PNNL-14286

Pacific Northwest
National Laboratory
Operated by Battelle for the
U.S. Department of Energy

STOMP
Subsurface Transport Over Multiple Phases

Version 3.0

User's Guide

M. D. White
M. Oostrom

June, 2003

Prepared for the U.S. Department of Energy
under Contract DE-AC06-76RL01830

DISCLAIMER

PNNL-14286

STOMP
Subsurface Transport Over Multiple Phases

Version 3.0

User's Guide

M. D. White
M. Oostrom

June 2003

Prepared for the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest National Laboratory
Richland, Washington 99352

Preface

This guide describes the general use, input file formatting, compilation and execution of the STOMP (Subsurface Transport Over Multiple Phases) simulator, a scientific tool for analyzing single and multiple phase subsurface flow and transport. A description of the simulator's governing equations, constitutive functions and numerical solution algorithms are provided in a companion theory guide. In writing these guides for the STOMP simulator, the authors have assumed that the reader comprehends concepts and theories associated with multiple-phase hydrology, heat transfer, thermodynamics, radioactive chain decay, and relative permeability-saturation-capillary pressure constitutive relations. The authors further assume that the reader is familiar with the computing environment on which they plan to compile and execute the STOMP simulator.

The STOMP simulator is written in the FORTRAN 77 and 90 languages, following American National Standards Institute (ANSI) standards. The simulator utilizes a variable source code configuration, which allows the execution memory and speed be tailored to the problem specifics, and essentially requires that the source code be assembled and compiled through a software maintenance utility.The memory requirements for executing the simulator are dependent on the complexity of physical system to be modeled and the size and dimensionality of the computational domain. Likewise execution speed depends on the problem complexity, size and dimensionality of the computational domain, and computer performance.

Summary

The STOMP (Subsurface Transport Over Multiple Phases) simulator has been developed by the Pacific Northwest National Laboratory[a] for modeling subsurface flow and transport

[a] Pacific Northwest National Laboratory is operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract DE-AC06-

systems and remediation technologies. The development of the STOMP simulator started when the U. S. Department of Energy, through the Office of Technology Development, requested the demonstration of remediation technologies for the cleanup of volatile organic compounds and associated radionuclides within the soil and groundwater at arid sites. This demonstration program, called the VOC-Arid Soils Integrated Demonstration Program (Arid-ID), has been initially directed at a volume of unsaturated and saturated soil contaminated with carbon tetrachloride, on the Hanford Site near Richland, Washington. A principal subtask of the Arid-ID program involved the development of an integrated engineering simulator for evaluating the effectiveness and efficiency of various remediation technologies. The engineering simulator's intended users include scientists and engineers who are investigating hydrologic and multifluid flow phenomena associated with remediation technologies. Principal design goals for the engineer simulator include broad applicability, verified algorithms, quality assurance controls, and validated simulations against laboratory and field-scale experiments. An important goal for the simulator development subtask involved the ability to scale laboratory and field-scale experiments to full-scale remediation technologies, and to transfer acquired technology to other arid sites.

The STOMP simulator's fundamental purpose is to produce numerical predictions of thermal and hydrogeologic flow and transport phenomena in variably saturated subsurface environments, which are contaminated with volatile or nonvolatile organic compounds. Auxiliary applications include numerical predictions of solute transport processes including radioactive chain decay processes. Quantitative predictