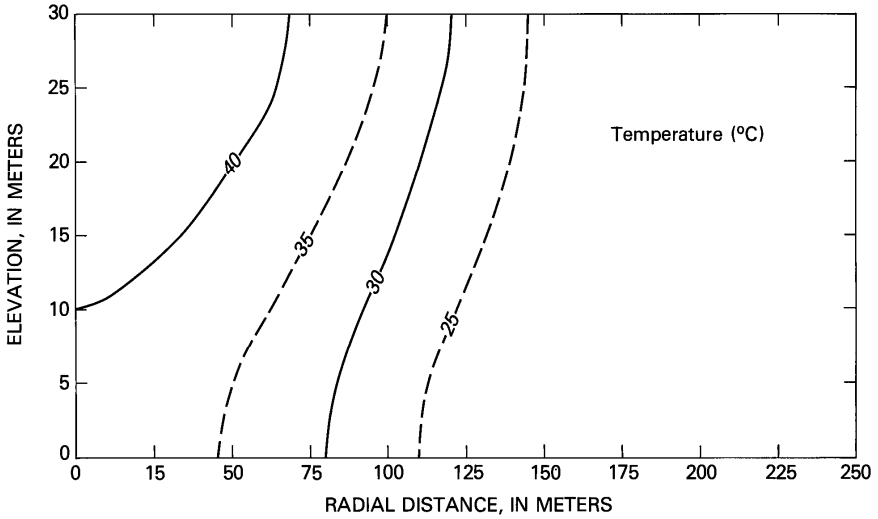


Figure 6.18. SUTRA results after 90 days of pumping (180 days total elapsed time).

6.7 Constant-Density Unsaturated Flow and Solute Transport (Example from Warrick, Biggar and Nielsen (1971))

Physical Setup:

Water containing solute infiltrates an initially unsaturated solute-free soil for about two hours. Solute-free water continues to infiltrate the soil after the initial two hours. The moisture front and a slug of solute move downward through the soil column under conservative, nonreactive, constant-density transport conditions, as described in a field experiment by Warrick, Biggar, and Nielsen (1971).

Objective:

To simulate the transient propagation of the moisture front and solute slug as they move downwards through the soil column, under a simulation setup equivalent to that used by Van Genuchten (1982) to represent the field experiment. The solutions should match the best fine grid, fine time step simulation results of Van Genuchten (1982), which were obtained with a number of different finite difference and finite element numerical methodologies.

Simulation Setup:

The mesh consists of a single 2.0 [m] long and 0.01 [m] wide vertical column of 100 elements oriented in the direction of gravity. The number of nodes and elements is NN = 202 and NE = 100, respectively. Each element is 0.01 [m] wide and 0.02 [m] high. Mesh thickness is 1.0 [m]. The vertical coordinate, x, is measured downward from the top of the column.

The time step is constant at $\Delta t = 30.0$ [s], and because of the small time step, only one iteration is done per step. The simulation is carried out for nine hours of infiltration.

Outputs are obtained once each hour, but are only compared at two hours and nine hours. There is one pressure solution and one concentration solution each time step.

Parameters:

$$k_r = 1.235376 \times 10^{-6} \exp(13.604 \ \ S_w) \tag{6.10}$$

$$S_w = \begin{cases} 1.52208 - 0.0718947 \ln(-p) \\ \qquad \text{for } -2892.38 < p \le -1421.96 \ [\text{kg}/(\text{m} \cdot \text{s}^2)] \\ 2.94650 - 0.250632 \ln(-p) \\ \qquad \text{for } p < 2892.38 \ [\text{kg}/(\text{m} \cdot \text{s}^2)] \end{cases} \tag{6.11}$$

$S_{op} = 0.0$ [m · s^2/kg]

$\rho = 1000.$ [kg/m^3]

$k = 4.4558 \times 10^{-13}$ [m^2]

$\sigma_w = 0.0$ [m^2/s]

$\varepsilon = 0.38$

$\alpha_L = 0.01$ [m]

$\mu = 1.0 \times 10^{-3}$ [kg/m · s] $\alpha_T = 0.0$ [m]

$|\underline{g}| = 9.81$ [m/s^2]

Boundary Conditions:

The top boundary, which represents an infiltration pond, is held fully saturated, $S_w = 1.0$ (water content $\varepsilon S_w = 0.38$) during the simulation by specification of pressure at $p = -1421.96$ [kg/(m · s^2)]. The bottom boundary is held at a specified saturation of $S_w = 0.526316$, (water content $\varepsilon S_w = 0.20$) by specification of pressure, $p = -15616.5$ [kg/(m · s^2)]. No flow occurs across either side boundary, but flow enters the top boundary due to the pressure specification. The concentration of inflowing fluid at the top is held at $C = 209.0$ [meq/liter] until time $t = 168.0$ [min], at which time the concentration of the inflow drops to $C = 0.0$ [meq/liter]. Note that the concentration units are arbitrary (need not be mass fractions) because this is a constant-density simulation.

Initial Conditions:

Initially, pressures are set to obtain the following initial distribution of saturation, shown in Figure 6.19:

$$S_w(x, t = 0) = \begin{cases} 0.394737 + 0.219289x & 0.0 < x \le 0.60\,[\text{m}] \\ 0.526316 & 0.6 < x \le 1.25\,[\text{m}] \end{cases} \qquad (6.12)$$

Initial concentrations are set to zero.

Results:

SUTRA results after two hours and nine hours of infiltration are shown with the finely discretized solutions of Van Genuchten (1982) for saturation in Figure 6.19, and for concentration in Figure 6.20. The results coincide almost exactly for both early and late time, so only one curve can be shown for each time. Although the SUTRA results are obtained with a noniterative solution and small time steps, similar results may be obtained with longer time steps and a few iterations per step. The concentration front lags behind the moisture front, as the volume between the concentration front and top boundary represents the water that has infiltrated. The volume of water between the moisture front and concentration front represents the initial water in the medium that has been displaced by the infiltrating water.

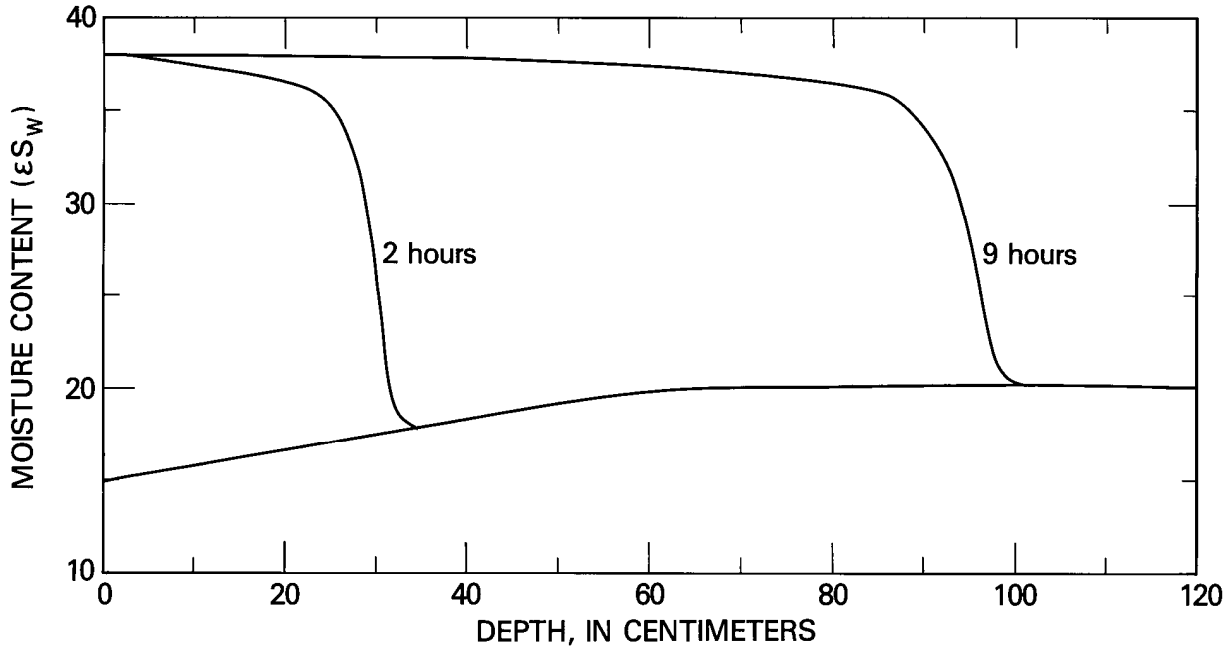


Figure 6.19. Propagation of moisture front for unsaturated flow and solute-transport example. Results of Van Genuchten (1982) and SUTRA shown in same solid line. The lowest curve is the initial condition.

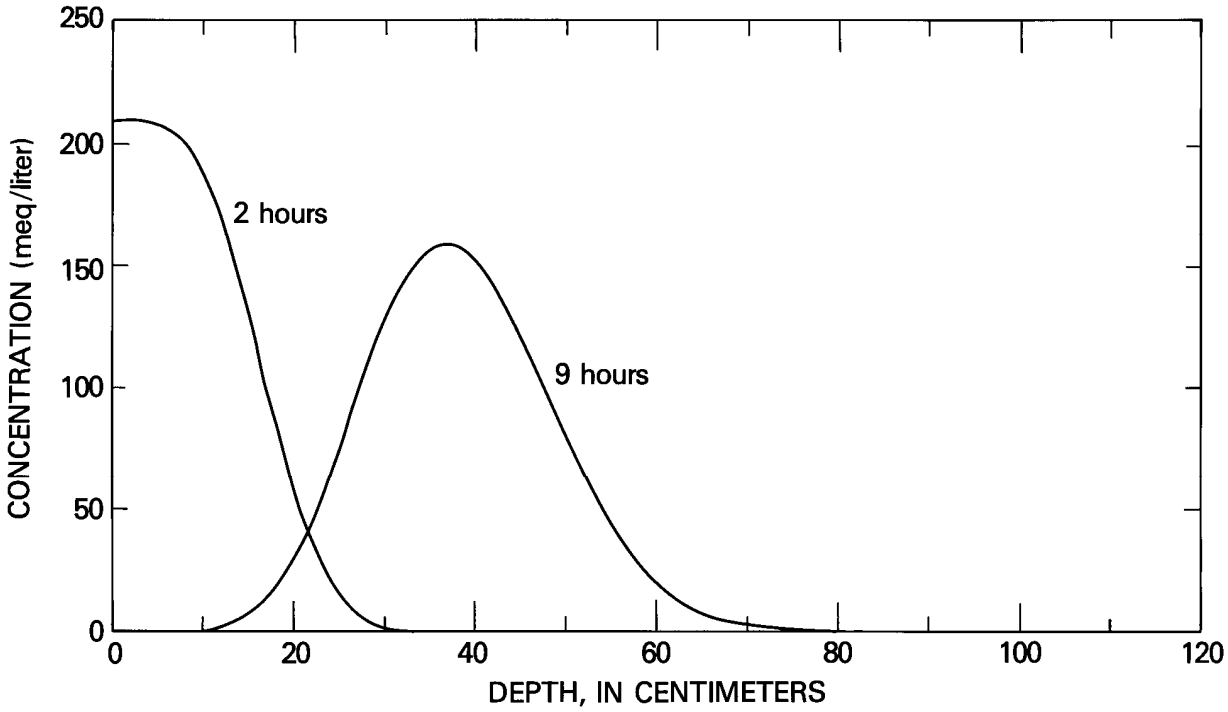


Figure 6.20. Propagation of solute slug for unsaturated flow and solute-transport example. Results of Van Genuchten (1982) and SUTRA shown in same solid line.

6.8 Variable-Density Saturated-Unsaturated Flow and Solute Transport (Comparison of 2D-Radial and Fully 3D SUTRA Solutions)

Physical Setup:

A circular island in the sea undergoes a prolonged drought, during which the water table declines to sea level, and all ground water beneath the island becomes saline. Then, renewed freshwater recharge finally restores the island's freshwater lens. This example concerns simulation of the post-drought recharge and restoration of the lens.

Following the drought, the water table is at sea level and both the saturated aquifer below sea level and the unsaturated zone above sea level contain only seawater. Fresh rainwater recharge to the surface of the island begins and continues at a constant rate, raising the water table on the island, flushing out seawater, and eventually establishing a stable freshwater lens and a diffuse saltwater-freshwater interface. The aquifer on the island is unconfined with both unsaturated and saturated zones and the material properties are generally homogeneous but permeability is anisotropic.

Objective:

The problem is simulated twice, using both a 2D radial mesh and a fully 3D, vertically aligned mesh that is irregular in the two "horizontal" node numbering directions and layered in the vertical direction. The 3D steady-state solution is compared with the 2D solution to verify that 3D simulation gives results equivalent to those obtained using the well-established 2D SUTRA code. Although the solution is radially symmetric, the 3D simulation must arrive at this result using rectangular (x, y, z) (not radial) coordinates and a finite-element mesh that does not inherently favor a radially symmetric solution. To reduce the size of the simulation, the 3D mesh represents only one fourth of the entire island, taking some advantage of radial symmetry of the 3D solution.

Simulation Setup:

The 2D mesh has 60 elements in the radial direction and 25 elements vertically, giving NN=1,586 and NE=1,500. See Figure 6.21. Elements are 20 m wide and 5 m high, except within 5 m of the top surface, where they are 1 m high, and within 100 m of the coast, where they are (200 m)/30 ≈ 6.7 m wide. Vertical discretization in the unsaturated zone is relatively coarse because, in this problem, the intent is to approximately locate the water table and the details of the saturation distribution are of less interest. Mesh thickness at node i is given by $B_i=2\pi r_i$, thereby providing a radial coordinate system.

The 3D mesh is discretized vertically into 25 layers of elements, with 1,567 elements in each layer, giving NN=42,432 and NE=39,175. See Figures 6.22 and 6.23. Symmetry is invoked to reduce the size of the problem while maintaining a fully 3D mesh; only one quadrant of the island is simulated. The outer boundary approximates a circle of radius 800 m and is sufficiently distant from the island that it does not significantly influence the results.

The runs are transient in both pressure and concentration. The time step size is $\Delta t = 6311520.$ s (0.2 yr). Because only the long-time (steady-state) behavior of the system is of interest, a single iteration for resolving nonlinearities is used per time step. The system essentially achieves a new steady state after 100 time steps (20 yr).

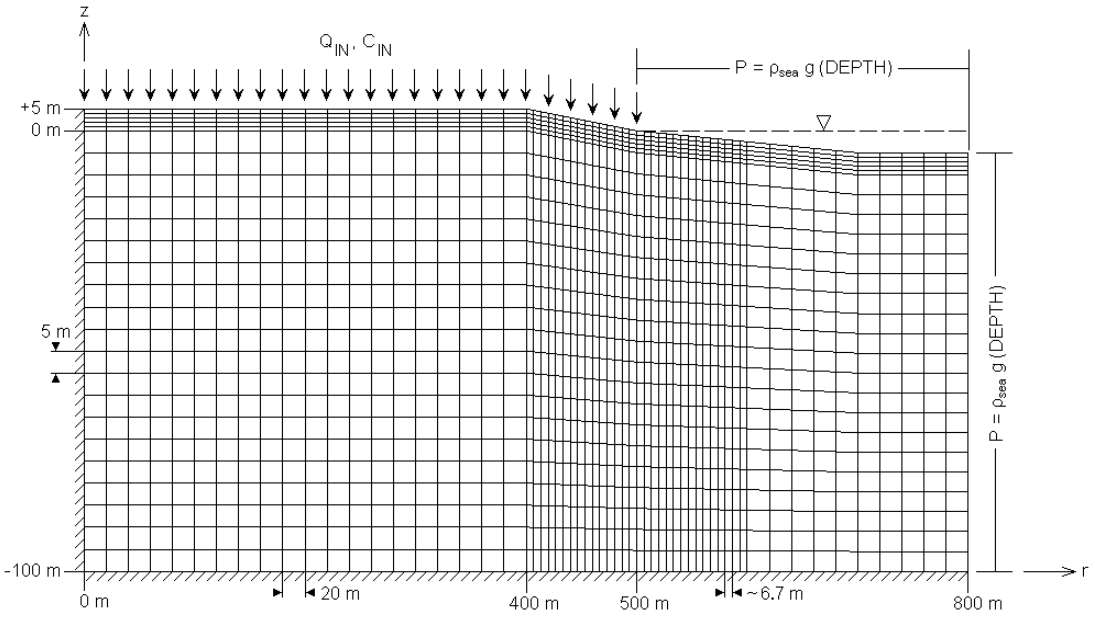


Figure 6.21. Boundary conditions and finite-element mesh for the 2D island model. Vertical exaggeration = 4x.

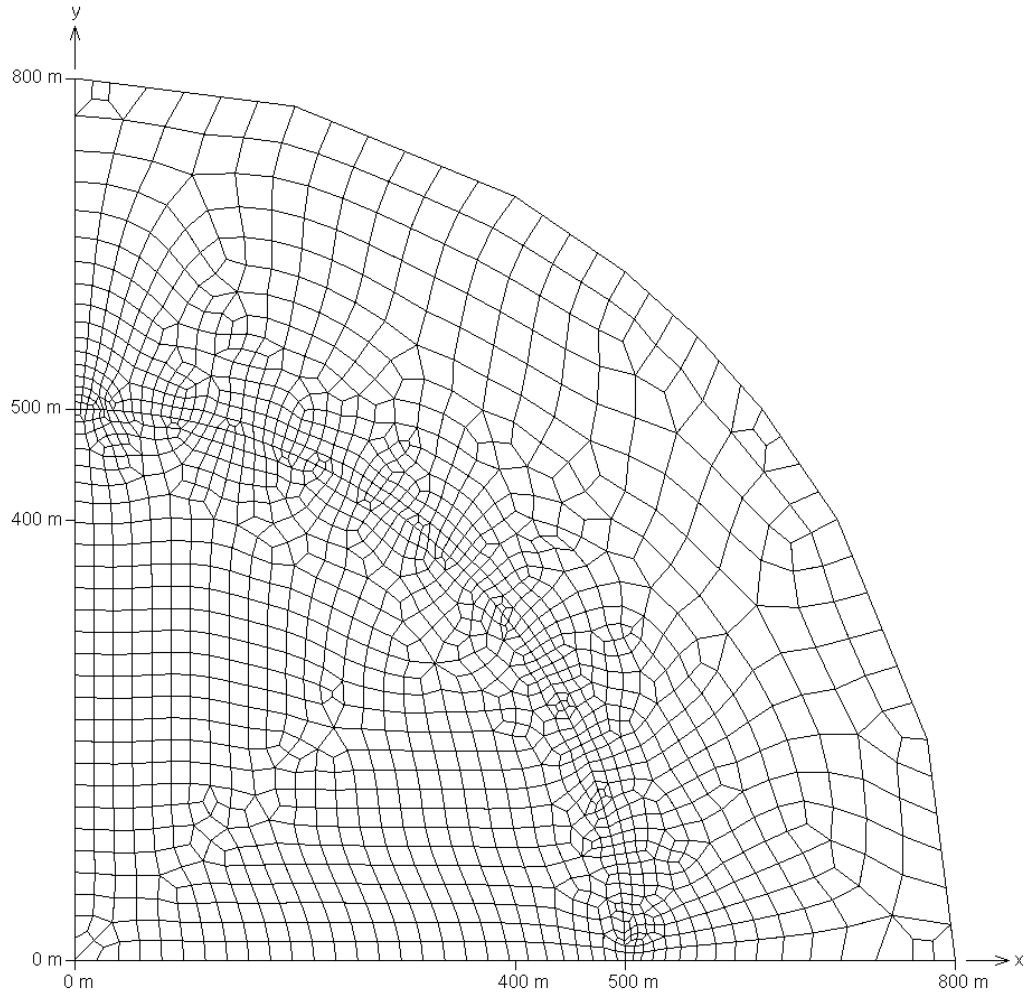


Figure 6.22. Top view of the 3D finite-element mesh for the island model.

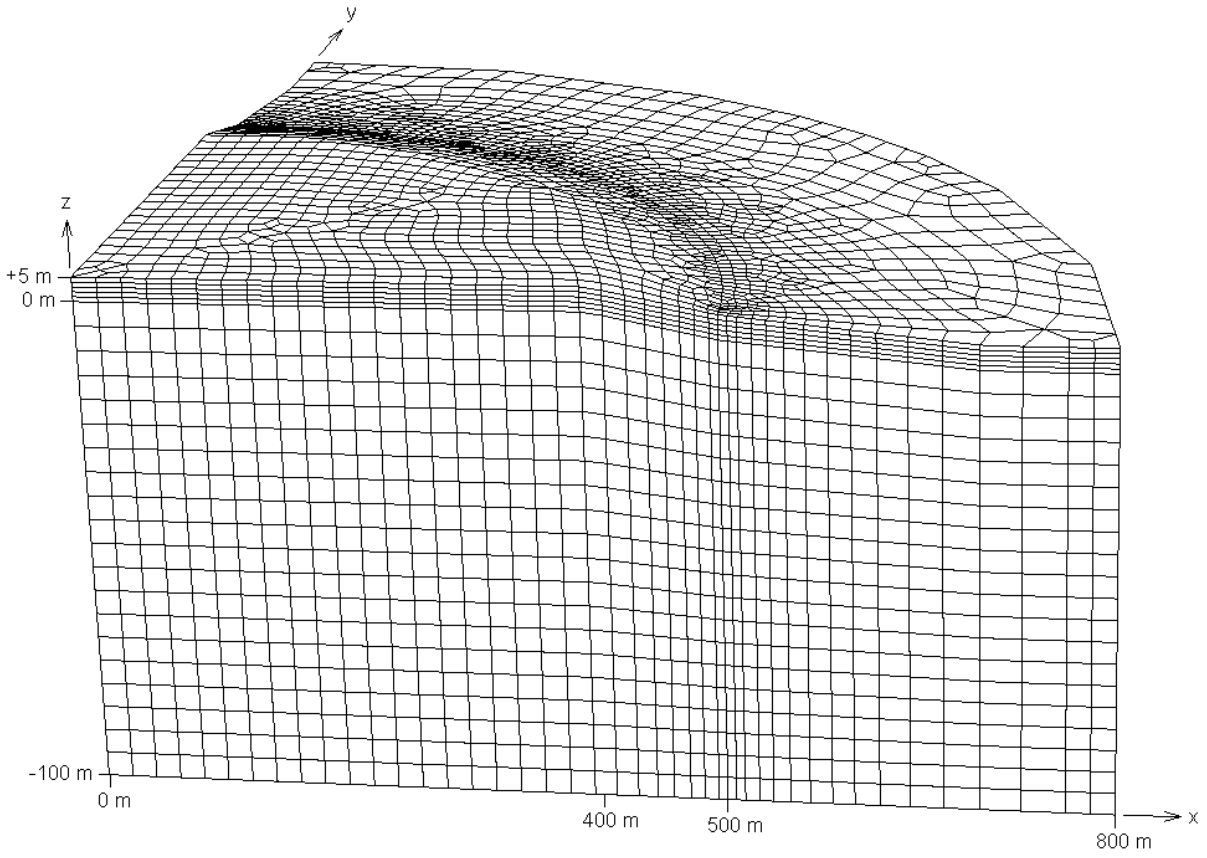


Figure 6.23. Oblique view of the 3D finite-element mesh for the island model. Vertical exaggeration = 4x.

For 2D, the direct solver is used. For 3D, the CG solver is used for p solutions, and the ORTHOMIN solver is used for C solutions, both with a convergence tolerance of 1×10^{-13}. Each 3D p solution requires from 5 to 29 solver iterations, and each 3D C solution requires from 5 to 17 iterations.

Parameters:

$\alpha = 1.0 \times 10^{-8}$ [m·s^2/kg] $\beta = 4.47 \times 10^{-10}$ [m·s^2/kg]

(The α and β values imply that $S_{op} = 9.0447 \times 10^{-9}$ [m·s^2/kg].)

$k_H = 5.0 \times 10^{-12}$ [m^2] $\varepsilon = 0.1$

$k_V = 5.0 \times 10^{-13}$ [m^2] $\mu = 1.0 \times 10^{-3}$ [kg/m·s]

$|\underline{g}| = 9.81$ [m/s^2] $\sigma_w = 1.0 \times 10^{-9}$ [m^2/s]

$\rho_0 = 1000.$ [kg/m^3]

$\rho_{sea} = 1024.99$ [kg/m^3]

$$C_{sea} = 0.0357 \left[\frac{(\text{kg dissolved solids})}{(\text{kg seawater})} \right]$$

$$\frac{\partial \rho}{\partial C} = 700. \left[\frac{(\text{kg seawater})/\text{m}^3}{(\text{kg dissolved solids})} \right]$$

$$\alpha_L = \begin{cases} 10. & \text{for horizontal flow} \\ 2.5 & \text{for vertical flow} \end{cases} \quad [\text{m}]$$

$D_m = 1.0 \times 10^{-9}$ [m^2/s] $\alpha_T = 0.1$ [m]

Q_{IN} = 2.3766x10^{-5} [kg/(m^2 of horizontally projected land surface area)·s] on land
(equivalent to 75. [cm/yr] of recharge)

$$C_{IN} = 0. \left[\frac{(\text{kg dissolved solids})}{(\text{kg seawater})} \right]$$

$$\left.\begin{aligned} S_w &= \begin{cases} 1 \text{ for } p \geq 0 \\ 0.3 + 0.7\left[1 + \left(5 \times 10^{-5}\ p\right)^2\right]^{-0.5} \text{ for } p < 0 \end{cases} \\ k_r &= \left(\frac{S_w - 0.3}{0.7}\right)^{0.5} \left(1 - \left\{1 - \left(\frac{S_w - 0.3}{0.7}\right)^2\right\}^{0.5}\right)^2 \end{aligned}\right\}$$

Unsaturated properties functions of Van Genuchten (1980) with $S_{wres} = 0.3$, $a = 5 \times 10^{-5}$ [m · s^2/kg], and n = 2. (See eqns. 2.8 and 2.21.)

Boundary Conditions:

In the 2D cylindrical model (see Figure 5.1), no flow crosses the inner boundary (the axis of radial symmetry, r = 0) and the bottom boundary (z = -100 m). Specified pressure is set at hydrostatic seawater pressure along the vertical outer boundary (r = 800 m). Along the top boundary, nodes at or above sea level (r ≤ 500 [m]) receive freshwater recharge (equivalent to 75.0 cm/yr) totaling 18.665773 kg/s of recharge for the entire circular island. The amount of recharge at each node is determined by the surface area of its cell on the top surface of the cylindrical 2D model; the concentric ring-shaped cells have areas that increase as 2πr. In the region where the island surface slopes down towards the coast, 400 m ≤ r ≤ 500 m, the surface area used for calculating recharge is the horizontal projection of this sloping area (area reduced by cosine of the dip angle). At nodes below sea level, the pressure is specified to be hydrostatic seawater pressure. Any fluid that enters at points of specified pressure has the concentration of seawater. The value for the specified pressure boundary condition factor, GNUP, in 2D is 1.0x10^{+5}.

In the 3D model, no flow crosses the planes of symmetry (x = 0 and y = 0). All other 3D boundary conditions are directly analogous to those in the 2D formulation. The value for the specified pressure boundary condition factor, GNUP, in 3D is 100.0. If 2D and 3D simulations are set up using the graphical preprocessor, SutraGUI (Winston and Voss, 2003), small discrepancies may be expected between these models in some parameters, such as recharge to the top surface. However, despite the obvious differences in spatial discretization, according to the fluid budgets output by SUTRA, the total recharge to the top surface of the entire island in 2D and 3D representations matches to five significant figures: 18.665773 kg/s in 2D and 18.665651 kg/s in 3D. Note that the 3D value is for the entire island; because of symmetry, only one-quarter of the island is simulated.

Initial Conditions:

Seawater concentration and natural steady-state pressures are initially set everywhere in the aquifer. The natural initial pressure values are obtained through an extra initial simulation that calculates steady pressures for the conditions of seawater concentration throughout, zero recharge at the surface of the island, and specified hydrostatic pressures along the sea bottom and the outer boundary.

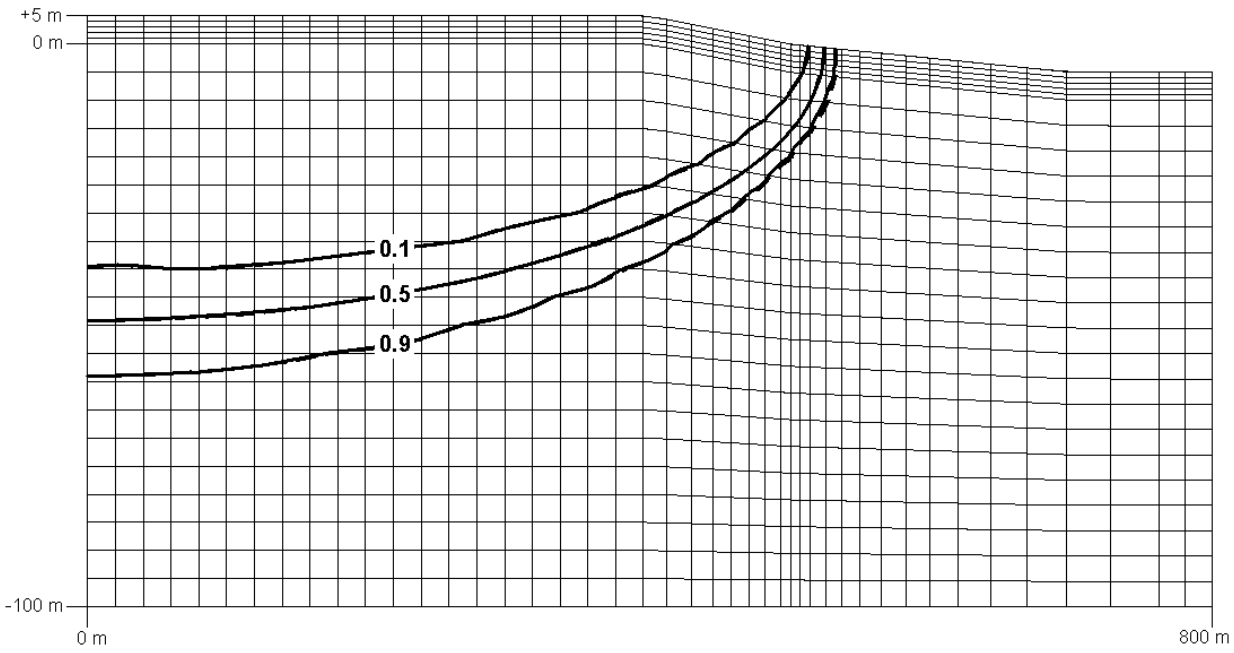


Figure 6.24. Comparison of results from the 2D and 3D models of the island problem; solute concentrations at t = 20 yr. Solid lines indicate 2D results. Dashed lines indicate 3D results, which are shown within the vertical 3D plane y = 0 m. Concentration is expressed as the fraction of seawater concentration. The 3D mesh is shown at y = 0 m. Vertical exaggeration = 4x.

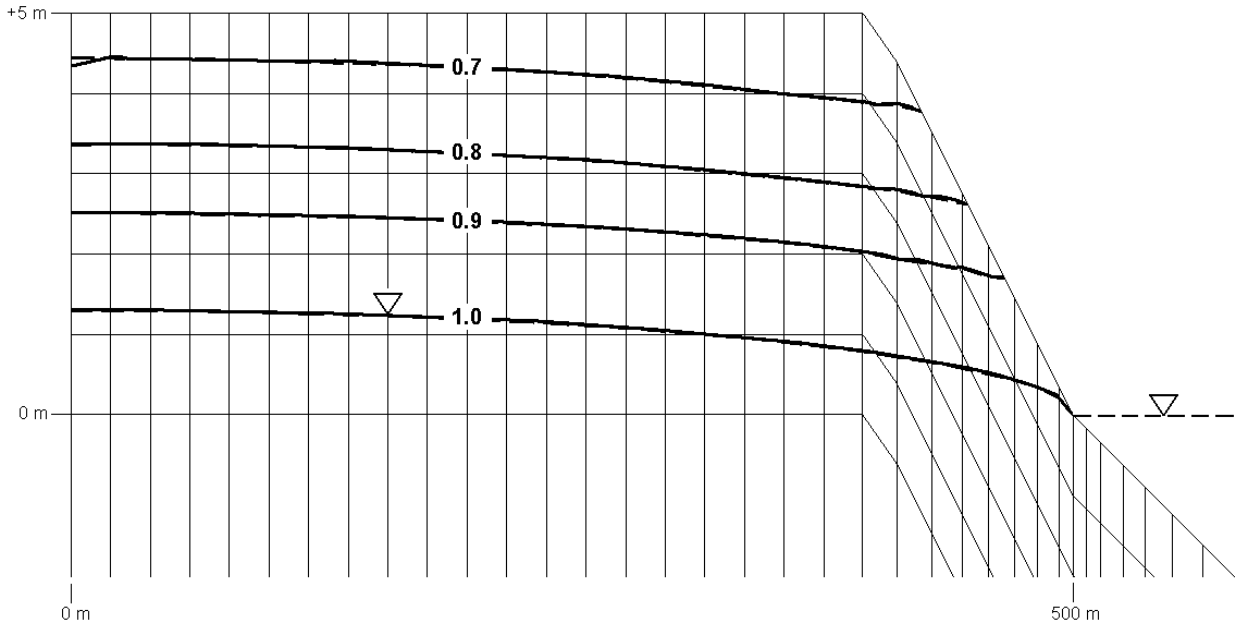


Figure 6.25. Comparison of water saturation, S_w, results from the 2D and 3D models of the island problem at t = 20 yr. Thick solid line indicates 2D result. Thick dashed line indicates 3D result, which is shown within the vertical 3D plane y = 0 m. Thin dashed line indicates sea level. Water saturations of one plotted along zero pressure contours from 2D and 3D models.The 3D mesh is shown at y = 0 m. Vertical exaggeration = 40x.

Results:

Results are reported 20 years after recharge begins, by which time the system has nearly reached a new steady state. (Note that more-exact steady-state solutions may be obtained by running longer simulations, e.g. 40 years or more.) To verify that results from the 2D and 3D models are consistent, solute concentrations, saturations and the water table location along a cross section of the 3D model at y=0 are compared with those obtained using the 2D model. See Figures 6.24 and 6.25. In both figures, contoured results from the 2D and 3D models are practically identical and are difficult to distinguish. Further, to verify that the 3D results are radially symmetric, concentrations at 35 m below sea level are plotted in Figure 6.26. This example demonstrates that (even for relatively coarse meshes) 2D SUTRA simulation and 3D SUTRA simulation provide consistent saturated-unsaturated, variable-density fluid flow and solute transport results.

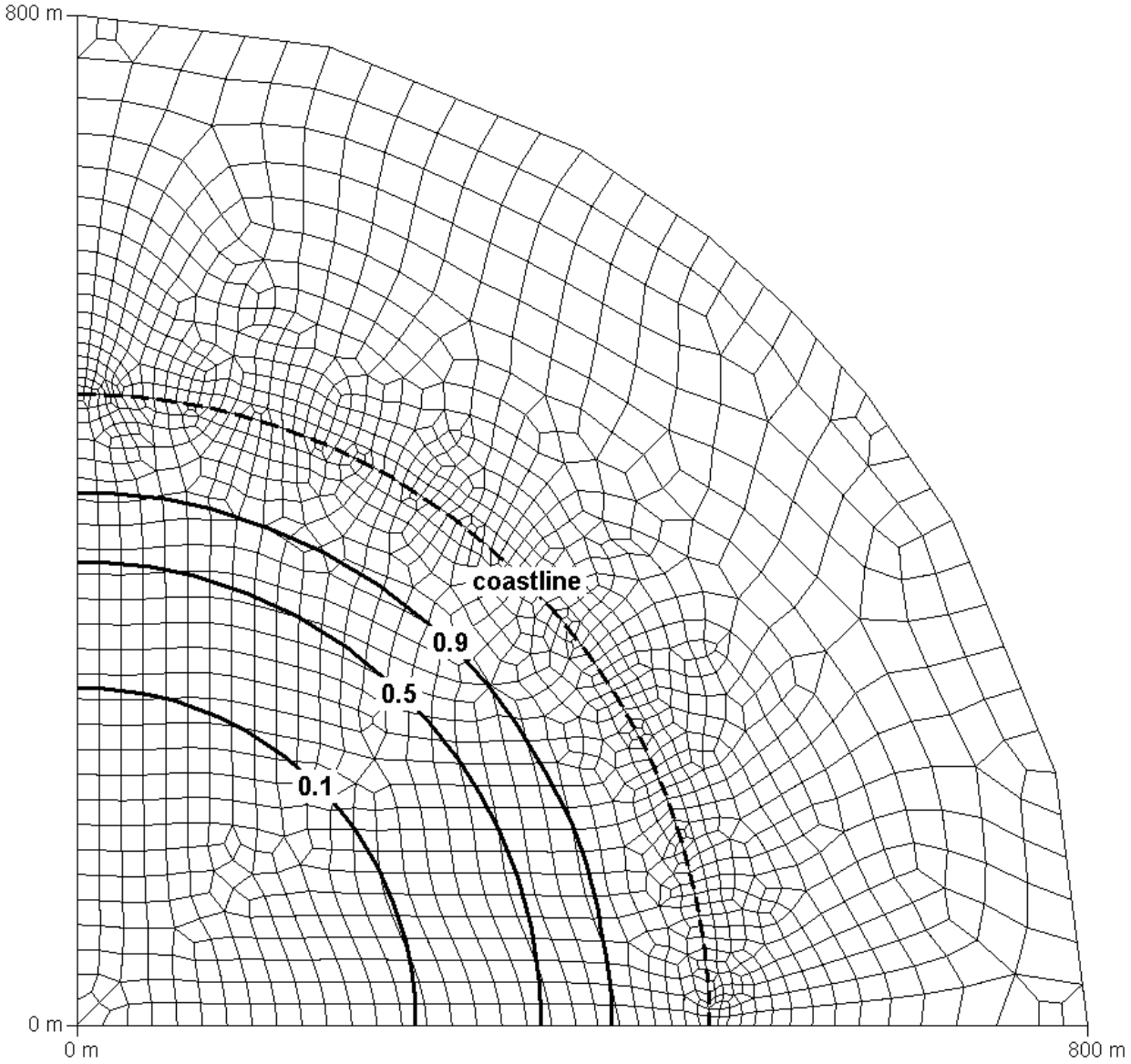


Figure 6.26. Areal view of results from the 3D model of the island problem; solute concentrations at 35 m below sea level at t = 20 yr. Concentration is expressed as the fraction of seawater concentration. Dashed line indicates the coastline, which is circular.

SUTRA SIMULATION SETUP

Chapter 7: Simulation Setup

7.1 SUTRA Data Requirements

The following is a complete list of data required to setup a simulation with SUTRA. (1) The information included in the list is the parameter name used in this report (if it has been mentioned), (2) the parameter units, (3) the parameter name in the input data list, and (4) a short explanation of the parameter.

Mesh and coordinate data

g_x	$[L/s^2]$	GRAVX	x-component of gravity vector
g_y	$[L/s^2]$	GRAVY	y-component of gravity vector
g_z	$[L/s^2]$	GRAVZ	z-component of gravity vector (3D only)
x_i	[L]	X(I)	x coordinate of node i, for all nodes in mesh
y_i	[L]	Y(I)	y coordinate of node i, for all nodes in mesh
z_i	[L]	Z(I)	z coordinate of node i, for all nodes in mesh (3D only)
		MSHSTR	mesh structure: "2D IRREGULAR" "2D REGULAR" "2D BLOCKWISE" "3D IRREGULAR" "3D LAYERED" "3D REGULAR" "3D BLOCKWISE"
NN		NN	total number of nodes in mesh
NN1		NN1	number of nodes in the first numbering direction (REGULAR or BLOCKWISE mesh only)
NN2		NN2	number of nodes in the second numbering direction (REGULAR or BLOCKWISE mesh only)
		NN3	number of nodes in the third numbering direction (REGULAR or BLOCKWISE mesh only)

		IIN(1-8)	nodal incidence list in each element (for 2D, only IIN(1-4) are used)
NE		NE	total number of elements in mesh
		NBLK1	number of blocks in the first numbering direction (BLOCKWISE mesh only)
		NBLK2	number of blocks in the second numbering direction (BLOCKWISE mesh only)
		NBLK3	number of blocks in the third numbering direction (3D BLOCKWISE mesh only)
		LDIV1	list of number of elements into which to divide blocks along first numbering direction (BLOCKWISE mesh only)
		LDIV2	list of number of elements into which to divide blocks along second numbering direction (BLOCKWISE mesh only)
		LDIV3	list of number of elements into which to divide blocks along third numbering direction (3D BLOCKWISE mesh only)
		NLAYS	number of layers of nodes (LAYERED mesh only)
		NNLAY	number of nodes in a layer (LAYERED mesh only)
		NELAY	number of elements in a layer (LAYERED mesh only)
		LAYSTR	first numbering direction (LAYERED mesh only): "ACROSS" "WITHIN"

Flow parameters

β	$[M/(L\cdot s^2)]^{-1}$	COMPFL	fluid compressibility
α	$[M/(L\cdot s^2)]^{-1}$	COMPMA	solid matrix compressibility
μ	[1]	VISC0	for energy transport: scale factor for fluid viscosity
	or		
	$[M/(L\cdot s)]$	VISC0	for solute transport: fluid viscosity

ε_i	[1]	POR(I)	volumetric porosity of solid matrix at each node
k_{max_L}	[L^2]	PMAX(L)	maximum component of permeability in each element
k_{mid_L}	[L^2]	PMID(L)	middle component of permeability in each element (3D only)
k_{min_L}	[L^2]	PMIN(L)	minimum component of permeability in each element
θ_L or θ_{1L}	[°]	ANGLE1(L)	angle from +x-axis to k_{max}-axis in each element, measured within x,y-plane; denoted by θ_L in 2D and by θ_{1L} in 3D
θ_{2L}	[°]	ANGLE2(L)	angle from x,y-plane to k_{max}-axis in each element, measured vertically from x,y-plane (3D only)
θ_{3L}	[°]	ANGLE3(L)	angle from x,y-plane to k_{mid}-axis in each element, measured as counterclockwise rotation about k_{max}-axis (looking down $+k_{max}$-axis toward origin) (3D only)
ρ_o	[M/L^3]	RHOW0	fluid base density
$\frac{\partial \rho}{\partial U}$	[$M/L^3 \cdot °C$]	DRWDU	for energy transport: coefficient of fluid density change with temperature
	or		
	[$M^2/L^3 \cdot M_s$]	DRWDU	for solute transport: coefficient of fluid density change with concentration
U_o	[°C]	URHOW0	for energy transport: base temperature for density calculation
	or		
	[M_s/M]	URHOW0	for solute transport: base concentration for density calculation

Transport parameters

α_{Lmax_L}	[L]	ALMAX (L)	radius of longitudinal dispersivity ellipse (2D) or ellipsoid (3D) in direction of k_{max} in each element

α_{Lmid_L}	[L]	ALMID (L)	radius of longitudinal dispersivity ellipsoid in direction of k_{mid} in each element (3D only)
α_{Lmin_L}	[L]	ALMIN (L)	radius of longitudinal dispersivity ellipse (2D) or ellipsoid (3D) in direction of k_{min} in each element
α_{Tmax_L}	[L]	ATMAX (L)	radius of transverse dispersivity ellipse (2D) or ellipsoid (3D) in direction of k_{max} in each element
α_{Tmid_L}	[L]	ATMID (L)	radius of transverse dispersivity ellipsoid in direction of k_{mid} in each element (3D only)
α_{Tmin_L}	[L]	ATMIN (L)	radius of transverse dispersivity ellipse (2D) or ellipsoid (3D) in direction of k_{min} in each element
σ_w	[E/(L·°C·s)]	SIGMAW	for energy transport: fluid thermal conductivity
	or		
	[m^2/s]	SIGMAW	for solute transport: molecular diffusivity of solute in fluid
σ_s	[E/(L·°C·s)]	SIGMAS	for energy transport: solid grain thermal conductivity (equals zero for solute transport)
c_w	[E/(M·°C)]	CW	for energy transport: fluid specific heat capacity (equals one for solute transport)
c_S	[E/(M·°C)]	CS	for energy transport: solid grain specific heat capacity (not specified in input data for solute transport)
ρ_s	[M/L^3]	RHOS	density of a solid grain in the solid matrix

Reaction and production parameters

Linear Sorption Isotherm

χ_1	[L_f^3/M_G]	CHI1	linear distribution coefficient (2.34a) (χ_2 is zero for this isotherm)

Freundlich Sorption Isotherm

χ_1	$[L_f^3/M_G]$	CHI1	Freundlich distribution coefficient (2.35a)
χ_2	[1]	CHI2	Freundlich coefficient (2.35a)

Langmuir Sorption Isotherm

χ_1	$[L_f^3/M_G]$	CHI1	Langmuir distribution coefficient (2.36a)
χ_2	$[L_f^3/M_s]$	CHI2	Langmuir coefficient (2.36a)

Production

γ_1^w	$[s^{-1}]$	PRODF1	for solute transport: rate of first-order production of adsorbate mass in the fluid mass (equals zero for energy transport)
γ_1^s	$[s^{-1}]$	PRODS1	for solute transport: rate of first order production of solute mass in the immobile phase (equals zero for energy transport)
γ_o^w	[(E/M)/s]	PRODFØ	for energy transport: zero-order rate of energy production in the fluid
	….or		
	$[(M_s/M)/s]$	PRODFØ	for solute transport: zero-order rate of solute mass production in the fluid
γ_o^s	$[(E/M_G)/s]$	PRODSØ	for energy transport: zero-order rate of energy production in the immobile phase
	….or		
	$[(M_s/M_G)/s]$	PRODSØ	for solute transport: zero-order rate of adsorbate mass production in the immobile phase

Boundary conditions and source data

Flow Data—Specified Pressures

NPBC	NPBC	number of nodes at which pressure is a specified constant or function of time
$IPBC_{ipu}$	IPBC(IPU)	node number at which pressure is specified (for all NPBC nodes)

PBC_{ipu}	$[M/(L \cdot s^2)]$	PBC(IPU)	value of specified pressure at node IPBC (for all NPBC nodes)
UBC_{ipu}	[°C]	UBC(IPU)	for energy transport: value of temperature of any fluid that enters the system at node IPBC
	or		
	$[M_s/M]$	UBC(IPU)	for solute transport: value of concentration of any fluid that enters the system at node IPBC

Flow Data—Specified Flows and Fluid Sources

NSOP		NSOP	number of nodes at which a source of fluid mass is specified
$IQCP_{iqp}$		IQCP, IQSOP (IQP)	node number at which a fluid source is specified (for all NSOP nodes)
Q_{IN_i}	[M/s]	QINC, QIN(I)	fluid source rate at source node IQCP (for all nodes)
U_{IN_i}	[°C]	UINC, UIN(I)	for energy transport: value of temperature of any fluid that enters the system at source node IQCP
	or		
	$[M_s/M]$	UINC, UIN(I)	for solute transport: value of concentration of any fluid that enters the system at source node IQCP

Energy or Solute Data—
Specified Temperatures or Concentrations

NUBC		NUBC	number of nodes at which temperature or concentration is a specified constant or function of time.
$IUBC_{ipu}$		IUBC(IPU)	node number at which temperature or concentration is specified (for all NUBC nodes)

UBC	[°C]	UBC(IPU)	for energy transport: value of specified temperature at node IUBC (for all NUBC nodes)
	or		
	[M_s/M]	UBC(IPU)	for solute transport: value of specified concentration at node IUBC (for all NUBC nodes)

Energy or Solute Data—
Diffusive Fluxes of Energy or Solute Mass at Boundaries

NSOU		NSOU	number of nodes at which a diffusive energy or solute mass flux (source) is specified
IQCU		IQCU, IQSOU(IQU)	node number at which a flux (source) is specified (for all NSOU nodes)
Ψ_{IN_i}	[E/s]	QUINC	for energy transport: energy flux (source) rate at node IQCU (one value for each of NSOU nodes)
	or		
	[M_s/s]	QUINC	for solute transport: solute mass flux (source) rate at node IQCU (one value for each of NSOU nodes)

Initial conditions

	[s]	TICS	time to which the initial conditions correspond (not necessarily not equal to the starting time of the simulation, t_o)
		CPUNI	= "UNIFORM" (uniform initial P) = "NONUNIFORM" (nonuniform initial P)
		CUUNI	= "UNIFORM" (uniform initial U) = "NONUNIFORM" (nonuniform initial U)
$p_i(t=t_o)$	[M/(L·s^2)]	PVEC(II)	initial pressure at all nodes in mesh (for UNIFORM, a single value; for NONUNIFORM, a list of values)

$U_i(t=t_o)$	[°C]	UVEC(II)	for energy transport: initial temperature at all NN nodes in the mesh (for UNIFORM, a single value; for NONUNIFORM, a list of values)
	or		
	[M_s/M]	UVEC(II)	for solute transport: initial concentration at all NN nodes in the mesh (for UNIFORM, a single value; for NONUNIFORM, a list of values)

Numerical and temporal control data

ν_{p_i}	[Ls]	GNUP	specified pressure boundary condition "conductance" factor (4.111)
ν_{U_i}	[Ls]	GNUU	specified concentration boundary condition "conductance" factor (4.143)
UP	[1]	UP	fractional upstream weight for asymmetric weighting functions (4.23) and (4.24) in 2D, and (4.66) – (4.68) in 3D
		NSCH	number of schedules
		SCHNAM	name of schedule (one of which, 'TIME_STEPS' defines the initial time of the simulation, t_o, and the subsequent time stepping)
		SCHTYP	schedule type: "TIME LIST" "TIME CYCLE" "STEP LIST" "STEP CYCLE"
		CREFT	time reference (time-based schedules only): "ABSOLUTE" "ELAPSED"
		SCALT	scale factor for times (time-based schedules only)
		NTLIST	number of times listed (TIME LIST schedules only)
		TLIST	list of times (TIME LIST schedules only)

	NTMAX	maximum number of time cycles allowed (TIME CYCLE schedules only)
	TIMEI	initial time for a time cycle (TIME CYCLE schedules only)
	TIMEL	limiting time for a time cycle (TIME CYCLE schedules only)
	TIMEC	initial time increment for a time cycle (TIME CYCLE schedules only)
	NTCYC	number of cycle after which the time increment is updated (TIME CYCLE schedules only)
	TCMULT	multiplier for time increment (TIME CYCLE schedules only)
	TCMIN	minimum time increment allowed (TIME CYCLE schedules only)
	TCMAX	maximum time increment allowed (TIME CYCLE schedules only)
	NSLIST	number of time steps listed (STEP LIST schedules only)
	ISLIST	list of time steps (STEP LIST schedules only)
	NSMAX	maximum number of time step cycles allowed (STEP CYCLE schedules only)
	ISTEPI	initial time step for a time step cycle (STEP CYCLE schedules only)
	ISTEPL	limiting time step for a time step cycle (STEP CYCLE schedules only)
	ISTEPC	time step increment for a time step cycle (STEP CYCLE schedules only)
NPCYC	NPCYC	time steps in pressure solution cycle
NUCYC	NUCYC	time steps in temperature or concentration solution cycle

	ITRMAX	maximum number of iterations for nonlinearities per time step
$[M/(L \cdot s^2)]$	RPMAX	pressure convergence criterion for iterations
[°C]	RUMAX	for energy transport: temperature convergence criterion for resolving nonlinearities
or		
$[M_s/M]$	RUMAX	for solute transport: concentration convergence criterion for resolving nonlinearities

Matrix equation solver data

CSOLVP	solver for p (flow equation): = "DIRECT" (Gaussian elimination) = "CG" = "GMRES" = "ORTHOMIN"
CSOLVU	solver for U (transport equation): = "DIRECT" (Gaussian elimination) = "GMRES" = "ORTHOMIN"
ITRMXP	maximum number of solver iterations during p solution
ITRMXU	maximum number of solver iterations during U solution
TOLP	convergence tolerance for solver iterations during P solution
TOLU	convergence tolerance for solver iterations during U solution

Data for options

CREAD	= "COLD" (new simulation – cold start) = "WARM" (restart simulation – warm start)
ISTORE	≥ 1 store simulation results for later restart = 0 do not store results

Simulation mode options

SIMULA { = "SUTRA ENERGY" (energy transport)
= "SUTRA SOLUTE" (solute transport)

CUNSAT { = "UNSATURATED" (sat/unsat flow)
= "SATURATED" (saturated flow)

CSSFLO { = "STEADY" (steady flow)
= "TRANSIENT" (transient flow)

CSSTRA { = "STEADY" (steady transport)
= "TRANSIENT" (transient transport)

Velocity Output Option

CVEL { = "Y" (output fluid velocity at element centroids)
= "N" (no velocity output)

Observation Option

NOBS number of observation points

NOBLIN maximum number of observations output to a single line in a ".obs" file

OBSNAM observation point name

XOBS X coordinate of observation point

YOBS Y coordinate of observation point

ZOBS Z coordinate of observation point (3D only)

OBSSCH name of schedule that controls output for an observation point

OBSFMT { output format for an observation point:
= "OBS"
= "OBC"

Budget Option

CBUDG { = "Y" (output fluid mass and energy or solute mass budgets to ".lst" file)
= "N" (no budgets)

Output Controls

CNODAL	= "Y" (output nodewise input data to ".lst" file) = "N" (cancel output)
CELMNT	= "Y" (output elementwise input data to ".lst" file) = "N" (cancel output)
CINCID	= "Y" (output incidence lists to ".lst" file) = "N" (cancel output)
NPRINT	results are output to ".lst" file every NPRINT time steps
NCOLPR	results are output to ".nod" file every NCOLPR time steps
LCOLPR	results are output to ".ele" file every LCOLPR time steps
NCOL	list of variables to be output in columns in the ".nod" file
LCOL	list of variables to be output in columns in the ".ele" file

7.2 Discretization Rules of Thumb

Proper discretization in space and time is the vital factor in obtaining accurate simulation of the physics of flow and transport with a numerical model such as SUTRA. Adequate discretization is vital for two reasons: 1) the ability of a model to represent the variations in system parameters and to simulate complex processes depends on the fineness of discretization, and, 2) the accuracy and stability of the numerical methods used to represent system processes, in particular, transport, depends on the spatial and temporal discretization. This section describes some general guidelines for designing adequate discretization for simulation with SUTRA.

A "sufficiently good" discretization allows for accurate simulation of the processes and parameter variations at the scale of interest, and thus the goodness of a discretization is a relative rather than absolute factor. A better discretization is always obtained by making existing discretization finer, but the finer the discretizations are, the more computationally expensive the simulations become.

Relative to a certain adequate level of fineness, even finer discretizations do not practically improve the accuracy of simulation. In contrast, discretization that is too coarse may completely obscure parameter variations and processes of interest in a simulation, and give highly inaccurate results. Unfortunately, simulation results based on inadequate discretization may appear to be a reasonably good representation of flow and transport physics in a particular system. The only way to explicitly check for inadequate discretization of a system is to simulate with a discretization that is assumed to be adequate and then with a significantly finer discretization and compare results. If there are no telling differences in the results, then the coarser simulation indeed has been adequately discretized.

Some general guidelines for obtaining adequate discretization, both for parameter representation and for accuracy and stability of numerical methods are given below.

1) Nodes are required where boundary conditions and sources are specified. Should accurate simulation of processes near these specified points be required, then a finer mesh is needed in these areas.

2) A finer mesh is required where parameters vary faster in space. This is often the case near sources or boundary conditions specifying inflows of fluid, solute or energy. The fineness required is that which makes the nodewise, cellwise, or elementwise discretization of the parameter values a good representation of the actual distributions. When a parameter distribution is known a priori, then this discretization is straightforward. However, when the parameter distribution depends on the simulation results then judgment must be exercised in discretization, and the result may be tested by experiment with various discretizations.

It is important to recognize that each node or element does not alone represent a physical entity in an aquifer system. This is demonstrated in the following example, which shows that one layer of elements is not a good representation in cross section of a semiconfining layer or aquifer unit. Although permeability is specified elementwise and the permeability of two aquifer units separated by confining layer, viewed in cross section, is clearly represented visually by three layers of elements, the numerical model does not “see” three distinct layers of permeability. Each node at the boundary of these layers experiences some average of the two permeabilities rather than either one. Thus, no node in the system experiences the assigned low permeability of confining layer, and the three-layer discretization is inadequate. More layers of elements are required in each unit to obtain adequate discretization although the model always experiences an average permeability in the elements making up the boundaries of the units. Further refinement of discretization would be required to represent the pressure distribution should accurate simulation of sharply varying pressures across the confining layer be required.

Discretization of the spatial distribution of transport variables, concentration or temperature, often is that which requires the finest mesh. The spatial distributions of these variables often include a “front” at which the concentration or temperature changes sharply from high values on one side to low values on the other side. A rule of thumb is that at least five elements should divide the front in order to guarantee that the simulated front width arises from simulated physical processes rather than from spreading due to inadequate discretization. When such fronts travel with the flow across a mesh during simulation, the mesh must be designed fine enough to adequately represent the front at all points along its path. In regions external to the front path, coarser discretization is usually adequate, and an expanding mesh may be used in this region.

3) The spatial stability of the numerical approximation of the unified transport equation (2.52) depends on the value of a mesh Peclet number, Pe_m, given by:

$$Pe_m = \frac{\varepsilon S_w \left|\underline{v}\right| \Delta L_L}{\left[\varepsilon S_w \left(\sigma_w + \alpha_L \left|\underline{v}\right|\right) + (1-\varepsilon)\sigma_s\right]} \tag{7.1}$$

where ΔL_L is the local distance between element sides along a streamline of flow. Spatial instability appears as one or more oscillations in concentration or temperature. Stability is guaranteed in all cases when $Pe_m \leq 2$, which gives a criterion for choosing a maximum allowable element dimension, ΔL_L, along the local flow direction. This criterion significantly affects discretization. Spatial stability is usually obtained with SUTRA when

$$Pe_m \leq 4 \tag{7.2}$$

which gives a less-stringent criterion. Mesh design according to the criterion is critical when concentrations or temperatures change significantly along streamlines, such as when a front is propagated in the direction of flow. When concentrations or temperatures exhibit small changes along streamlines, then the criterion (7.2) may safely be violated, even by a few orders of magnitude, without inducing spatial instability. An example of this may be cross sectional simulation of an aquifer containing freshwater and saltwater. In such a case, flow often is directed parallel to the front between freshwater and saltwater, allowing use of discretization with large mesh Peclet numbers.

In the typical case of solute or energy transport with longitudinal dispersion primarily due to longitudinal mixing, the mesh Peclet number becomes:

$$Pe_m \approx \left(\frac{\Delta L_L}{\alpha_L}\right) \tag{7.3}$$

A discretization rule of thumb for simulation with SUTRA that guarantees spatial stability in most cases is:

$$\Delta L_L \leq 4\alpha_L \tag{7.4}$$

While (7.4) deals with adequate discretization for numerical stability, it may be interpreted from another point of view. Taken in combination with the considerations of guideline (2) requiring at least five elements across a front, (7.4) implies that a minimum front width which may be simulated when the mesh is designed according to $\Delta L_L \sim 4\alpha_L$ is $20\alpha_L$. Thus for early times following onset of localized energy or solute source, the sharp front that should result may be simulated inaccurately, as its width is less than $20\alpha_L$.

4) Discretization for transverse dispersion also may be related to dispersivity. Although an exact guideline is not given, the object of transverse discretization is to make the local element dimension perpendicular to a streamline small relative to the total transverse dispersivity:

$$\Delta L_T < \alpha_T + \frac{1}{|\underline{v}|}\left[\varepsilon S_w \sigma_w + (1-\varepsilon)\sigma_s\right] \tag{7.5}$$

where ΔL_T is the local element dimension transverse to the flow direction. In the case where the transverse mixing rather than diffusion dominates the transverse dispersion, an adequate but stringent rule of thumb may be $\Delta L_T < 10\alpha_T$, although simulation results should be compared for various transverse discretizations.

5) Radial/cylindrical meshes with a well require very fine discretization near the center axis to accommodate the sharply curving pressure distribution. The radial element dimensions may increase outward and become constant at, for example, a size of $4\alpha_L$.

6) Unsaturated flow simulation requires at least as fine discretization as does transport. Spatial instability appears as an oscillation in saturation values. Unsaturated flow parameters may vary sharply in space, especially during wetting events. A rule of thumb is to design the mesh to have at least five elements across a saturation front.

7) Discretization in time is done by choosing the size of time steps. Actual time step sizes may be as large as possible while providing adequate discretization of parameter changes in time. As with spatial discretization, the adequacy of a temporal discretization may be tested only by comparing results of simulations carried out with different time step sizes.

For saturated flow simulation, temporal discretization begins with fine time steps, which may become significantly larger as the system response slows. The time-step multiplier feature is provided in SUTRA input data to allow this type of temporal discretization.

For unsaturated flow simulation with SUTRA, temporal discretization must be fine enough to keep saturation changes at each node to be small over any time step. A rule of thumb is that over a time step, the maximum saturation change is about 0.1.

For transport simulation, temporal changes in concentration or temperature at a point in space are often due to the movement of fronts with the fluid flow. Therefore, adequate discretization of these parameters in time is often related to both fluid velocity and spatia1 gradients in the parameters. The higher the longitudinal spatial gradient and fluid velocity, the smaller the time step required for adequate temporal discretization. A general guideline is that relatively sharp fronts require time discretization, which allows them to move only a fraction of an element per time step. Broad fronts with low gradient in concentration or temperature have adequate temporal discretization when time steps are chosen to move the front one or more elements per step.

Usually a constant time step size is chosen for transport simulation when flow velocities remain relatively constant during a simulation. For saturated flow and transport, if adequate temporal pressure discretization would allow larger time steps than the temporal transport discretization, then a pressure solution may be done only every n time steps for transport. For example, if the adequate pressure time step is ten times that of transport, then SUTRA input data requires the specification: NPCYC = 10, NUCYC = 1.

7.3 Program Dimensions

The main program computes the dimensions of the various arrays used in the SUTRA code. These arrays are dynamically allocated in the main program. The table below lists the maximum total storage required by **SUTRA *Version 2.1*** for dynamically allocated arrays of real numbers, integers, character variables, and pointers, depending on the dimensionality of the problem and the type of solver(s) used:

		sum of real array dimensions	sum of integer array dimensions	sum of character array effective dimensions*	sum of dimensions of arrays of pointers
2D	direct solver	(2*NBI + 27)*NN + 19*NE + 3*NBCN + 6*NOBS + 2*NSCH + 22	NN + 5*NE + NSOP + NSOU + 2*NBCN + NOBS + 3*NSCH + 4	73*NOBS + 89*NSCH	2*NSCH
	iterative solver(s)	2*NELT + 28*NN + 19*NE + 3*NBCN + 6*NOBS + 2*NSCH + NWF + 220	NELT + 2*NN + 5*NE + NSOP + NSOU + 2*NBCN + NOBS + 3*NSCH + NWI + 2	73*NOBS + 89*NSCH	2*NSCH
3D	direct solver	(2*NBI + 27)*NN + 45*NE + 3*NBCN + 6*NOBS + 2*NSCH + 8	NN + 9*NE + NSOP + NSOU + 2*NBCN + NOBS + 3*NSCH + 4	73*NOBS + 89*NSCH	2*NSCH
	iterative solver(s)	2*NELT + 28*NN + 45*NE + 3*NBCN + 6*NOBS + 2*NSCH + NWF + 6	NELT + 2*NN + 9*NE + NSOP + NSOU + 2*NBCN + NOBS + 3*NSCH + NWI + 2	73*NOBS + 89*NSCH	2*NSCH

* For the present purpose, the effective dimension of a character array is defined as the array dimension multiplied by the length of the character variable. For example, an array of dimension 100 and type CHARACTER*40 has an effective dimension of 100*40=4,000.

The quantities in the table above are defined as follows:

NN	= number of nodes
NE	= number of elements
NBI	= full bandwidth of matrix (NBI is equal to one plus twice the maximum difference in node numbers in the element containing the largest node number difference in the mesh. See Figure 7.1)
NSOP	= number of fluid source nodes
NSOU	= number of solute or energy source nodes
NPBC	= number of specified pressure nodes
NUBC	= number of specified U nodes
NBCN	= NPBC + NUBC + 1
NOBS	= number of observation points
NSCH	= number of schedules
NN1	= number of nodes in the first node numbering direction
NN2	= number of nodes in the second node numbering direction
NELT	= length of matrix storage arrays for iterative solvers = 9*NN - 6*NN1 - 2 in 2D = 27*NN - 6*NN1*(3*NN2 + 1) - 2 in 3D
NWF	= length of floating-point workspace array for iterative solvers = NL + 5*NN + 1 for CG solver = NELT + 16*NN + 132 for GMRES solver = NELT + 39*NN + 11 for ORTHOMIN solver (if two different iterative solvers are used, the larger value applies)
NWI	= length of integer workspace array for iterative solvers = 2*NL + 11 for CG solver = 2*NELT + 31 for GMRES solver = 2*NELT + 11 for ORTHOMIN solver (if two different iterative solvers are used, the larger value applies)
NL	= (NELT + NN)/2 (defined for CG solver only)

Please see the main program listing in the SUTRA source code for the memory requirements of the most recent version of SUTRA. The memory requirements are calculated by SUTRA and reported in the ".lst" output file. Note that the expressions given in the table above include only the maximum memory used by dynamically allocated arrays and pointers, though these account for the vast majority of the memory usage. SUTRA uses a small amount of additional memory for the storage of various program variables.

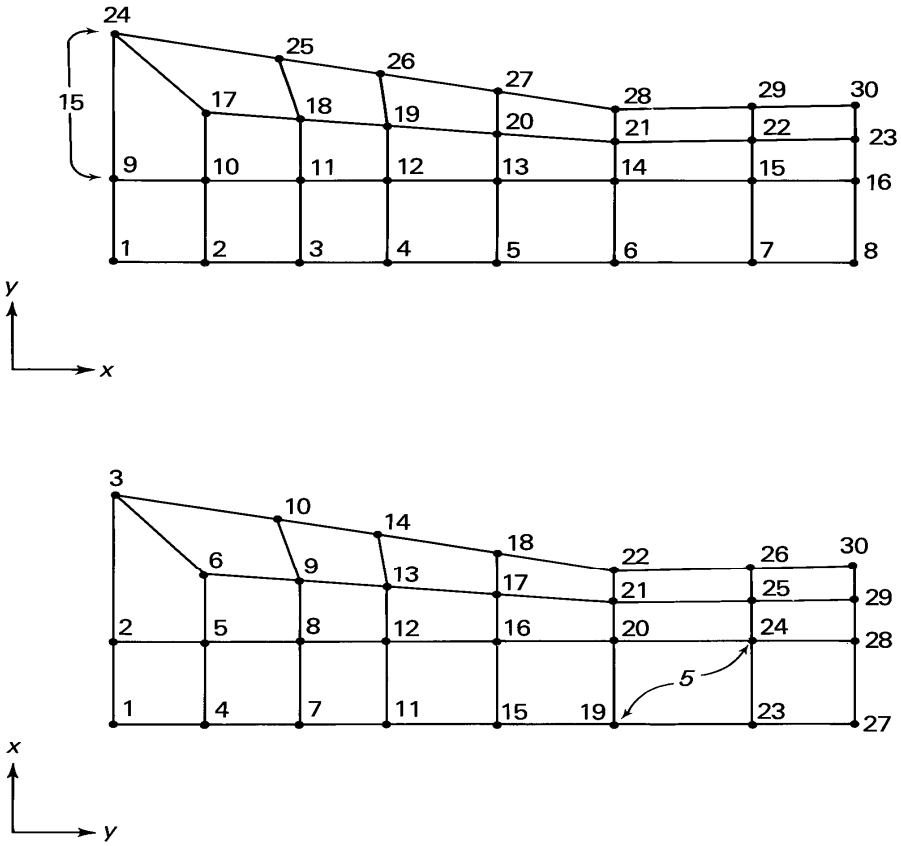


Figure 7.1. Minimization of bandwidth by careful numbering of nodes. In this 2D example, the same mesh has been numbered two different ways. In the first numbering scheme, the largest difference between node numbers in a single element is 15, giving a bandwidth of 1+2(15)=31. In the second numbering scheme, the largest difference between node numbers in a single element is 5, giving a bandwidth of 1+2(5)=11. The same principle applies to 3D meshes; the bandwidth equals one plus the maximum difference between nodes numbers in the element that contains the largest node number difference in the mesh.

7.4 Input and Output Files

SUTRA reads information from three input files and writes information to as many as seven types of output files. One of the input files, permanently assigned the name "**SUTRA.FIL**", contains the file name (and, optionally, the Fortran unit number assignments) for the remaining input and output files. (See Section 7.7 for details regarding the input format for the "**SUTRA.FIL**" file.) The other two input files and one of the output files must always be assigned by the user. For the remaining output files, assignment is either entirely optional or required only if certain simulation and output options are desired. In the list below, "filename" is a user-specified prefix for the input and output files to be used in a specific simulation. The **boldface** suffixes (such as **.inp**) are the recommended file type extensions for the input and output files.

INPUT FILES:

SUTRA.FIL

UNIT-K0

The UNIT-K0 file, "SUTRA.FIL", is a permanently assigned file that contains user determined file names and (optionally) unit number assignments for units *K1* – *K8* in each simulation. SUTRA sets *K0* = 10.

filename**.inp**

UNIT-K1

A file must be assigned as fortran-unit-*K1*. This "**.inp**" file contains all of the input data necessary for simulation except initial conditions.

filename**.ics**

UNIT-K2

A file must be assigned as fortran-unit-*K2*. This "**.ics**" file contains initial conditions of pressure and concentration or temperature for the simulation to be done**.**

OUTPUT FILES:

SUTRA.SMY or filename.**smy**

UNIT-K00

A file can be assigned as fortran-unit-*K00*. (If not assigned by the user, it defaults to file name "SUTRA.SMY".) This "**.smy**" file summarizes simulation progress, captures errors and convergence information.

filename**.lst**

UNIT-K3

A file must be assigned as fortran-unit-*K3*, in which the primary output of the simulation will be placed. This is the "**.lst**" file.

filename**.rst**

UNIT-K4

An *optional* output file must be assigned as fortran-unit-*K4* if the option to save the solution of the most recently completed time step for later restart is chosen in *UNIT-K1* **(**when ISTORE ≥ 1). Data will be placed in this "**.rst**" file in a format equivalent to *UNIT-K2* data so that this file may later be used as *UNIT-K2* ("**.ics**") initial conditions.

filename**.nod**

UNIT-K5

An *optional* output file must be assigned as fortran-unit-*K5* to save nodewise simulation results in a columnwise format at a time step interval specified by the user. This is the "**.nod**" file.

filename**.ele**

UNIT-K6

An *optional* output file must be assigned as fortran-unit-*K6* to save elementwise simulation results in a columnwise format at a time step interval specified by the user. This is the "**.ele**" file.

filename**.obs**

UNIT-K7

An *optional* output file must be assigned to save simulation results at observation points in a format that writes results time step by time step, with observation

points listed across the page. This is the "**.obs**" file. All observation results for which the OBS format and the same output schedule have been specified are written to the same "**.obs**" file. SUTRA automatically opens the required number of "**.obs**" files. The "**.obs**" file currently being written to is assigned to fortran-unit-*K7*.

filename**.obc**

UNIT-K8

An *optional* output file must be assigned to save simulation results at observation points in a format that writes results time step by time step, with observation points listed down the page. This is the "**.obc**" file. All observation results for which the OBC format and the same output schedule have been specified are written to the same "**.obc**" file. SUTRA automatically opens the required number of "**.obc**" files. The "**.obc**" file currently being written to is assigned to fortran-unit-*K8*.

The data lists and formats for the input files are given in Appendix B, "SUTRA Input Data List."

7.5 User-Supplied Programming

When SUTRA is used for simulation of systems with unsaturated flow, the user must code the desired unsaturated flow functions in subroutine UNSAT. When the SUTRA simulation includes time-dependent boundary conditions or sources, the desired temporal variations must be coded by the user in subroutine BCTIME.

Subroutine UNSAT

The general operation of this subroutine is described in section 5.9, "Program Structure." Given a single value of pressure, UNSAT must provide values of S_w, ($\partial S_w / \partial p$), and k_r. UNSAT consists of three sections. The user must supply code in each of these sections. An example using the unsaturated flow functions (2.8), (2.11), and (2.21a) and (2.21b) is given in subroutine UNSAT itself.

The first section requires specification of saturation, S_w, as a function of pressure, p. The second section requires specification of the derivative of saturation with respect to pressure, p, or saturation, S_w. The third section requires specification of the relative permeability, k_r, as a function of saturation, S_w, or pressure, p. The pressure value that is passed to UNSAT is the projected value, the most recent iterate, or the newly obtained solution. The values are either at Gauss points or at nodes.

Any convenient programming algorithm may be used to implement these functions in UNSAT. Some possibilities are use of explicit expressions, as in the example; use of data statements; use of logical statements to give piecewise continuous functions; or use of READ statements to input new data to the functions from either the ".inp" input file or a new data file. In some cases involving entry pressure or residual saturation, logical statements may be used to apply different functions for different ranges of S_w or p.

Subroutine BCTIME

The general operation of this subroutine is described in section 5.9, "Program Structure and Program Unit Descriptions." At the beginning of each time step, BCTIME must provide all specified time-varying pressure values and temperature or concentration values of fluid inflow at these nodes; values of specified time-varying temperature or concentration; values of specified time-varying fluid sources (or sinks) and temperatures or concentrations of these flows if they are inflows; and values of time-varying energy or solute mass sources (or sinks). BCTIME consists of four sections, each dealing with one of the above types of specification. The user must supply code in the section (or sections) of BCTIME that specifies the particular type of time-varying boundary condition or source desired.

The first section is used for specifying either time variation of pressure, or time variation of the temperature or concentration of any fluid that enters the system at a point of specified pressure, or both. The coding must be entered within a loop that checks all NPBC specified pressure nodes for the time-variability flag. This flag is a negative node number in the list of specified pressure nodes IPBC(IP). The counter for the list is IP. When the loop finds that the IP^{th} node number, IPBC(IP), is negative, then the actual node number is given by I = -IPBC(IP). In this case, the user must supply code that specifies a value appropriate for the current time step, for both PBC(IP), which is the specified pressure for the IP^{th} specified pressure node (node I), and for UBC(IP), which is the specified temperature or concentration of any inflow at the IP^{th} specified pressure node (node I). The loop skips over node numbers in the list IPBC(IP) that are positive.

The second section is used for specifying time variation of temperature or concentration. The coding must be entered within a loop that checks all NUBC specified temperature or concentration (U) nodes for the time-variability flag. This flag is a negative node number in the list of specified U nodes, IUBC(IU). The list begins in the $(NPBC + l)^{th}$ element of IUBC as shown in the description of subroutine BOUND in section 5.9, "Program Structure and Program Unit Descriptions." The first NPBC elements of IUBC are blank. The counter for the list is IU. If the loop finds that the IU^{th} node number, IUBC(NPBC + IU), is negative, then the actual node number is given by I = -IUBC(NPBC + IU). In this case, the user must supply code that specifies a value, appropriate for the current time step, for UBC(NPBC + IU), which is the specified temperature or concentration for the IU^{th} specified U node (node I). The loop skips over node numbers in the list IUBC(NPBC + IU) that are positive.

The third section is used for specifying time variation of either fluid sources (or sinks), temperature or concentration of inflowing fluid at sources, or both. The coding must be entered within a loop that checks all NSOP fluid source nodes for the time-variability flag. This flag is a negative node number in the list of fluid source nodes, IQSOP(IQP). The counter for the list is IQP. If the loop finds that the IQP^{th} node number IQSOP(IQP), is negative, then the actual node number is given by I = - IQSOP(IQP). In this case, the user must supply code that specifies a value appropriate for the current time step, for both QIN(I), which is the specified fluid source for node I (the IQP^{th} specified fluid source node), and for UIN(I), which is the temperature or concentration of inflowing fluid at node I. The loop skips over node numbers in the list IQSOP(IQP) that are positive.

The fourth section is used for specifying time variation of energy or solute mass sources. The coding must be entered within a loop that checks all NSOU specified energy or solute mass source nodes for the time-variability flag. This flag is a negative node number in the list of

specified energy or solute mass source nodes, IQSOU(IQU). The counter for the list is IQU. If the loop finds that the IQUth node number, IQSOU(IQU), is negative, then the actual node number is given by I = -IQSOU(IQU). In this case, the user must supply code that specifies a value appropriate for the current time step, for QUIN(I), which is the specified energy or solute mass source for node I (the IQUth specified energy or solute mass source node). The loop skips over node numbers in the list IQSOU(IQU) that are positive.

The current time at the end of the present time step in seconds, TSEC, and in other time units is available for use in specifying time variations. Any convenient programming algorithm may be used to implement the time-variations in BCTIME. Some possibilities are use of expressions as explicit functions of time such as, for example, a sine function to represent tidal pressure variations; use of data statements and new arrays explicitly dimensioned in BCTIME; use of logical statements to give stepped or piecewise continuous functions; or use of READ statements to input the time-varying values directly from the ".inp" input file or a new data file. If different functions or values are to be specified at various nodes, then the user must also supply code to distinguish which functions apply to which specified node numbers.

7.6 Modes and Options

Simulation modes

SUTRA may simulate flow and transport in three temporal modes for either energy or solute transport: (1) transient flow and transport, (2) steady flow with transient transport, and (3) steady flow and steady transport. Mode (1) is the most computationally expensive, and mode (3) is the least expensive. Modes (2) and (3) are not applicable to all problems. The classes of problems amenable to solution by each mode are given below.

(1) *Transient Flow* and *Transient Transport*

Allows for simulation of any physical problem that SUTRA deals with: either saturated or unsaturated flow or both; variable fluid density and viscosity; any sorption isotherm; energy or solute transport.

(2) *Steady-State Flow* and *Transient Transport*

Allows for simulation of a restricted class of SUTRA problems: saturated flow only; constant fluid density and viscosity; any sorption isotherm; energy transport with only small variations in temperature, or solute transport.

(3) *Steady-State Flow* and *Steady-State Transport*

Allows for simulation of the most restricted class of SUTRA problems: saturated flow only; constant fluid density and viscosity; linear sorption isotherm only; energy transport with only small variations in temperature, or solute transport.

These modes are specified in the ".inp" input data file by the values of CSSFLO, CSSTRA, and SIMULA.

".lst" file output options

To help the user interpret SUTRA simulation results, two options are available for output to the ".lst" file. These are (1) velocity output and (2) budget output.

(1) *Velocity Output*

An output of fluid velocity is available that may be used to plot velocity vectors with computer graphics software supplied by the user, or using SutraPlot (in 2D or 3D; Souza (1999)), SutraGUI (in 2D; Winston and Voss (2003)), or ModelViewer (in 2D or 3D; Hsieh and Winston, 2002). These velocities are calculated and output on each time step that a pressure solution is output. One velocity is calculated in each finite element, at the location of the element centroid, as described in section 5.7, "Velocity Calculation for Output." Velocity output occurs in groups of values: first, the magnitude of the velocity vector at each element centroid; then, each of the angles that describe the orientation of the velocity vector. In 2D, only one angle is reported; the angle measured counterclockwise from the +x-axis to the velocity vector. In 3D, an additional angle is reported; the angle measured vertically from the x,y-plane to the velocity vector. Velocity values are lagged one time step if a noniterative solution is used. (In this case, they are calculated not with the new pressure solution, but with the solution of the previous time step and with fluid density values of the step before that. This keeps the velocity calculations consistent in time.) This option is controlled by the ".inp" input file parameter CVEL. The user can choose to report the x-, y-, and z-components of velocities at element centroids in the optional ".ele" file.

(2) *Budget Output*

A fluid mass and energy or solute mass budget output is available as an aid in tracking the simulated behavior of a system. When the direct solver is used, the budget is not a rigorous check on numerical accuracy of the model, as the calculations involved in determining the budget are less accurate than the calculations used to carry out the SUTRA simulation. However, when the iterative solvers are used, the budget imbalances may be used to judge convergence of the iterative matrix equation solution. The budget is output on each time step with printed output to the ".lst" file, and tallies total system changes in fluid mass, and energy or solute mass for the time step. In addition to these totals of these quantities for the entire simulated region, the budget lists time step total gains and losses in these quantities at each specified pressure node, fluid source node, and energy or solute mass source node in the mesh. More information about the budget calculations is given in section 5.8, "Budget Calculations." This option is controlled by the ".inp" input file parameter CBUDG.

References

Bear, Jacob, 1979, Hydraulics of Groundwater: McGraw-Hill, New York, 567 p.

Desai, C.S., and Contractor, D.N., 1977, Finite element analysis of flow, diffusion, and salt water intrusion in porous media: *in* Formulation and Computational Algorithms in Finite Element Analysis, Bathe, K.J., and others (editor), MIT Press, p. 958-983.

Freeze, R.A., and Cherry, J.A., 1979, Groundwater: Prentice-Hall, Englewood Cliffs, NJ, 604 p.

Frind, E.O., 1982, Simulation of long-term transient density-dependent transport in groundwater: Advances in Water Resources, v. 5, p. 73-97.

Gelhar, L.W., and Axness, C.L., 1983, Three-dimensional stochastic analysis of macrodispersion in aquifers: Water Resources Research, v. 19, no. 1, p. 161-180.

Gelhar, L.W., and Collins, M.A., 1971, General analysis of longitudinal dispersion in nonuniform flow: Water Resources Research, v. 7, no. 6, p. 1511-1521.

Goode, D.J., 1992, Modeling transport in transient ground-water flow; an unacknowledged approximation: Ground Water, v.30, no.2, p.257-261.

Henry, H.R., 1964, Effects of dispersion on salt encroachment in coastal aquifers: in Sea Water in Coastal Aquifers: U.S. Geological Survey Water-Supply Paper 1613-C, p. C71-C84.

Hoopes, J.A., and Harleman, D.R.F., 1967, Dispersion in radial flow from a recharge well: Journal of Geophysical Research, v. 72, no. 14, p. 3595-3607.

Hsieh, P.A., and Winston, R.B., 2002, User's guide to ModelViewer, a program for three-dimensional visualization of ground-water model results: U.S. Geological Survey Open-File Report 02-106, 18 p. http://water.usgs.gov/nrp/gwsoftware/modelviewer/ModelViewer.html

Huyakorn, P.S., and Pinder, G.F., 1983, Computational Methods in Subsurface Flow: Academic Press, New York, 473 p.

Huyakorn, P., and Taylor, C., 1976, Finite element models for coupled groundwater flow and convective dispersion: *in* Finite Elements in Water Resources by Gray, W.G., Pinder, G. F., and Brebbia, C. A. (editors), Pentech Press, London, 1.131-1.151.

INTERA, 1979, Revision of the documentation for a model for calculating effects of liquid waste disposal in deep saline aquifers: U.S. Geological Survey Water Resources Investigations 79-96, 73 p.

Konikow, L.F., 1977, Modeling chloride movement in the alluvial aquifer at the Rocky Mountain Arsenal, Colorado: U.S. Geological Survey Water-Supply Paper 2044, 43 p.

Lohman, S.W., 1979, Ground Water Hydraulics: U.S. Geological Survey Professional Paper 708, 70 p.

Nesse, W.D., 1986, Introduction to Optical Mineralogy: Oxford University Press, New York, 325 p.

Pinder, G.F., and Cooper, H.H., Jr., 1970, A numerical technique for calculating the transient position of the saltwater front: Water Resources Research, v. 6, no. 3, p. 875-882.

Pinder, G.F., and Gray, W.G., 1977, Finite Element Simulation in Surface and Subsurface Hydrology: Academic Press, New York, 295 p.

Provost, A.M., 2002, SutraPrep, a pre-processor for SUTRA, a model for ground-water flow with solute or energy transport: U.S. Geological Survey Open-File Report 02-376, 43 p. http://water.usgs.gov/nrp/gwsoftware/sutra.html

Seager, M.K., 1989, A SLAP for the Masses: *in* Parallel Supercomputing: Methods, Algorithms and Applications, Carey, G. F. (editor): Wiley, p.135-155.

Segol, G., Pinder, G.F., Gray, W.G., 1975, A Galerkin-finite element technique for calculating the transient position of the saltwater front: Water Resources Research, v. 11, no. 2, p. 343-346.

Souza, W.R., 1999, SutraPlot, a graphical post-processor for SUTRA, a model for ground-water flow with solute or energy transport: U.S. Geological Survey Open-File Report 99-220, 30 p. http://water.usgs.gov/nrp/gwsoftware/sutra.html

Vandevender, W.H., and Haskell, K.H., 1982, The SLATEC mathematical subroutine library: SIGNUM Newsletter, v. 17, no. 3, p. 16-21.

Van Genuchten, M.Th., 1980, A closed-form equation for predicting the hydraulic conductivity of unsaturated soils: Soil Science Society of America Journal, v. 44, no. 5, p. 892-898.

Van Genuchten, M.Th., 1982, A comparison of numerical solutions of the one-dimensional unsaturated-saturated flow and mass transport equations: Advances in Water Resources, v. 5, no. 1, p. 47-55.

Voss, C.I., 1984, SUTRA – A finite-element simulation model for saturated-unsaturated, fluid-density-dependent ground-water flow with energy transport or chemically-reactive single-species solute transport: U.S. Geological Survey Water-Resources Investigations Report 84-4369, 409 p. http://water.usgs.gov/nrp/gwsoftware/sutra.html

Voss, C.I., 1999, USGS SUTRA code – History, practical use, and application in Hawaii: *in* Seawater Intrusion in Coastal Aquifers – Concepts, Methods and Practices, Bear, J., Cheng, A. H.-D., Sorek, S., Ouazar, D., and Herrera, I. (editors), Kluwer Academic Publishers, Boston, p. 249-313.

Voss, C.I., and Souza, W.R., 1987, Variable density flow and solute transport simulation of regional aquifers containing a narrow freshwater-saltwater transition zone: Water Resources Research, v. 23-10, p. 1851-1866.

Winston, R.B. and Voss, C.I., 2003, SutraGUI, a graphical-user interface for SUTRA, a model for ground-water flow with solute or energy transport: U.S. Geological Survey Open-File Report 03-285, 114 p. http://water.usgs.gov/nrp/gwsoftware/sutra.html

Wang, H.F., and Anderson, M.P., 1982, Introduction to Groundwater Modeling: Freeman and Co., San Francisco, 237 p.

Warrick, A.W., Biggar, J.W., and Nielsen, D.R., 1971, Simultaneous solute and water transfer for an unsaturated soil: Water Resources Research, v. 7, no. 5, p. 1216-1225.

APPENDICES

Appendix A: List of Symbols

Generic Units

[1]	dimensionless
[E]	energy units, or $[M \cdot L^2/s^2]$
[L]	length units
$[L_f^3]$	fluid volume
$[L_G^3]$	solid grain volume
[M]	fluid mass units
$[M_G]$	solid grain mass units
$[M_s]$	solute mass units

Units

[°C]	degrees Celsius
[cm]	centimeters
[d]	days
[h]	hours
[J]	Joules or $[kg \cdot m^2/s^2]$
[kg]	kilograms
[lbm]	pounds mass
[m]	meters
[min]	minutes
[mo]	months
[s]	seconds

Special Notation

$\frac{\partial \Psi}{\partial t}$ or $\frac{d\Psi}{dt}$	time derivative of Ψ
$\underline{v} = \underline{i}\, v_x + \underline{j}\, v_y + \underline{k}\, v_z$	vector $\underline{v}$ with components in $\underline{i}$, $\underline{j}$, and $\underline{k}$ directions
$\underline{\nabla}\psi = \underline{i}\frac{\partial \Psi}{\partial x} + \underline{j}\frac{\partial \Psi}{\partial y} + \underline{k}\frac{\partial \Psi}{\partial z}$	gradient of scalar Ψ
$\underline{\nabla} \bullet \underline{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$	divergence of vector $\underline{v}$
$i = \overline{1, NN} = 1,2,3,4,...., NN$	index i takes on all integer values between one and NN
$\lvert\Psi\rvert$	absolute value of scalar Ψ
$\lvert\underline{v}\rvert$	magnitude of vector $\underline{v}$
$\langle\langle\Psi\rangle\rangle$	approximate or discretized value of Ψ
$\Delta\Psi$	discrete change in value of Ψ (e.g.: $\Delta\Psi = \Psi_2 - \Psi_1$)
Ψ_o	initial condition or zeroth value of Ψ
Ψ_{BC}	value of Ψ as specified at a boundary condition node
Ψ_i or Ψ_j	value of Ψ at node or cell i or j
Ψ_{IN}	value of Ψ in inflow
Ψ_{KG}	value of Ψ at the KG^{th} Gauss point
Ψ_L	value of Ψ in element L
v_s	component of a vector $\underline{v}$ along a stream line
v_x	component of a vector $\underline{v}$ in the x direction
v_y	component of a vector $\underline{v}$ in the y direction

v_z	component of a vector $\underline{v}$ in the z direction
v_ξ	component of a vector $\underline{v}$ in the ξ direction
v_η	component of a vector $\underline{v}$ in the η direction
v_ζ	component of a vector $\underline{v}$ in the ζ direction
Ψ^L	value of Ψ in element L
Ψ^n	value of Ψ at time step n
Ψ^{n+1}	value of Ψ at time step n+l
$\Psi^{(n+1)*}$	value of Ψ evaluated at previous time step on first iteration, and at most recent iteration on subsequent iterations
Ψ^{proj}	value of Ψ projected from previous time steps on first iteration
$\langle\langle \underline{v} \rangle\rangle^*$	consistently evaluated velocity
$\langle\langle \rho \underline{g} \rangle\rangle^*$	consistently evaluated density-gravity term
$\sum_{i=1}^{NN} \psi = \psi_1 + \psi_2 + \psi_3 + \cdots + \psi_{NN}$	summation

Greek Lowercase

α	(2.17)	$[M/(L \cdot s^2)]^{-1}$	Porous matrix compressibility
$\alpha_L(x,y[,z],t)$	(2.40a) (2.42a)	[L]	Longitudinal dispersivity
$\alpha_{Lmax}(x,y[,z])$	(2.43a,b) (2.45a,b)	[L]	Longitudinal dispersivity in the maximum permeability direction
$\alpha_{Lmid}(x,y,z)$	(2.45a,b)	[L]	Longitudinal dispersivity in the middle permeability direction in 3D

$\alpha_{Lmin}(x,y[,z])$	(2.43a,b) (2.45a,b)	[L]	Longitudinal dispersivity in the minimum permeability direction
$\alpha_T(x,y,t)$	(2.40b) (2.42b)	[L]	Transverse dispersivity in 2D
$\alpha_{T1}(x,y[,z],t)$	(2.42b)	[L]	First transverse dispersivity in 3D
$\alpha_{T2}(x,y[,z],t)$	(2.42c)	[L]	Second transverse dispersivity in 3D
$\alpha_{Tmax}(x,y[,z])$	(2.44a,b) (2.46a,b)	[L]	Transverse dispersivity in the maximum permeability direction
$\alpha_{Tmid}(x,y,z)$	(2.46a,b)	[L]	Transverse dispersivity in the middle permeability direction in 3D
$\alpha_{Tmin}(x,y[,z])$	(2.44a,b) (2.46a,b)	[L]	Transverse dispersivity in the minimum permeability direction
β	(2.15)	$[M/(L\cdot s^2)]^{-1}$	Fluid compressibility
$\gamma_o^s(x,y[,z],t)$	(2.25)	$[E/M_G\cdot s]$	Energy source in solid grains
γ_o^s	(2.37b)	$[(M_s/M)/s]$	Zero-order adsorbate mass production rate
$\gamma_o^w(x,y[,z],t)$	(2.25)	$[E/M\cdot s]$	Energy source in fluid
γ_o^w	(2.37a)	$[(M_s/M)/s]$	Zero-order solute mass production rate
γ_1^s	(2.37b)	$[s^{-1}]$	First-order mass production rate of adsorbate
γ_1^w	(2.37a)	$[s^{-1}]$	First order mass production rate of solute
δ_{ij}	(4.112a)	[1]	Kronecker delta
$\varepsilon(x,y[,z],t)$	defined after (2.6)	[1]	Porosity
ζ	(4.29)	[L]	ζ local coordinate
η	(4.3),(4.27)	[L]	η local coordinate

$\theta(x,y)$	defined after (2.20b)	[°]	Angle from +x-coordinate axis to direction of maximum permeability in 2D
$\theta_1(x,y,z)$	defined after (2.20b)	[°]	Angle from +x-coordinate axis to direction of maximum permeability, measured within the x,y-plane in 3D
$\theta_2(x,y,z)$	defined after (2.20b)	[°]	Angle from x,y-plane to direction of maximum permeability, measured upward from the x,y-plane in 3D
$\theta_3(x,y,z)$	defined after (2.20b)	[°]	Angle from x,y-plane to direction of middle permeability measured within the plane perpendicular to the maximum permeability direction in 3D
$\theta_{kv}(x,y,t)$	(2.43a,b) (2.44a,b)	[°]	Angle from maximum permeability direction to local flow direction in 2D
$\theta_{kv1}(x,y,z,t)$	(2.45a,b)	[°]	Angle from maximum permeability direction to local flow direction, measured within (max,mid)-plane in 3D
$\theta_{kv2}(x,y,z,t)$	(2.45a,b)	[°]	Angle upward from the (max,mid)-plane to local flow direction in 3D
$\kappa_1(C,C_s)$	(2.32b)	$[M/M_G]$	First general sorption coefficient
$\kappa_2(C,C_s)$	(2.32b)	$[M/M_G \cdot s]$	Second general sorption coefficient
$\kappa_3(C,C_s)$	(2.32b)	$[M_s/M_G \cdot s]$	Third general sorption coefficient
$\lambda(x,y[,z],t)$	(2.25)	$[E/(s \cdot L \cdot °C)]$	Bulk thermal conductivity of solid matrix plus fluid
λ_s	(2.26)	$[E/(s \cdot L \cdot °C)]$	Solid thermal conductivity (about $\lambda_s \sim 3.5$ $[J/(s \cdot m \cdot °C)]$ at 20°C)
λ_w	(2.26)	$[E/(s \cdot L \cdot °C)]$	Fluid thermal conductivity (about $\lambda_w \sim 0.6$ $[J/(s \cdot m \cdot °C)]$ at 20°C)
μ	(2.5), (2.6)	$[M/(L \cdot s)]$	Fluid viscosity

ν_{p_i}	(4.93)	[L•s]	Conductance for specified pressure in cell i
ν_p	(4.83)	[s/L^2]	Conductance for specified pressure nodes
ν_{U_i}	(4.143)	[E/(s•°C)] or [s^{-1}]	Conductance for specified temperature or concentration in cell i
ξ	(4.1),(4.25)	[L]	ξ local coordinate
ρ_o	(2.3),(2.4)	[M / L$_f^3$]	Base fluid density at C=C$_o$ or T=T$_o$
ρ(x,y[,z],t)	(2.1)	[M / L$_f^3$]	Fluid density
ρ_s	defined after (2.24), (2.30)	[M$_G$ / L$_G^3$]	Density of solid grains in solid matrix
ρ_w	defined after (2.2)	[M / L$_f^3$]	Density of pure water
σ'	(2.17)	[M/(L•s^2)]	Intergranular stress
σ_s	(2.47)	[L^2/s]	Diffusivity in solid phase in unified transport equation
σ_w	(2.47)	[L^2/s]	Diffusivity in fluid phase in unified transport equation
ϕ_j	(3.4)	[1]	Symmetric bilinear basis function in global coordinates at node j
χ_1	(2.34a,b)	[L$_f^3$ / M$_G$]	Linear distribution coefficient
χ_1	(2.35a,b)	[L$_f^3$ / M$_G$]	A Freundlich distribution coefficient
χ_1	(2.36a,b)	[L$_f^3$ / M$_G$]	A Langmuir distribution coefficient
χ_2	(2.35a,b)	[1]	Freundlich coefficient
χ_2	(2.36a,b)	[L$_f^3$ / M$_s$]	Langmuir coefficient

ψ_{BC_i}	(4.143)	[E/s] or [M_s/M•s]	Source of energy or solute mass at specified temperature or concentration node
ψ_{IN_i}	(4.144) and defined after (4.122)	[E/s] or [M_s/M•s]	Source of energy or solute mass at node i
ψ_{OUT_i}	defined after (4.122)	[E/s] or [M_s/M•s]	Sink of energy or solute mass at node i
ω_i	(4.88)	[1]	Asymmetric weighting function in global coordinates at node i

Greek Uppercase

Γ	(3.17)	[L^2]	External boundary (area) of simulated region
$\Gamma_s(x,y[,z],t)$	(2.30)	[M_s/M_G•s]	Adsorbate mass source (per unit solid matrix mass) due to production reactions within adsorbed material itself
$\Gamma_w(x,y[,z],t)$	(2.29)	[M_s/M•s]	Solute mass source in fluid (per unit fluid mass) due to production reactions
ΔL_L	(7.4)	[L]	Distance between sides of element L along streamline
ΔL_T	(7.5)	[L]	Distance between sides of element L perpendicular to streamline
Δt_n	(3.33)	[s]	Time step n
Δt_{n+1}	(3.29)	[s]	Time step n+1
H_+	(4.4),(4.28)	[1]	One-dimensional basis function in η direction
H_-	(4.3),(4.27)	[1]	One-dimensional basis function in η direction
H^*	(4.18),(4.56)	[1]	Asymmetric portion of η weighting function
θ_i	(4.13)-(4.16) (4.47)-(4.54)	[1]	Asymmetric weighting function at node i

Ξ_+	(4.2),(4.26)	[1]	One-dimensional basis function in ξ direction
Ξ_-	(4.1),(4.25)	[1]	One-dimensional basis function in ξ direction
Ξ^*	(4.17),(4.55)	[1]	Asymmetric portion of ξ weighting function
$\Upsilon(x,y[,z],t)$	(2.22)	$[M_s/(L^3 \cdot s)]$	Solute mass source (e.g., dissolution of solid matrix or desorption)
Ψ_+	(4.30)	[1]	One-dimensional basis function in ζ direction
Ψ_-	(4.29)	[1]	One-dimensional basis function in ζ direction
Ψ^*	(4.57)	[1]	Asymmetric portion of ζ weighting function
Ω_i	(4.5)-(4.8) (4.31)-(4.38)	[1]	Bilinear symmetric basis function at node i

Roman Lowercase

a_ξ, a_η, a_ζ	(4.23),(4.24) (4.66)-(4.68)	[1]	Asymmetric weighting function coefficients
$c(x,y[,z],t)$	(2.1)	$[M_s / L_f^3]$	Solute volumetric concentration (mass solute per volume total fluid)
c_s	(2.27b)	$[E/(M_G \cdot °C)]$	Solid grain specific heat (about $c_s \sim 8.4 \times 10^2$ [J/kg•°C] for sandstone at 20°C)
c_w	(2.25)	$[E/(M \cdot °C)]$	Specific heat of water (about $c_w \sim 4.182 \times 10^3$ [J/kg•°C] at 20°C)
$d_L(x,y[,z],t)$	(2.39a-g) (2.41a-g)	$[L^2/s]$	Longitudinal dispersion coefficient
$d_T(x,y,t)$	(2.39a-g) (2.41a-c)	$[L^2/s]$	Transverse dispersion coefficient in 2D

$d_{T1}(x,y,z,t)$	(2.41d-g)	$[L^2/s]$	First transverse dispersion coefficient in 3D
$d_{T2}(x,y,z,t)$	(2.41d-g)	$[L^2/s]$	Second transverse dispersion coefficient in 3D
det J	(4.73b),(4.74)	[1]	Determinant of Jacobian matrix
e_s	defined after (2.24)	$[E/M_G]$	Energy per unit mass solid matrix
e_w	defined after (2.24)	[E/M]	Energy per unit mass water
f(x,y[,z],t)	(2.30)	$[M_s/(L^3 \cdot s)]$	Volumetric adsorbate source (gain of adsorbed species by transfer from fluid per unit from fluid per unit total volume)
$f_s(x,y[,z],t)$	(2.32a)	$[M_s/M_G \cdot s]$	Specific solute mass adsorption rate (per unit mass solid matrix)
$\underline{g}$	(2.19a,b)	$[L/s^2]$	Gravitational acceleration (gravity vector)
h(x,y[,z],t)	(2.20) (3.1)	[L]	Hydraulic head (sum of pressure head and elevation head)
$\underline{\underline{k}}(x,y[,z])$	(2.19a)	$[L^2]$	Solid matrix permeability
$k_{max}(x,y[,z])$	defined after (2.20b)	$[L^2]$	Maximum value of permeability
$k_{mid}(x,y,z)$	defined after (2.20b)	$[L^2]$	Middle value of permeability in 3D
$k_{min}(x,y[,z])$	defined after (2.20b)	$[L^2]$	Minimum value of permeability
$k_r(x,y[,z],t)$	(2.19a)	[1]	Relative permeability to fluid flow (assumed to be independent of direction).
p(x,y[,z],t)	defined before (2.1)	$[M/(L \cdot s^2)]$	Fluid pressure
$p_c(x,y[,z],t)$	(2.7)	$[M/(L \cdot s^2)]$	Capillary pressure
p_{cent}	defined after (2.7)	$[M/(L \cdot s^2)]$	Entry capillary pressure

p_{BC_i}	(4.83)	[M/(L·s^2)]	Specified pressure value at node i
q_{IN_i}	defined after (4.89)	[M/s]	Fluid mass flux in across boundary at node i
q_{OUT_i}	(4.89)	[M/s]	Fluid mass flux out across boundary at node i
r^*	(6.3a)	[L]	Parameter in analytical solution for radial transport
s^*	(6.1a)	[L]	Drawdown for pump test example
s_L	(4.131)	[1]	Left side coefficient contribution of sorption isotherm to U equation
s_R	(4.131)	[M_s/M]	Right side contribution of sorption isotherm to U equation
t	(3.4)	[s]	Time
v(x,y[,z],t)	(2.39)	[L/s]	Magnitude of velocity $\underline{v}$
$\underline{v}$(x,y[,z],t)	(2.19a)	[L/s]	Average fluid velocity
$\underline{v}_s$	(2.49)	[L/s]	Net solid matrix velocity
v_x(x,y[,z],t)	(2.39)	[L/s]	Magnitude of x-component of $\underline{v}$
v_y(x,y[,z],t)	(2.39)	[L/s]	Magnitude of y-component of $\underline{v}$
v_z(x,y,z,t)	(2.39)	[L/s]	Magnitude of z-component of $\underline{v}$ in 3D
x		[L]	x coordinate
x_{max}	defined after (2.20b)	[L]	Coordinate along direction of maximum permeability
x_{mid}	defined after (2.20b)	[L]	Coordinate along direction of middle permeability direction in 3D
x_{min}	defined after (2.20b)	[L]	Coordinate along direction of minimum permeability direction in 3D
y		[L]	y coordinate
z		[L]	z coordinate in 3D

Roman Uppercase

A	(6.3b)	$[L^2/s]$	Factor in analytical solution for radial transport
AF_i	(4.97)	$[L \cdot s^2]$	Matrix coefficient of pressure time derivative
AT_i	(4.133)	[E/°C] or [1]	Matrix coefficient of U time derivative
B(x,y,t)	(3.2)	[L]	Aquifer thickness in 2D mesh
BASE(x,y)	(3.2)	[L]	Elevation of aquifer base for example problem
BF_{ij}	(4.99) (4.101)	$[L \cdot s]$	Matrix coefficient in pressure equation
BT_{ij}	(4.135) (4.139)	$[E/(s \cdot °C)]$ or $[s^{-1}]$	Matrix coefficient in U equation
C_o	(2.4)	$[M_s/M]$	Base solute concentration in fluid
C(x,y[,z],t)	(2.1)	$[M_s/M]$	Solute mass fraction (or solute concentration) in fluid (mass solute per mass total fluid)
C_s(x,y[,z],t)	(2.30)	$[M_s/M_G]$	Specific concentration of adsorbate on solid grains (mass adsorbate/(mass solid grains plus adsorbate))
C^*(x,y[,z],t)	(2.29)	$[M_s/M]$	Solute concentration of fluid sources (mass fraction))
CF_i	(4.98)	[M/°C] or [M]	Matrix coefficient of U time derivative in pressure equation
$\underline{\underline{D}}$(x,y[,z],t)	(2.25),(2.29)	$[L^2/s]$	Dispersion tensor
D_m	(2.29)	$[L^2/s]$	Apparent molecular diffusivity of solute in solution in a porous medium including tortuosity effects, (D_m~1. x10^{-9} $[m^2/s]$ for NaCl at 20.°C)

D_{ij}	(2.39c) (2.41c)	$[L^2/s]$	Element of dispersion tensor
D_{xx}, D_{xy}, D_{xz}, D_{yx}, D_{yy}, D_{yz}, D_{zx}, D_{zy}, D_{zz}	(2.38a,b)	$[L^2/s]$	Elements of dispersion tensor in (x,y,z) coordinates
DF_i	(4.100) (4.102)	[M/s]	Element of vector on right side of pressure equation
DT_{ij}	(4.134) (4.138)	$[E/(s\cdot°C)]$ or $[s^{-1}]$	Matrix coefficient of U equation
ET_i	(4.137)	[E/s] or $[M_s/M\cdot s]$	Element of vector on right side of U equation
G_{KG}	(4.77)	[1]	Coefficient of Gauss integration
G_sTL	(4.136b)	$[E/(s\cdot°C)]$ or $[s^{-1}]$	Element of vector on left side of U equation
G_sTR	(4.136c)	[E/s] or $[M_s/M\cdot s]$	Element of vector on right side of U equation
GT_i	(4.136a)	$[E/(s\cdot°C)]$ or $[s^{-1}]$	Element of vector on left side of U equation
$\underline{\underline{I}}$	(2.25),(2.29)	[1]	Identity tensor (ones on diagonal, zeroes elsewhere)
I_{ij}	(3.23)	$[L^2/s]$	Matrix arising from integral in example problem
K(x,y[,z])	(2.20),(3.1)	[L/s]	Hydraulic conductivity
KG	(4.77)		Gauss point number
NE	(3.3)		Number of elements in mesh
NELT	§7.3		Length of matrix storage arrays for iterative solvers
NN	(3.4)		Number of nodes in mesh
NP	(4.77)		Number of Gauss points
NPBC	§7.1		Number of specified pressure nodes in mesh

NSOP	§7.1		Number of specified fluid source nodes in mesh
NSOU	§7.1		Number of specified U source nodes in mesh
NUBC	§7.1		Number of specified U nodes in mesh
NPCYC	§7.1		Pressure solution cycle
NUCYC	§7.1		U solution cycle
NSCH	§7.1		Number of schedules
NWF	§7.3		Length of floating-point workspace array for iterative solver
NWI	§7.3		Length of integer workspace array for iterative solver
O	(3.7)	$[s^{-1}]$	Fluid mass balance expression for the example problem
O_p	(4.83)	$[M/(L^3 \cdot s)]$	Fluid mass balance expression
O_u	(4.113)	$[E/(L^3 \cdot s)]$ $[M_s/(L^3 \cdot s)]$	Energy or solute mass balance expression
Pe_m	(7.1)	[1]	Mesh Peclet number
PBC_{ipu}	§7.1	$[M/(L \cdot s^2)]$	The ipu^{th} pressure boundary condition value
Q_i	(4.94)	[M/s]	Total fluid mass source to cell i
$Q_p(x,y[,z],t)$	(2.22)	$[M/(L^3 \cdot s)]$	Fluid mass source (including pure water mass plus solute mass dissolved in source water)
$Q^*(x,y[,z])$	(3.1)	$[s^{-1}]$	Volumetric fluid source for example problem (volume fluid injected per time / volume aquifer)
Q_{PBC}	(4.95)	$[M/(L^3 \cdot s)]$	Fluid mass source rate due to a specified pressure

Q_{BC_i}	(3.38)	$[L^3/s]$	Fluid volumetric source due to a specified head in the example problem
Q_{BC_i}	(4.111)	[M/s]	Fluid mass source due to a specified pressure node
Q_{IN_i}	(3.20)	$[L^3/s]$	Fluid volume efflux at boundary for example problem
Q_{TOT}	(6.1a)	[M/s]	Total pumping rate for pump-test example
Q_i^*	(3.28)	$[L^3/s]$	Fluid volumetric source for example problem
R	(3.8)	$[s^{-1}]$	Residual of discretized equation
$S_{op}(x,y[,z])$	(2.13)	$[M_f/(L{\cdot}s^2)]^{-1}$	Specific pressure storativity
$S_o(x,y)$	(3.1)	$[L^{-1}]$	Specific storativity for example problem
$S_w(x,y[,z],t)$	defined after (2.6)	[1]	Water saturation (saturation) (volume of water per volume of voids)
T_o	(2.3)	[°C]	Base fluid temperature
T(x,y[,z],t)	defined before (2.1)	[°C]	Fluid temperature (degrees Celsius)
T(x,y,t)	(3.2)	$[L^2/s]$	Aquifer transmissivity for example problem
$T^*(x,y[,z],t)$	(2.25)	[°C]	Temperature of source fluid
U	(2.47)	[°C] or $[M_s/M]$	Either T or C depending on type of simulation
$\underline{U}$	(2.41)	[1]	Unit eigenvector of the dispersion tensor, perpendicular to the flow direction
U_{BC}	(4.113)	[°C] or $[M_s/M]$	U value of inflow at point of specified pressure

U^*	(2.47a)	[°C] or $[M_s/M]$	U value of fluid source
UP	(4.23),(4.66)	[1]	Upstream weighting factor
$\underline{V}$	(2.41)	[1]	Unit eigenvector of the dispersion tensor, aligned with the flow direction
V_i	(3.15)	$[L^3]$	Cell volume at node i
VOL	(2.9)	$[L^3]$	Volume (total)
VOL_w	(2.13)	$[L_f^3]$	Fluid volume
$\underline{W}$	(2.41)	[1]	Unit eigenvector of the 3D dispersion tensor, perpendicular to the flow direction
W_o	(4.161b)	[1]	Weight for Langmuir isotherm
W_∞	(4.161a)	[1]	Weight for Langmuir isotherm
W_i	(4.84)	[1]	Weighting function
W(u)	(6.1a)	[1]	Well function for pump test example

Appendix B: SUTRA Input Data List

List of Input Data for the File Assignment Input File (**SUTRA.FIL**)

Model Version: ***SUTRA 2.1***

The file "SUTRA.FIL" contains file assignments (one line for each assignment) in the following format:

Variable	Type	Description
FTYPE	Character	File type (within single quotes ' ' as shown below). Valid values are as follows: 'INP' = "**.inp**" input file (main **inp**ut) 'ICS' = "**.ics**" input file (**i**nitial **c**onditions) 'LST' = "**.lst**" output file (main output **lst**ing) 'RST' = "**.rst**" output file (**rst**art conditions) 'NOD' = "**.nod**" output file (**nod**ewise results) 'ELE' = "**.ele**" output file (**ele**mentwise results) 'OBS' = "**.obs**" output file (**obs**ervations) 'OBC' = "**.obc**" output file (**ob**servations) 'SMY' = "**.smy**" output file (simulation **s**u**m**mar**y**)
IUNIT	Integer	FORTRAN unit number to be assigned to the file. If IUNIT is not a valid FORTRAN unit number or if it is already assigned to another file, SUTRA will assign the next available unit number after IUNIT. (Unit numbers less than 11 are assumed to be unavailable.)
FNAME	Character	Full name of the file (within single quotes ' ' as shown below).

Notes:

Assignments for the "**.nod**", "**.ele**", and "**.obs**", and "**.obc**" files are optional. If any of these assignments are omitted, the corresponding output files will not be created by SUTRA. Assignment for the "**.smy**" file is also optional – if not assigned, it will receive the file name "**SUTRA.SMY**" and an automatically generated unit number, by default. Assignment for the "**.rst**" file is required if ISTORE ≠ 0 in dataset 4. Assignments for the "**.inp**", "**.ics**", and "**.lst**" files are always required. Assignments may be listed in any order. Assignment of unit numbers is performed in the order in which the files are listed, except for the OBS and OBC files. The latter are always assigned last.

For the observation output files (OBS and OBC) FNAME is a base filename from which the actual filenames are automatically derived by SUTRA. SUTRA generates one observation output file for each combination of schedule and output format that appears in the observation specifications in dataset 8D of the main input (INP) file.

Example:

'INP'	50	'project.inp'
'ICS'	55	'project.ics'
'LST'	60	'project.lst'
'RST'	66	'project.rst'
'NOD'	70	'project.nod'
'ELE'	80	'project.ele'
'OBS'	90	'project.obs'
'OBC'	92	'project.obc'
'SMY'	95	'project.smy'

General Format of the "**.inp**" and "**.ics**" Input Files

SUTRA reads the "**.inp**" and "**.ics**" input files in a list-directed fashion (except for dataset 1 of the "**.inp**" file):

- Input data appearing on the same line should be space- or tab-separated.

- As a rule, any data that are not optional must be given values in the input file (blanks are not sufficient) and must appear within the first 1,000 characters of a line. SUTRA reads only the first 1,000 characters of each line; subsequent characters are ignored. The except to this rule is dataset 8D, in which lists of times or time steps may extend beyond the 1,000-character limit.

- Enclose input variables of "character" type in single quotation marks (unless specified otherwise) to provide maximum compatibility across computing platforms.

- Comment lines may be placed within the "**.inp**" and "**.ics**" files, subject to the following restrictions:
  - Comment lines must either
    - be empty (i.e., contain only a carriage return), or
    - have a pound sign, #, in the first column.
  - Comment lines can be placed before or after any dataset.
  - Comment lines can be placed within any dataset except those in which a single line of data can be optionally broken up over multiple lines, namely, time or time step lists in dataset 8D of the main input (INP) file, datasets 2 and 3 of the initial conditions (ICS) file, and the "restart" information that follows dataset 3 in a "**.rst**" (restart) file being used as a "**.ics**" (initial conditions) file.
  - Comment lines may not be placed within any of the "restart" information that follows DATASET 3 in a "**.rst**" (restart) file being used as a "**.ics**" (initial conditions) file.

- Comments (or any text) can be appended to the end of any line of input data, provided all the required parameters have first been entered on that line. (In the case of a line of input that is optionally continued over multiple lines, only the last line would meet this requirement.) Be sure to leave at least one space or tab between the last required parameter and the beginning of the comment.

- Data contained in separate files can be "inserted" into the main (INP) and initial conditions (ICS) input files using the "@INSERT" command.

  - For example, including the line

    ```
    @INSERT  52  'project.inp7'
    ```

    in the INP file causes SUTRA to open file 'project.inp7' on FORTRAN unit 52 (or the next available unit number, according to the convention for files listed in "SUTRA.FIL") and begin reading input data from it as though it were reading

from the INP file. When the end of the inserted file is reached, SUTRA closes it and resumes reading from the INP file.

- "Inserts" can be nested, i.e., a file that contains an "@INSERT" statement can itself be "inserted" into another file. Nesting can be up to twenty levels deep.
- Like comment lines, "inserts" can be placed within any dataset except time or time step lists in dataset 8D of the main input (INP) file, dataset 2 or 3 of the initial conditions (ICS) file, or the "restart" information that follows dataset 3 in a "**.rst**" (restart) file being used as a "**.ics**" (initial conditions) file.

List of Input Data for the Main Input File (**.inp**)

Model Version: ***SUTRA 2.1***

DATASET 1: Output Heading (two lines)

Variable	Type	Description
TITLE1	Character	First line of heading for the input data set.
TITLE2	Character	Second line of heading for the input data set. The first 80 characters of each line are printed as a heading on SUTRA output. In this dataset, the character inputs need not be enclosed in quotation marks.

DATASET 2A: Simulation Type (one line)

Variable	Type	Description
SIMULA	Character	Four* words. The first word must be "SUTRA". The second and third words indicate the SUTRA version number and must be either "VERSION 2.1", "VERSION 2.0", or "VERSION 2D3D.1". The fourth word indicates the type of transport, and must be either "ENERGY" or "SOLUTE". Any additional words are ignored by SUTRA. * If the version specification is omitted, SUTRA will read all input datasets in the Version 2.0 (2D3D.1) format.

Examples:

For energy-transport simulation, one may write the following:
'SUTRA VERSION 2.1 ENERGY TRANSPORT'

For solute-transport simulation, one may write the following:
'SUTRA VERSION 2.1 SOLUTE TRANSPORT'

In these examples, the word "TRANSPORT" is ignored by SUTRA but is included to make the input more readable.

DATASET 2B: Mesh Structure (four lines)

This information is input for convenience of post-processing only. Except for the difference between 2D and 3D, calculations in the SUTRA code itself are not influenced by the mesh structure. Calculations in SUTRA for all mesh structures are handled as though the mesh were fully irregular.

Variable	Type	Description
Line 1:		
MSHSTR	Character	Two words. The first word indicates the dimensionality of the mesh, and must be either "2D" or "3D". The second word indicates the regularity of the mesh, and must be either "REGULAR", "BLOCKWISE", "LAYERED", or "IRREGULAR". Any additional words are ignored by SUTRA. By definition, a LAYERED mesh must be 3D. (See note at the end of this dataset for descriptions of the four types of mesh.)
For a 'REGULAR' mesh:		
NN1	Integer	Number of nodes in the first numbering direction. Must have NN1≥2.
NN2	Integer	Number of nodes in the second numbering direction. Must have NN2≥2.
NN3	Integer	For a 3D mesh, the number of nodes in the third numbering direction. Must have NN3≥2. May be omitted if the mesh is 2D.
For a 'LAYERED' mesh:		
NLAYS	Integer	Number of layers of nodes in the mesh. Must have NLAYS≥2.
NNLAY	Integer	Number of nodes in a layer. Must have NNLAY≥4.
NELAY	Integer	Number of elements in a layer. Must have NELAY≥1. In this context, the "number of elements in a layer" is the number of quadrilateral element faces defined by the nodes in a layer. One other words, if a layer is viewed as a 2D SUTRA mesh, then NELAY is the number of "2D elements" in a layer.

LAYSTR	Character	One word. Must be either "ACROSS" or "WITHIN". Indicates whether node numbering proceeds first across layers or within a layer.

For an 'IRREGULAR' mesh:

(No additional information required on Line 1.)

Omit lines 2 – 4 if mesh is NOT 'BLOCKWISE'.

Line 2:

NBLK1	Integer	Number of blocks in the first numbering direction.
LDIV1	Integer	A list of the number of elements into which to divide each of the NBLK1 blocks along the first numbering direction.

Line 3:

NBLK2	Integer	Number of blocks in the second numbering direction.
LDIV2	Integer	A list of the number of elements into which to divide each of the NBLK2 blocks along the second numbering direction.

Line 4:

NBLK3	Integer	Number of blocks in the third numbering direction.
LDIV3	Integer	A list of the number of elements into which to divide each of the NBLK3 blocks along the third numbering direction.

Notes:

A REGULAR mesh is a logically rectangular 2D or 3D mesh. Node numbering starts at a node at one of the eight "corners" of the mesh and proceeds in a "natural" order along rows, columns, and vertical strings of nodes. (Because the logically rectangular mesh can be geometrically deformed, "vertical" strings of nodes need not lie strictly along the Z-direction, and rows and columns need not lie strictly within an X-Y plane. As used here, "vertical" implies a numbering direction that is effectively perpendicular to the "horizontal" directions defined by the rows and columns.) For example, numbering might proceed first along a vertical string of nodes, continue with successive vertical strings until all nodes within the first row are numbered, and then continue row-by-row until all nodes are numbered. In this case, the nodes would be said to be numbered first along vertical strings (across layers), then along rows, and finally along columns.

A BLOCKWISE mesh is a special type of REGULAR mesh that is created by the preprocessor SutraPrep (Provost, 2002).

A LAYERED mesh is a 3D mesh that can be thought of as being formed from a vertical stack of 2D meshes. Each 2D mesh in the stack has the same connectivity, though that connectivity need not be logically rectangular. Node numbering starts at a node that lies in either the top or the bottom layer of the stack and proceeds in either of two ways:

- across the layers, i.e., along the vertical string of nodes until the opposite (bottom or top) layer is reached, or
- within a layer, i.e., along nodes in the same layer as the starting node.

Once numbering within a vertical string or layer is complete, it continues with another string or the next layer above or below. The order in which nodes are numbered within each string or layer is analogous to that in the first string or layer.

An IRREGULAR mesh is a 2D or 3D mesh that lacks the special structural features possessed by the other three mesh types. Nodes may be numbered in any order.

SUTRA uses the information in dataset 2B as follows:

- It checks that the mesh dimensions specified in datasets 2B and 3 are valid and mutually consistent.
- It writes mesh structure information and dimensions to output files, when they can be read by postprocessing software.

If the structure of the mesh is not known or is not of interest for postprocessing, the user may simply specify the mesh to be IRREGULAR, regardless of the actual structure. This will affect neither the way in which SUTRA solves the numerical problem nor the solution that is obtained.

Examples:

For a 3D, regular (logically rectangular), 10x20x30-node mesh, one may write the following:

```
'3D REGULAR MESH'  10  20  30
```

For a layered mesh with 10 layers of nodes, each containing 2560 nodes and 2210 elements, and with node numbering proceeding first across the layers, one may write:

```
'3D LAYERED MESH'  10  2560  2210  'ACROSS'
```

For a 2D, irregular mesh, one may write the following:

```
'2D IRREGULAR MESH'
```

In these examples, the word "MESH" is ignored by SUTRA but is included to make the input more readable.

DATASET 3: Simulation Control Numbers (one line)

Variable	Type	Description
NN	Integer	Exact number of nodes in finite element mesh.
NE	Integer	Exact number of elements in finite element mesh.
NPBC	Integer	Exact number of nodes at which pressure is a specified constant value or function of time.
NUBC	Integer	Exact number of nodes at which temperature or concentration is a specified constant value or function of time.
NSOP	Integer	Exact number of nodes at which a fluid source/sink is a specified constant value or function of time.
NSOU	Integer	Exact number of nodes at which an energy or solute mass source/sink is a specified constant value or function of time.
NOBS	Integer	Exact number of points at which observations will be made. Set to zero for no observations.

DATASET 4: Simulation Mode Options (one line)

Variable	Type	Description
CUNSAT	Character	One word. Set to 'SATURATED' to simulate only saturated flow. Set to 'UNSATURATED' to simulate unsaturated/saturated ground-water flow. (Note: When UNSATURATED flow is allowed, the unsaturated flow functions must be programmed by the user in subroutine UNSAT.)
CSSFLO	Character	One word. Set to 'TRANSIENT' for simulation of transient ground-water flow. Set to 'STEADY' for simulation of steady-state flow. (Note: Variable-density simulations generally require TRANSIENT flow.)
CSSTRA	Character	One word. Set to 'TRANSIENT' for simulation of transient solute or energy transport. Set to 'STEADY' for simulation of steady-state transport. (Note: Steady-state transport requires a steady-state flow field. So, if CSSTRA = 'STEADY', then also set CSSFLO = 'STEADY'.)
CREAD	Character	One word. Set to 'COLD' to read initial condition data ("**.ics**" file) for a "cold start" (the very first time step of a simulation). Set to 'WARM' to read initial condition data ("**.ics**" file) for a "warm restart" of a simulation using data that were previously stored by SUTRA in a "**.rst**" file. A "warm restart" is used only when continuing a previous simulation as though it had never been interrupted and with no changes in problem specification (except for changing time step size and extending simulation time).
ISTORE	Integer	To store results each ISTORE time steps in the "**.rst**" file for later use as initial conditions on a restart, set to +1 or greater value. To cancel storage, set to 0. This option is recommended as a backup for storage of results of intermediate time steps during long simulations. Should the execution halt unexpectedly, it may be restarted with initial conditions consisting of results of the last successfully completed time step stored in the "**.rst**" file. When ISTORE > 0, results are always stored in the "**.rst**" file after the last time step of a simulation regardless of whether the step is an even multiple of ISTORE. Any extra words included in the character variables in this dataset are ignored by SUTRA.

Example:

To simulate saturated, steady-state ground-water flow with transient solute or energy transport from a cold start, storing intermediate results every 10 time steps, one may write the following:

```
'SATURATED FLOW'   'STEADY FLOW'   'TRANSIENT TRANSPORT'  'COLD START'   10
```

In this example, the words “FLOW”, “TRANSPORT”, and “START” are ignored by SUTRA but may be included to make the input more readable.

DATASET 5: Numerical Control Parameters (one line)

Variable	Type	Description
UP	Real	Fractional upstream weight for stabilization of oscillations in results due to highly advective transport or unsaturated flow. UP may be given any value from 0.0 to +1.0. UP=0.0 implies no upstream weighting (Galerkin method). UP=0.5 implies 50% upstream weighting, UP=1.0 implies full (100%) upstream weighting. Recommended value is zero. WARNING: Upstream weighting increases the local effective longitudinal dispersivity of the simulation by approximately UP* (ΔL_L)/2 where ΔL_L is the local distance between element sides along the direction of flow (see section 7.2). Note that the amount of this increase varies from place to place depending on flow direction and element size. Thus, a nonzero value for UP actually changes the value of longitudinal dispersivity used by the simulation and broadens otherwise sharp concentration, temperature or saturation fronts.
GNUP	Real	Pressure boundary condition factor or "conductance". A high value causes SUTRA simulated and specified pressure values at specified pressure nodes to be equal in all significant figures. A low value causes simulated pressure to deviate significantly from specified values. The ideal value of GNUP causes simulated and specified pressures to match in the largest six or seven significant figures only, and deviate in the rest. Trial and error is required to determine an ideal GNUP value for a given simulation by comparing values specified with those calculated at the appropriate nodes for different values of GNUP. An initial guess of 0.01 is suggested.
GNUU	Real	Concentration/temperature boundary condition factor. A high value causes SUTRA simulated values and specified values at specified concentration/temperature nodes to be equal in all significant figures. A low value causes simulated values to deviate significantly from specified values. The ideal value of GNUU causes simulated and specified concentrations or temperatures to match in the largest six or seven significant figures only, and deviate in the rest. Trial and error is required to determine an ideal GNUU value for a given simulation by comparing specified values with those calculated at the appropriate nodes for different values of GNUU. An initial guess of 0.01 is suggested.

DATASET 6: Temporal Control and Solution Cycling Data (one line, followed by one* line for for each schedule, plus one line)

(Lists of times or time steps may be continued over multiple lines in 'TIME LIST' and 'STEP LIST' schedules.)*

Variable	Type	Description
Line 1:		
NSCH	Integer	Number of schedules. In the case of steady transport, NSCH may be set to zero to avoid unnecessarily defining schedules. See note at the end of this dataset for details. Allowed only if transport is steady-state, i.e., CSSTRA='STEADY' in dataset 4.
NPCYC	Integer	Number of time steps in pressure solution cycle. Pressure is solved on time steps numbered: n(NPCYC), where n is a positive integer, as well as on initial time step.
NUCYC	Integer	Number of time steps in temperature/concentration solution cycle. Transport equation is solved on time steps numbered: n(NUCYC), where n is a positive integer, as well as on initial time step. Either NPCYC or NUCYC must be set to 1.
Lines 2 to NSCH+1:		
SCHNAM	Character	Schedule name. May be up to 10 characters long and may include spaces. A given schedule name may not be used more than once. If transport is transient, the user must define a schedule named 'TIME_STEPS', which specifies the starting time for the simulation and the sequence of times at which time steps end. Schedule 'TIME_STEPS' must contain two or more time values, including the starting time.
SCHTYP	Character	One or two words. Schedule type. (See note at the end of this dataset for descriptions of the various schedule types.) Valid values are: 'TIME LIST' = a list of times 'TIME CYCLE' = a sequence of times generated at specified intervals 'STEP LIST' = a list of time steps 'STEP CYCLE' = a sequence of time steps generated at specified intervals

Schedule ‘TIME_STEPS’ must be defined using a time-based schedule type, i.e., either a ‘TIME LIST’ or a ‘TIME CYCLE’.

For a ‘TIME LIST’ schedule:

CREFT	Character	One word. To define the schedule in terms of absolute time (simulation clock time), specify “ABSOLUTE”. To define the schedule in terms of elapsed time (time relative to the simulation starting time), specify “ELAPSED”.
SCALT	Real	Scale factor to be applied to each time value in the list.
NTLIST	Integer	The number of times in the list.
TLIST	Real	The list of times. (May be continued over multiple lines of input.) <u>For schedule ‘TIME_STEPS’</u>: If schedule ‘TIME_STEPS’ is defined as a ‘TIME LIST’ in terms of ELAPSED times, the times in TLIST are assumed to be relative to time TICS specified in dataset 1 of the initial conditions (ICS) file. In that case, the first time in TLIST must be set to zero for schedule ‘TIME_STEPS’, and TICS becomes the (absolute) starting time of the simulation.

For a ‘TIME CYCLE’ schedule:

CREFT	Character	One word. To define the schedule in terms of absolute time (simulation clock time), specify “ABSOLUTE”. To define the schedule in terms of elapsed time (time relative to the simulation starting time), specify “ELAPSED”.
SCALT	Real	Scale factor to be applied to each time value in the list.
NTMAX	Integer	Maximum number of time cycles allowed, i.e., the maximum number of times allowed in the schedule, not including the initial time.
TIMEI	Real	Initial time. Cycling begins at time = TIMEI. <u>For schedule ‘TIME_STEPS’</u>: If schedule ‘TIME_STEPS’ is defined as a ‘TIME CYCLE’ in terms of ELAPSED times, then TIMEI and TIMEL (see below) are assumed to be relative to time TICS specified in dataset 1 of the initial conditions (ICS) file. In that case, the TIMEI must be set to zero for schedule ‘TIME_STEPS’, and TICS becomes the (absolute) starting time of the simulation.

TIMEL	Real	Limiting time. Cycling continues until time ≥ TIMEL.
TIMEC	Real	Initial time increment.
NTCYC	Integer	Number of cycles after which the time increment is updated. The current time increment is multiplied by TCMULT (see below) after every NTCYC cycles. (The value of the time increment is limited by TCMIN and TCMAX; see below.)
TCMULT	Real	Factor by which the time increment is multiplied after every NTCYC cycles.
TCMIN	Real	Minimum time increment allowed.
TCMAX	Real	Maximum time increment allowed.

For a 'STEP LIST' schedule:

NSLIST	Integer	The number of time steps in the list.
ISLIST	Integer	The list of (integer) time steps. (May be continued over multiple lines of input.)

For a 'STEP CYCLE' schedule:

NSMAX	Integer	Maximum number of time step cycles allowed, i.e., the maximum number of time steps allowed in the schedule, not including the initial time step.
ISTEPI	Integer	Initial time step. Cycling begins at time step = ISTEPI.
ISTEPL	Integer	Limiting time step. Cycling continues until time step ≥ ISTEPL.
ISTEPC	Integer	Time step increment.

Last line:

	Character	A single dash, '-', must be placed on the line below the last schedule.

Notes:

The schedules defined in this dataset determine the sequence of time steps (via schedule "TIME_STEPS") and control the timing of observation output (see also dataset 8D). Time or time step values may not be repeated, i.e., each value must be unique, within a given schedule.

For a time-based schedule (TIME LIST or TIME CYCLE), the scale factor SCALT is applied to all numeric time values in the schedule definition. This feature is useful for converting time units. For example, in the definition of a TIME LIST, the list of times TLIST could be input in terms of days and converted to seconds by setting SCALT=86400.

The method of defining a TIME CYCLE is similar to that used to specify time stepping in the former (Version 2.0, 2D3D.1) input format. After the scale factor SCALT has been applied, NTMAX, TIMEL, TIMEC, NTCYC, TCMULT, and TCMAX are analogous to ITMAX, TMAX, DELT, ITCYC, DTMULT, and DTMAX, respectively. However, there are two important differences. First, in a TIME CYCLE, the user specifies the initial time TIMEI; in the former method, it is assumed to be TICS (formerly called TSTART). Second, in a TIME CYCLE, a new time increment size is begun when the cycle number minus one is an integer multiple of NTCYC; the former method begins a new time step size when the time step number is an integer multiple of ITCYC. This second difference is illustrated in the following example, in which DELT=TIMEC=1., DTMULT=TCMULT=2., and ITCYC=NTCYC=5:

Time step or cycle number	Time step or time increment size	
	Old time stepping method	TIME CYCLE method
1	1.	1.
2	1.	1.
3	1.	1.
4	1.	1.
5	2.	1.
6	2.	2.
7	2.	2.
8	2.	2.
9	2.	2.
10	4.	2.

If transport is steady-state, no schedules need be defined, and NSCH may be set to 0. In this case, SUTRA will not read the remainder of Line 1, as this information is not needed. Reading will commence with the next line of data. However, the end-of-list marker, '-', is still required.

Examples:

The following examples assume that time is ultimately specified in seconds. The scale factor 3.15576e+7 is used to convert from years to seconds.

To define a schedule named "A" that consists of five absolute times, 10., 25., 30., 40., and 75. years, one may write the following:

```
'A'  'TIME LIST'  'ABSOLUTE'  3.15576e+7  5  10.  25.  30.  40.  75.
```

To define a schedule named “TIME_STEPS” (which controls time stepping in the simulation), starting at (absolute) time TICS and taking fifty 1-year time steps followed by fifty 2-year time steps, one may write the following:

```
'TIME_STEPS'  'TIME CYCLE'  'ELAPSED'  3.15576e+7  100  0.  1e+99  1.  50  2.  0.  1e+99
```

To define a schedule named “B” that consists of four time steps, 5, 10, 90, and 95, one may write the following:

```
'B'  'STEP LIST'  4  5  10  90  95
```

To define a schedule named “C” that begins at time step 40 and includes every 2nd time step up to and including time step 80, one may write the following:

```
'C'  'STEP CYCLE'  20  40  9999  2
```

DATASET 7A: Iteration Controls for Resolving Nonlinearities (one line)

Variable	Type	Description
ITRMAX	Integer	Maximum number of iterations allowed per time step to resolve nonlinearities. Set to a value of +1 for noniterative solution. Noniterative solution may be used for saturated aquifers when density variability of the fluid is small or for unsaturated aquifers when time steps are chosen to be small.
RPMAX	Real	Absolute iteration convergence criterion for pressure solution. Pressure solution has converged when largest pressure change from the previous iteration's solution at every node in mesh is less then RPMAX. May be omitted for noniterative solution.
RUMAX	Real	Absolute iteration convergence criterion for transport solution. Transport solution has converged when largest concentration or temperature change from the previous iteration's solution at every node in mesh is less than RUMAX. May be omitted for noniterative solution.

DATASET 7B: Matrix Equation Solver Controls for Pressure Solution (one line)

Variable	Type	Description
CSOLVP	Character	Name of solver to be used to obtain the pressure solution. Select one of the following: 'DIRECT' = Banded Gaussian elimination 'CG' = IC-preconditioned conjugate gradient 'GMRES' = ILU-preconditioned generalized minimum residual 'ORTHOMIN' = ILU-preconditioned orthomin Mesh type and dimensionality do not constrain the choice of solver. However, if the DIRECT solver is used, it must be used for **both** the pressure and the transport solution; if either CSOLVP or CSOLVU (DATASET 7C) is set to 'DIRECT', then the other must also be set to 'DIRECT'. Also, the CG solver may be used only in the absence of upstream weighting (UP=0. in dataset 5).
ITRMXP	Integer	Maximum number of solver iterations during pressure solution. May be omitted if the DIRECT solver is used.
TOLP	Real	Convergence tolerance for solver iterations during pressure solution. May be omitted if the DIRECT solver is used.

DATASET 7C: Matrix Equation Solver Controls for Transport Solution (one line)

Variable	Type	Description
CSOLVU	Character	Name of solver to be used to obtain the transport solution. Valid values are as follows: 'DIRECT' = Banded Gaussian elimination 'GMRES' = ILU-preconditioned generalized minimum residual 'ORTHOMIN' = ILU-preconditioned orthomin Mesh type and dimensionality do not constrain the choice of solver. However, if the DIRECT solver is used, it must be used for **both** the pressure and the transport solution; if either CSOLVU or CSOLVP (DATASET 7B) is set to 'DIRECT', then the other must also be set to 'DIRECT'.
ITRMXU	Integer	Maximum number of solver iterations during transport solution. May be omitted if the DIRECT solver is used.
TOLU	Real	Convergence tolerance for solver iterations during transport solution. May be omitted if the DIRECT solver is used.

DATASET 8A: Output Controls and Options for "**.lst**" (Main Output) File and Screen Output (one line)

Variable	Type	Description
NPRINT	Integer	NPRINT is the main output cycle for transient simulation. Output is produced in the .lst file on time steps numbered n | NPRINT | (where n is a positive integer). Also for transient solutions, output is produced for initial conditions and on the first and last time steps. To cancel printed output for the first time step of a transient simulation, set NPRINT to a negative number (i.e., place a minus sign before the desired output cycle). For steady-state solutions, output is produced irrespective of the value of NPRINT.
CNODAL	Character	A value of 'N' cancels output of node coordinates, nodewise element thicknesses, and nodewise porosities. Set to 'Y' for full printout.
CELMNT	Character	A value of 'N' cancels output of elementwise permeabilities and elementwise dispersivities. Set to 'Y' for full output.
CINCID	Character	A value of 'N' cancels output of node incidences in elements. Set to 'Y' for full output.
CVEL	Character	Set to a value of 'Y' to calculate and output fluid velocities at element centroids each time output is produced. Note that for transient flow, velocities are based on results and pressures of the previous time step or iteration and not on the newest values. Set to 'N' to cancel option.
CBUDG	Character	Set to a value of 'Y' to calculate and output a fluid mass budget and energy or solute mass budget each time output is produced. A value of 'N' cancels the option.
CSCRN	Character	Set to a value of 'Y' to write a summary of simulation progress to the screen during the simulation. A value of 'N' suppresses all output to the screen except for certain error messages.
CPAUSE	Character	Set to a value of 'Y' to have SUTRA pause for a user response at the end of the run so that simulation progress can be reviewed on the screen. A value of 'N' cancels the option except for certain error messages. Affects output only if CSCRN='Y'.

Note:

If a "**.smy**" file is assigned in the "**SUTRA.FIL**" input file, a summary of simulation progress may be reviewed after completion of the simulation in the "**.smy**" file (regardless of the value of CSCRN).

DATASET 8B: Output Controls and Options for "**.nod**" File
(Nodewise Results Listed in Columns) (one line)

Variable	Type	Description
NCOLPR	Integer	Nodewise output cycle for transient simulation. Output is produced in the "**.nod**" file on time steps numbered n | NCOLPR | (where n is a positive integer). In addition, for transient solutions, output is produced for initial conditions and on the first and last time steps. To cancel printed output for the first time step of a transient simulation, make NCOLPR a negative number (i.e., place a minus sign before the desired output cycle). For steady-state solutions, output is produced irrespective of the value of NCOLPR.
NCOL	Character	List of names of variables to be printed in columns in the "**.nod**" file. Up to nine columns may be specified. The ordering of columns corresponds to the ordering of variable names in the list. Names may be repeated and may appear in any order, except as noted below. Valid names are as follows: 'N' = node number (if used, it must appear first in list) 'X' = x-coordinate of node 'Y' = y-coordinate of node 'Z' = z-coordinate of node (3D only) 'P' = pressure 'U' = concentration or temperature 'S' = saturation '-' = end of list (any names following '-' are ignored) The symbol '-' (a single dash) must be used at the end of the list.

Example:

To output the 3D node coordinates, pressure, and solute concentration in columns in the "**.nod**" file every 5 time steps, but not on the first time step, write the following:

-5 'X' 'Y' 'Z' 'P' 'U' '-'

DATASET 8C: Output Controls and Options for "**.ele**" File
(Velocities at Element Centroids Listed in Columns)
(one line)

Variable	Type	Description
LCOLPR	Integer	Elementwise output cycle for transient simulation. Output is produced in the "**.ele**" file on time steps numbered n | LCOLPR | (where n is a positive integer). In addition, for transient solutions, output is produced for initial conditions and on the first and last time steps. For steady-state solutions, output is produced irrespective of the value of LCOLPR, and the velocities are reported only once (for time step 1). Velocities for time step 1 are always reported.
LCOL	Character	List of names of variables to be printed in columns in the "**.ele**" file. Up to nine columns may be specified. The ordering of columns corresponds to the ordering of variable names in the list. Names may be repeated and may appear in any order, except as noted below. Valid names are as follows: 'E' = element number (if used, it must appear first in list) 'X' = x-coordinate of element centroid 'Y' = y-coordinate of element centroid 'Z' = z-coordinate of element centroid (3D only) 'VX' = x-component of fluid velocity 'VY' = y-component of fluid velocity 'VZ' = z-component of fluid velocity (3D only) '-' = end of list (any names following '-' are ignored) The symbol '-' (a single dash) must be used at the end of the list.

Note:

Reported velocities for time step 1 are based on initial or steady-state pressures. Reported velocities for subsequent time steps are based on pressures from the previous time step. Velocities used to formulate the transport equation within SUTRA are based on pressures from the previous nonlinearity iteration; thus, the updated velocities used internally may be different from the values reported for each time step in the "**.ele**" file.

Example:

To output the 3D element centroid coordinates and velocity components in columns in the "**.ele**" file every 10 time steps, write the following:

```
10 'X' 'Y' 'Z' 'VX' 'VY' 'VZ' '-'
```

DATASET 8D: Output Controls and Options for "**.obs**" and "**.obc**" Files
(Observation Point Results Listed in Columns)
(one line for each observation point, plus one line)

O M I T when there are no observation points

Variable	Type	Description
Line 1:		
NOBLIN	Integer	NOBLIN is the maximum number of observations output to a single line in a "**.obs**" file. If the total number of observations exceeds NOBLIN, line wrapping will be used to limit the number of observations per line to NOBLIN. Has no effect on output to "**.obc**" files.
Lines 2 to NOBS+1:		
OBSNAM	Character	Observation point name. May be up to 40 characters long and may include spaces. The same observation name may be used more than once.
XOBS	Real	X coordinate of the observation point.
YOBS	Real	Y coordinate of the observation point.
ZOBS	Real	Z coordinate of the observation point. **Omit for 2D problems.**
OBSSCH	Character	Name of the schedule that controls output for this observation point. For transient transport: In addition to the scheduled output, observations are made automatically at time step "0", which represents initial and/or steady-state conditions, depending on the mode in which SUTRA is run. If flow is transient, time step "0" represents initial conditions. If flow is steady-state, time step "0" represents steady-state pressures and saturations and initial concentrations or temperatures. Observations are also made automatically at the end of the last time step. For steady-state transport: User-defined schedules are ignored, and observations are made at time step "1", which represents steady-state results.
OBSFMT	Character	Output format. Must be either "OBS" or "OBC". See the note below for details.

Last line:

Character	A single dash, '-', must be placed on the line below the last observation point.

Notes:

The OBS and OBC output formats correspond to the "**.obs**" and "**.obc**" output files, respectively. Both formats present the same information.

The number of distinct "**.obs**" and "**.obc**" files generated by SUTRA depends on the combinations of output schedule and output format that appear in the list of observation points. All observations that are assigned the same schedule and format are written to the same file. The filename is generated by SUTRA using the corresponding base filename specified by the user in "SUTRA.FIL" and the output schedule.

Example:

The following defines five observation points in a 3D model:

```
2
'well_27'     1007.   1294.   -133.     'A'   'OBS'
'well_29'     1165.    980.   -142.     'A'   'OBS'
'well_30'     1102.    981.   -126.     'A'   'OBS'
'well_198'    2662.    703.   -289.     'B'   'OBS'
'well_344'     155.     49.    -90.     'A'   'OBC'
'-'
```

Assume the base filenames specified in "SUTRA.FIL" for OBS and OBC files are 'project.obs' and 'project.obc'. Then observation data for the first three points listed above are written to the same file, "project_A.obs", in OBS format. Observation data for the third point listed are written to file "project_B.obs" in OBS format. In each OBS file, output is wrapped to the next line after the first two observations are written. Observation data for the last point listed are written to file "project_A.obc" in OBC format.

DATASET 9: Fluid Properties (one line)

Variable	Type	Description
COMPFL	Real	Fluid compressibility, $\beta=(1/\rho)(\partial \rho/\partial p)$. $[M/(L\cdot s^2)]^{-1}$. Note, specific pressure storativity is $S_{op} = (1-\varepsilon)\alpha + \varepsilon\beta$
CW	Real	Fluid specific heat, c_w. [E/(M·ºC)] Set to any arbitrary number (e.g., zero) for solute-transport simulation.
SIGMAW	Real	Fluid diffusivity, σ_w. For energy transport, represents fluid thermal conductivity, λ_w. [E/(L·ºC·s)]. For solute transport represents molecular diffusivity of solute in pure fluid, D_m [L^2/s]. May be decreased from value in pure water to account for tortuosity of fluid paths.
RHOWØ	Real	Density of fluid at base concentration or temperature. [M/L^3].
URHOWØ	Real	Base value of solute concentration (as mass fraction) or temperature of fluid at which base fluid density, RHOWØ, is specified. [M_s/M] or [ºC].
DRWDU	Real	Coefficient of fluid density change with concentration (fraction) or temperature: ρ = RHOWØ + DRWDU (U-URHOWØ). [$M^2/(L^3\cdot M_s)$] or [$M/(L^3\cdot$ºC)]
VISCØ	Real	For solute transport: fluid viscosity, μ, [M/L·s]. For energy transport, this value is a scale factor. It multiplies the viscosity, which is calculated internally in units of [kg/m·s]. VISCØ may be used for energy transport to convert units of [kg/m·s] to desired units of viscosity.

DATASET 10: Solid Matrix Properties (one line)

Variable	Type	Description
COMPMA	Real	Solid matrix compressibility, $\alpha=(1-\varepsilon)^{-1}\ \partial\varepsilon/\partial p$. $[M/(L\cdot s^2)]^{-1}$
CS	Real	Solid grain specific heat, c_s. [E/(M·ºC)] Set to any arbitrary number (e.g., zero) for solute-transport simulation.
SIGMAS	Real	Solid grain diffusivity, σ_s. For energy transport, represents thermal conductivity of a solid grain. [E/(L·°C·s)] Set to any arbitrary number (e.g., zero) for solute-transport simulation.
RHOS	Real	Density of a solid grain, ρ_s. $[M/L^3]$. Value used only for energy-transport simulation or solute-transport simulation with sorption.

DATASET 11: Adsorption Parameters (one line)

Variable	Type	Description
ADSMOD	Character	For no sorption or for energy-transport simulation, set to 'NONE' and leave rest of line blank. For linear sorption model, set to 'LINEAR'. For Freundlich sorption model, set to 'FREUNDLICH'. For Langmuir sorption model, set to 'LANGMUIR'.
CHI1	Real	Value of linear, Freundlich or Langmuir distribution coefficient, depending on sorption model chosen in ADSMOD, χ_1. $[L_f^3/M_G]$.
CHI2	Real	Value of Freundlich or Langmuir coefficient, depending on sorption model chosen in ADSMOD. Set to any real value for linear sorption. χ_2. [1] for Freundlich. $[L_f^3/M_s]$ for Langmuir.

DATASET 12: Production of Energy or Solute Mass (one line)

Variable	Type	Description
PRODFØ	Real	Zero-order rate of production in the fluid, γ_o^w. [(E/M)/s] for energy production, [(M_s/M)/s] for solute mass production.
PRODSØ	Real	Zero-order rate of production in the immobile phase, γ_o^s. [(E/M_G)/s] for energy production, [(M_s/M_G)/s] for adsorbate mass production.
PRODF1	Real	First-order rate of solute mass production in the fluid, γ_1^w. [s^{-1}]. Set to any arbitrary number (e.g., zero) for energy-transport simulation.
PRODS1	Real	First-order rate of adsorbate mass production in the immobile phase, γ_1^s. [s^{-1}]. Set to any arbitrary number (e.g., zero) for energy-transport simulation.

DATASET 13: Orientation of Gravity Vector (one line)

Variable	Type	Description
GRAVX	Real	Component of gravity vector in +x direction. [L/s^2] GRAVX = -$\left\|\underline{g}\right\|$ (∂ ELEVATION/∂ x), where $\left\|\underline{g}\right\|$ is the total acceleration due to gravity in [L/s^2].
GRAVY	Real	Component of gravity vector in +y direction. [L/s^2] GRAVY= -$\left\|\underline{g}\right\|$ (∂ ELEVATION/∂ y), where $\left\|\underline{g}\right\|$ is the total acceleration due to gravity in [L/s^2].
GRAVZ	Real	Component of gravity vector in +z direction. [L/s^2] GRAVZ= -$\left\|\underline{g}\right\|$ (∂ ELEVATION/∂ z), where $\left\|\underline{g}\right\|$ is the total acceleration due to gravity in [L/s^2]. Set to any arbitrary number (e.g., zero) for 2D problems.

DATASET 14A: Scale Factor for Nodewise Data (one line)

Variable	Type	Description
	Character	Line must begin with the word 'NODE'.
SCALX	Real	The scaled x-coordinates of nodes in DATASET 14B are multiplied by SCALX in SUTRA. May be used to change from map to field scales, or from English to SI units. A value of 1.0 gives no scaling.
SCALY	Real	The scaled y-coordinates of nodes in DATASET 14B are multiplied by SCALY in SUTRA. May be used to change from map to field scales or from English to SI units. A value of 1.0 gives no scaling.
SCALZ	Real	For 3D problems, the scaled z-coordinates of nodes in DATASET 14B are multiplied by SCALZ in SUTRA. May be used to change from map to field scales or from English to SI units. A value of 1.0 gives no scaling. For 2D problems, the scaled element (mesh) thicknesses at nodes in DATASET 14B are multiplied by SCALZ in SUTRA. May be used to easily change entire mesh thickness or to convert English to SI units. A value of 1.0 gives no scaling.
PORFAC	Real	The scaled nodewise porosities of DATASET 14B are multiplied by PORFAC in SUTRA. May be used to easily assign a constant porosity value to all nodes by setting PORFAC=porosity, and all POR(II)=1.0 in DATASET 14B.

DATASET 14B: Nodewise Data (one line for each of NN nodes)

Variable	Type	Description
II	Integer	Number of node to which data on this line refers, i.
NREG(II)	Integer	Unsaturated flow property region number to which node II belongs. Set to any integer value when flow simulation is saturated only. The node region number is usually the same as the region number of the elements in which it appears. When the node is at the boundary of two regions, it may be assigned to either region.
X(II)	Real	Scaled x-coordinate of node II, x_i. [L]
Y(II)	Real	Scaled y-coordinate of node II, y_i. [L]
Z(II)	Real	For 3D problems, scaled z-coordinate of node II, z_i. [L] For 2D problems, scaled thickness of mesh at node II. [L] To simulate radial cross sections, set $Z(II) = (2\pi)(radius_i)$, where $radius_i$ is the radial distance from the vertical center axis (axis of radial symmetry) to node i.
POR(II)	Real	Scaled porosity value at node II, ε_i. [1]

Note:

When the DIRECT solver is used, the order in which the nodes are numbered affects the bandwidth of the global banded matrix, NBI, which in turn affects computational and storage efficiency. In this case, **the user should take care to number the nodes to minimize NBI**. SUTRA sets NBI equal to one plus twice the maximum difference in node numbers in the element containing the largest node number difference in the mesh. See Figure 7.1 for an example. When an iterative solver is used, it is still advantageous to minimize NBI, although not as critical as in the case of the DIRECT solver.

DATASET 15A: Scale Factors for Elementwise Data (one line)

Variable	Type	Description
	Character	Line must begin with the word 'ELEMENT'.
PMAXFA	Real	The scaled maximum permeability values, PMAX, in DATASET 15B are multiplied by PMAXFA in SUTRA. May be used to convert units or to aid in assignment of maximum permeability values in elements.
PMIDFA	Real	The scaled middle permeability values, PMID, in DATASET 15B are multiplied by PMIDFA in SUTRA. May be used to convert units or to aid in assignment of maximum permeability values in elements. **Omit for 2D problems.**
PMINFA	Real	The scaled minimum permeability values, PMIN, in DATASET 15B are multiplied by PMINFA in SUTRA. May be used to convert units or to aid assignment of minimum permeability values in elements.
ANG1FA	Real	The scaled angles ANGLE1 in DATASET 15B are multiplied by ANG1FA in SUTRA. For 2D problems, may be used to easily assign a uniform direction of anisotropy by setting ANG1FA=angle, and all ANGLE1(L)=1.0 in DATASET 15B.
ANG2FA	Real	The scaled angles ANGLE2 in DATASET 15B are multiplied by ANG2FA in SUTRA. **Omit for 2D problems.**
ANG3FA	Real	The scaled angles ANGLE3 in DATASET 15B are multiplied by ANG3FA in SUTRA. **Omit for 2D problems.**
ALMAXF	Real	The scaled longitudinal dispersivities ALMAX in DATASET 15B are multiplied by ALMAXF in SUTRA. May be used to convert units or to aid in assignment of dispersivities.
ALMIDF	Real	The scaled longitudinal dispersivities ALMID in DATASET 15B are multiplied by ALMIDF in SUTRA. May be used to convert units or to aid in assignment of dispersivities. **Omit for 2D problems.**
ALMINF	Real	The scaled longitudinal dispersivities ALMIN in DATASET 15B are multiplied by ALMINF in SUTRA. May be used to convert units or to aid in assignment of dispersivities.

Dataset 15A is continued on next page

ATMAXF	Real	The scaled transverse dispersivities ATMAX in DATASET 15B are multiplied by ATMAXF in SUTRA. May be used to convert units or to aid in assignment of dispersivity.
ATMIDF	Real	The scaled transverse dispersivities ATMID in DATASET 15B are multiplied by ATMIDF in SUTRA. May be used to convert units or to aid in assignment of dispersivity. **Omit for 2D problems.**
ATMINF	Real	The scaled transverse dispersivities ATMIN in DATASET 15B are multiplied by ATMINF in SUTRA. May be used to convert units or to aid in assignment of dispersivity.

DATASET 15B: Elementwise Data (one line for each of NE elements)

Variable	Type	Description
L	Integer	Number of element to which data on this line refers.
LREG(L)	Integer	Unsaturated flow property region number to which this element belongs. Set to any integer value when flow simulation is saturated only.
PMAX(L)	Real	Scaled maximum permeability value of element L, $k_{max}(L)$. [L^2]
PMID(L)	Real	Scaled middle permeability value of element L, $k_{mid}(L)$. [L^2] Isotropic permeability requires: PMID(L)=PMAX(L). **Omit for 2D problems.**
PMIN(L)	Real	Scaled minimum permeability value of element L, $k_{min}(L)$. [L^2] Isotropic permeability requires: PMIN(L)=PMAX(L).
ANGLE1(L)	Real	Scaled first angle, $\theta_1(L)$ [°], used to define the directions of maximum, middle, and minimum permeability in element L.
ANGLE2(L)	Real	Scaled second angle, $\theta_2(L)$ [°], used to define the directions of maximum, middle, and minimum permeability in element L. **Omit for 2D problems.**
ANGLE3(L)	Real	Scaled third angle, $\theta_3(L)$ [°], used to define the directions of maximum, middle, and minimum permeability in element L. **Omit for 2D problems.** **See the note titled "Permeability" below for the definition of ANGLE1, ANGLE2, and ANGLE3.**

Dataset 15B is continued on next page

ALMAX(L)	Real	Scaled longitudinal dispersivity value of element L that controls longitudinal dispersion along the maximum permeability direction when the flow direction is in the maximum permeability direction, $\alpha_{Lmax}(L)$. [L]
ALMID(L)	Real	Scaled longitudinal dispersivity value of element L that controls longitudinal dispersion along the middle permeability direction when the flow direction is in the middle permeability direction, $\alpha_{Lmid}(L)$. [L] **Omit for 2D problems.**
ALMIN(L)	Real	Scaled longitudinal dispersivity value of element L that controls longitudinal dispersion along the minimum permeability direction when the flow direction is in the minimum permeability direction, $\alpha_{Lmin}(L)$. [L]
ATMAX(L)	Real	Scaled transverse dispersivity value of element L that controls transverse dispersion in the maximum permeability direction when the flow direction is perpendicular to the maximum permeability direction, $\alpha_{Tmax}(L)$. [L]
ATMID(L)	Real	Scaled transverse dispersivity value of element L that controls transverse dispersion in the middle permeability direction when the flow direction is perpendicular to the middle permeability direction, $\alpha_{Tmid}(L)$. [L] **Omit for 2D problems.**
ATMIN(L)	Real	Scaled transverse dispersivity value of element L that controls transverse dispersion in the minimum permeability direction when the flow direction is perpendicular to the minimum permeability direction, $\alpha_{Tmin}(L)$. [L]

Notes:

The SUTRA permeability model is described in detail in section 2.2, "Saturated-Unsaturated Ground-Water Flow." The SUTRA dispersion model is described in detail in section 2.5, "Dispersion." The notes that follow are included to further assist the user in specifying the input parameters for permeability and dispersion in SUTRA models, particularly in 3D.

Permeability

The "permeability ellipse" and "permeability ellipsoid" that form the basis of the SUTRA permeability model in 2D and 3D, respectively, are described in section 2.2 and pictured in Figure 2.2.

In 3D, the principal axes of the permeability ellipsoid are, by definition, mutually perpendicular and aligned with the directions of maximum, middle, and minimum permeability. SUTRA requires that the orientation of this ellipsoid relative to the x-, y-, and z-coordinate axes be defined for each element in the mesh by specifying three parameters -- ANGLE1, ANGLE2, and ANGLE3 – in each element. These three angles

may be thought of, in aeronautical terms, as the "yaw," "pitch," and "roll" of the permeability ellipsoid with respect to a reference orientation, which is described below.

In defining ANGLE1, ANGLE2, and ANGLE3, we make the following assumptions and definitions:

- The Cartesian (x, y, z) coordinate system is right-handed: it can be viewed such that the +x-axis points forward, the +y-axis points to the left, and the +z-axis points upward.
- The maximum, middle, and minimum permeability axes are the principal axes of the permeability ellipsoid. Initially, before the three rotations defined by ANGLE1, ANGLE2, and ANGLE3 have been performed, let these three principal axes be aligned with the x-, y-, and z-coordinate axes, respectively; this is the reference orientation. After the three rotations are completed, the maximum, middle, and minimum permeability axes will be aligned with the directions of maximum, middle, and minimum permeability, respectively.
- The *positive* maximum, middle, and minimum permeability axes (which will be called the "+max," "+mid," and "+min" axes below) are the semi-axes that are initially aligned with the +x-, +y-, and +z-axes, respectively.

The rotations and corresponding angles are then defined as follows:

- The first rotation of the permeability ellipse is about the z-axis. After this rotation has been performed, ANGLE1 is the angle between the +max axis and the +x-axis. It is measured within the x,y-plane, with a positive angle denoting *counterclockwise* rotation when viewed from positive z, looking toward the origin. It represents the azimuth of the +max axis.
- The second rotation of the permeability ellipse is upward or downward from the x,y-plane. After this rotation has been performed, ANGLE2 is the angle between the +max axis and the x,y-plane. It is measured perpendicularly from the x,y-plane, with a positive angle denoting upward rotation (toward the +z-axis). It represents the angular elevation or declination of the +max axis.
- The third rotation of the permeability ellipse is about the +max axis. After this rotation has been performed, ANGLE3 is the angle between the +mid axis and the x,y-plane. It is measured within the plane perpendicular to the +max axis, with a positive angle denoting *clockwise* rotation when viewed from the origin, looking along the +max axis.

In 3D simulations, ANGLE3 is arbitrary if the permeability and dispersion tensors are isotropic within the (MID,MIN)-plane, that is, if, after the application of scale factors, PMIN=PMID, ALMIN=ALMID, and ATMIN=ATMID. All three angles, ANGLE1, ANGLE2, and ANGLE3, are arbitrary if the permeability and dispersion tensors are completely isotropic, that is, if, after the application of scale factors, PMIN=PMID=PMAX, ALMIN=ALMID=ALMAX, and ATMIN=ATMID=ATMAX.

In 2D simulations, ANGLE1 is arbitrary if the permeability and dispersion tensors are isotropic, that is, if, after application of scale factors, PMIN=PMAX, ALMIN=ALMAX, and ATMIN=ATMAX.

The convention for determining the 2D transverse dispersivity, α_T, differs from the one used in versions of SUTRA (Voss, 1984) prior to version 2.0 (2D3D.1), as described in section 2.5.

In the SUTRA dispersion model in 3D, the effective longitudinal dispersivity is computed as the squared radius of a "longitudinal dispersivity ellipsoid" (pictured in Figure 2.4b) in the direction of ground-water flow, as described in section 2.5. For simplicity, the principal axes of this ellipsoid are assumed to be aligned with the directions of maximum, middle, and minimum permeability, which are mutually perpendicular and are specified by parameters ANGLE1, ANGLE2, and ANGLE3 for each element.

The dispersivities ALMAX(L), ALMID(L), and ALMIN(L) represent the squared radii of the longitudinal dispersivity ellipsoid in the maximum, middle, and minimum permeability directions, respectively, for element L. Thus, ALMAX, ALMID, and ALMIN are the effective longitudinal dispersivities for flow in the maximum, middle, and minimum permeability directions, respectively. Note that "MAX," "MID," and "MIN" do not refer to the relative magnitudes of the dispersivities, but rather to the direction in which they apply.

Use of different longitudinal dispersivities for various flow directions may be justified in a few ways. Differences in longitudinal dispersivity in various flow directions may either be due to a local anisotropy in porous medium or aquifer structure, or to the different sizes of heterogeneities experienced by flows along vertical and horizontal transport reaches in an aquifer system. Regional horizontal flows typically encounter much larger heterogeneities than flows occurring vertically through an aquifer, causing higher longitudinal dispersion for horizontal flows than for vertical flows.

The effective transverse dispersivities in 3D may be computed as the squares of two radii of a "transverse dispersivity ellipsoid" (pictured in Figure 2.4c) measured perpendicular to the direction of ground-water flow, as described in section 2.5. For simplicity, the principal axes of this ellipsoid are also assumed to be aligned with the directions of maximum, middle, and minimum permeability.

The dispersivities ATMAX(L), ATMID(L), and ATMIN(L) represent the squared radii of the transverse dispersivity ellipsoid in the maximum, middle, and minimum permeability directions, respectively, for element L. Note that:

- For all flow directions within the (MAX,MID)-plane, ATMIN is the effective dispersivity that controls transverse dispersion in the MIN direction.
- For all flow directions within the (MAX,MIN)-plane, ATMID is the effective dispersivity that controls transverse dispersion in the MID direction.
- For all flow directions within the (MID,MIN)-plane, ATMAX is the effective dispersivity that controls transverse dispersion in the MAX direction.

It follows that when the flow direction coincides with one of the principal permeability directions, the effective transverse dispersivities are those corresponding to the remaining two principal permeability directions:

- For flow in the MAX permeability direction, the effective transverse dispersivities are ATMID and ATMIN.
- For flow in the MID permeability direction, the effective transverse dispersivities are ATMAX and ATMIN.
- For flow in the MIN permeability direction, the effective transverse dispersivities are ATMAX and ATMID.

For any given flow direction in 3D, there are two transverse dispersivities. Thus, for flow in the maximum, middle, and minimum permeability directions, there would be a maximum of six different transverse dispersivity values (two for each of the three directions, assuming that flow in exactly opposite directions have the same transverse dispersivities). However, the SUTRA model assumes that transverse dispersivity in a given direction is the same irrespective of in which perpendicular direction the flow occurs, and thus allows only three different values to be specified for each element: ATMAX, ATMID, and ATMIN. The user must decide, based on the description of the dispersion model in section 2.5 and the information outlined above, which values best describe the behavior of the system being simulated.

Use of different transverse dispersivities for various flow directions is not as easily justified as flow-direction-dependent longitudinal dispersivities. Normally, the flow-direction-dependent transverse dispersivities should be set to the same value (unless the user has a specific dispersion behavior in mind). This results in the same effective transverse dispersion in all directions for all flow directions just as given by the classical model.

DATASET 16: *no longer used*

DATASET 17: Data for Fluid Sources and Sinks (one line for each of NSOP fluid source nodes as specified in DATASET 3, plus one line)

O M I T when there are no fluid source nodes

Variable	Type	Description
Lines 1 to NSOP:		
IQCP	Integer	Number of node to which source/sink data on this line refers. Specifying the node number with a negative sign indicates to SUTRA that the source flow rate, concentration, or temperature of the source fluid varies in a specified manner with time. Information regarding a time-dependent source node must be programmed by the user in Subroutine BCTIME.
QINC	Real	Fluid source (or sink) which is a specified constant value at node IQCP, Q_{IN}. [M/s]. A positive value is a source of fluid to the aquifer. May be omitted if this value is specified as time-dependent in Subroutine BCTIME (IQCP<0). Sources are allocated by cell as shown in Figure B.1 for equal-sized elements. For unequal-sized elements, sources are allocated in proportion to the cell length, area or volume over which the source fluid enters the system.
UINC	Real	Temperature or solute concentration (mass fraction) of fluid entering the aquifer, which is a specified constant value for a fluid source at node IQCP, U_{IN}. [°C] or [M_s/M] May be omitted if this value is specified as time-dependent in Subroutine BCTIME (IQCP<0) or if QINC≤0.
Last line:		
	Integer	Placed immediately following all NSOP fluid source node lines. Line must begin with the integer 0.

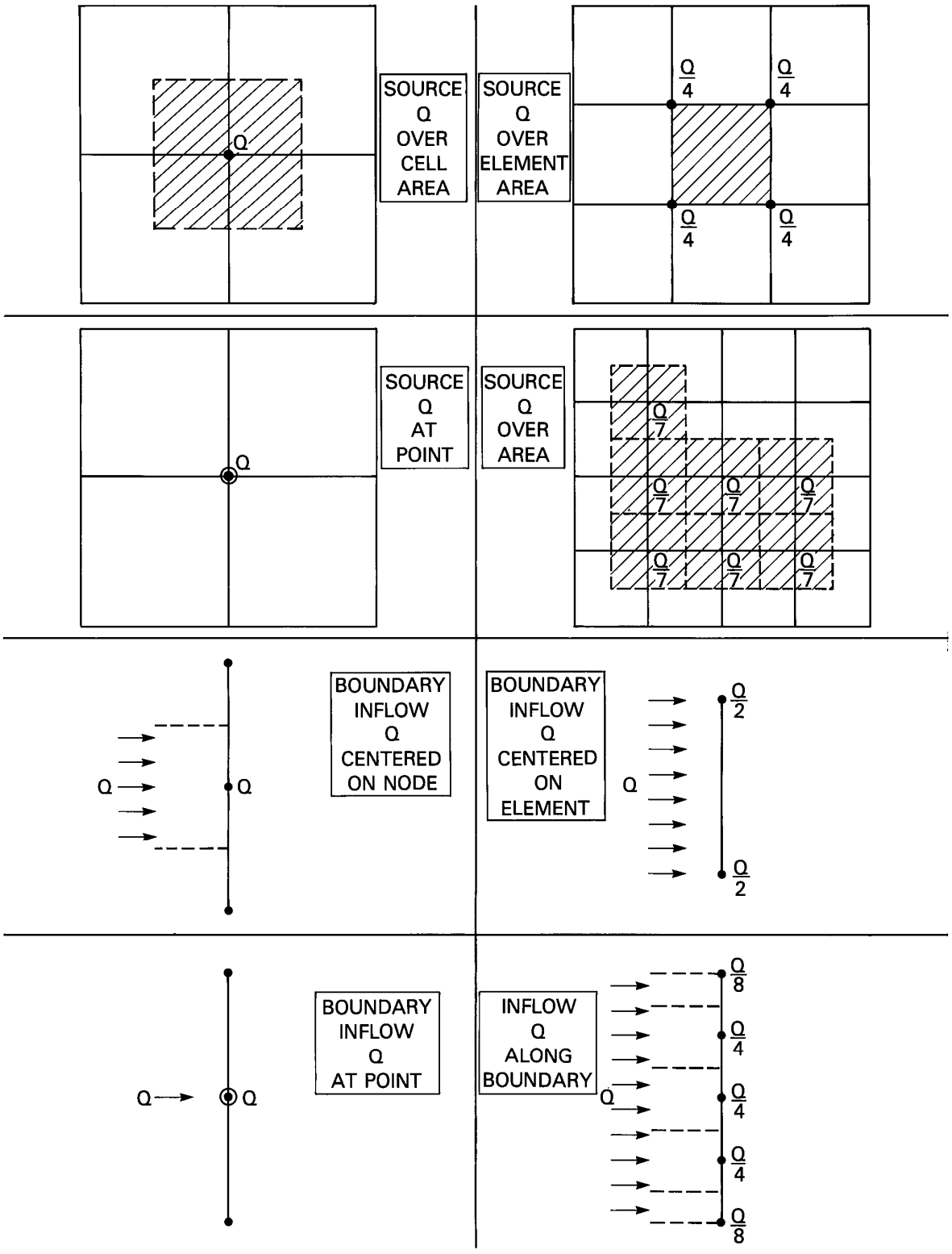


Figure B.1. Allocation of sources and boundary fluxes in equal-sized elements. The top four panels pertain to 2D areal and 3D meshes. The bottom four panels pertain to 2D cross-sectional meshes. Though sources are always specified at nodes, a variety of spatial source distributions may be obtained by appropriate specification of nodal source values.

DATASET 18: Data for Energy or Solute Mass Sources and Sinks
(one line for each NSOU energy or solute source nodes as specified in DATASET 3, plus one line)

O M I T when there are no energy or solute source nodes

Variable	Type	Description
Lines 1 to NSOU:		
IQCU	Integer	Number of node to which source/sink data on this line refers. Specifying the node number with a *negative sign* indicates to SUTRA that the source rate varies in a specified manner with time. All information regarding a time-dependent source node must be programmed by the user in Subroutine BCTIME. Sources are allocated by cell as shown in Figure B.1 for equal-sized elements. For unequal-sized elements, sources are allocated in proportion to the cell length, area or volume over which the source energy or solute mass enters the system.
QUINC	Real	Source (or sink) that is a specified constant value at node IQCU, ψ_{IN}. [E/s] for energy transport, [M_s/s] for solute transport. A positive value is a source to the aquifer. May be omitted if this value is specified as time-dependent in Subroutine BCTIME (IQCU <0).
Last line:		
	Integer	Placed immediately following all NSOU energy or solute mass source node lines. Line must begin with the integer 0.

DATASET 19: Data for Specified Pressure Nodes (one line for each of NPBC specified pressure nodes as indicated in DATASET 3, plus one line)

O M I T when there are no specified pressure nodes

Variable	Type	Description
Lines 1 to NPBC:		
IPBC	Integer	Number of node to which specified pressure data on this line refers. Specifying the node number with a negative sign indicates to SUTRA that the specified pressure value or inflow concentration or temperature at this node varies in a specified manner with time. Information regarding a time-dependent specified pressure node must be programmed by the user in Subroutine BCTIME.
PBC	Real	Pressure value which is a specified constant at node IPBC. $[M/(L \cdot s^2)]$. May be omitted if this value is specified as time-dependent in Subroutine BCTIME.
UBC	Real	Temperature or solute concentration of any external fluid that enters the aquifer at node IPBC. UBC is a specified constant value. [°C] or $[M_s/M]$. May be omitted if this value is specified as time-dependent in Subroutine BCTIME.
Last line:		
	Integer	Placed immediately following all NPBC specified pressure lines. Line must begin with the integer 0.

DATASET 20: Data for Specified Concentration or Temperature Nodes
(one line for each of NUBC specified concentration or temperature nodes indicated in DATASET 3, plus one line)

O M I T when there are no specified concentration or temperature nodes

Variable	Type	Description
Lines 1 to NUBC:		
IUBC	Integer	Number of node to which specified concentration or temperature data on this line refers. Specifying the node number with a negative sign indicates to SUTRA that the specified value at this node varies in a specified manner with time. This time-dependence must be programmed by the user in Subroutine BCTIME.
UBC	Real	Temperature or solute concentration value which is a specified constant at node IUBC. [°C] or [M_s/M]. May be omitted if IUBC is negative and this value is specified as time-dependent in Subroutine BCTIME.
Last line:		
	Integer	Placed immediately following all NUBC specified temperature or concentration lines. Line must begin with the integer 0.

DATASET 21: *no longer used*

DATASET 22: Element Incidence Data (one line, plus one line for each of NE elements)

Variable	Type	Description
Line 1:		
	Character	Line must begin with the word 'INCIDENCE'.
Lines 2 to NE+1:		
LL	Integer	Number of element to which data on this line refers.
IIN	Integer	Node incidence list; list of corner node numbers in element LL, beginning at any node. For 2D problems, the four nodes are listed in an order counterclockwise about the element. For 3D problems, the eight nodes are listed as follows. Approach the element from any of its six sides. On the face farthest away (the "back" face, viewed looking through the element), list the four nodes in an order counterclockwise about the face. Then, on the closest face (the "front" face), again list the four nodes counterclockwise, starting with the node directly in front of the node that was listed first. (This convention assumes a right-handed coordinate system.)

End of Input Data for the Main Input File (**.inp**)

List of Input Data for the Initial Conditions File (**.ics**)

Model Version: ***SUTRA 2.1***

The data in the "**.ics**" file need be created by the user only for the very first time step of a given simulation or series of restarted simulations. Thereafter, if the user has chosen to optionally store the final results of the simulation in a "**.rst**" file, this "**.rst**" file may be used directly as the "**.ics**" file for later restarts. The restart options are controlled by CREAD and ISTORE in DATASET 4 of the "**.inp**" file.

DATASET 1: Simulation Starting Time (one line)

Variable	Type	Description
TICS	Real	Time (in seconds) to which the initial conditions specified in the "**.ics**" file correspond. [s]. TICS can be used as the starting time of the simulation by defining schedule 'TIME_STEPS' in terms of ELAPSED times – see description of dataset 6 in the main input (INP) file. Usually set to a value of zero for a "cold start".

Example:

If the initial conditions correpsond to time 1990 years on the simulation clock, set TICS=(1990 yrs)(3.15576x10^7 s/yr)= 6.2799624x10^{10} s as follows:

```
6.2799624d+10
```

See the description of "**.inp**" dataset 6 for conversion factors between various time units.

DATASET 2: Initial Pressure Values at Nodes (two lines; second line can be broken up over multiple lines)

Variable	Type	Description
Line 1:		
CPUNI	Character	One word. Set to 'UNIFORM' to specify a uniform pressure for all nodes. Set to 'NONUNIFORM' to specify a separate pressure for each node.
Line 2:		
PVEC	Real	For UNIFORM pressure specification, a single value of initial (starting) pressure to be applied at all NN nodes at the start of the simulation. [$M/(L \cdot s^2)$] For NONUNIFORM pressure specification, a list of values of initial (starting) pressures at the start of the simulation, one value for each of NN nodes, in exact order of node numbers. [$M/(L \cdot s^2)$]. List can be broken up over multiple lines, and any number of values may be placed on each line (as long as no line contains more than 1000 characters). If the STEADY (steady-state) flow option in DATASET 4 of the "**.inp**" file has been chosen, PVEC serves as an initial guess for the pressure solution when an ITERATIVE solver is used, and is ignored when the DIRECT solver is used. Initial hydrostatic or natural pressures in a cross-sectional or 3D model may be obtained by running a single steady-flow time step with the store option. Then the natural pressures are calculated and stored in the "**.rst**" file, and may be copied to the corresponding section of the "cold start" "**.ics**" file without change in format, to be used as initial conditions for a transient run.

DATASET 3: Initial Temperature or Concentration Values at Nodes (two lines; second line can be broken up over multiple lines)

Variable	Type	Description
Line 1:		
CUUNI	Character	One word. Set to 'UNIFORM' to specify a uniform temperature to solute concentration for all nodes. Set to 'NONUNIFORM' to specify a separate value for each node.
Line 2:		
UVEC	Real	For UNIFORM temperature or solute concentration specification, a single initial (starting) value to be applied at all NN nodes at the start of the simulation. [°C] or [M_S/M] For NONUNIFORM temperature or solute concentration specification, a list of initial (starting) values at the start of the simulation, one value for each of NN nodes, in exact order of node numbers. [°C] or [M_S/M]. List can be broken up over multiple lines, and any number of values may be placed on each line (as long as no line contains more than 1000 characters).

End of Input Data for the Initial Conditions File (**.ics**)
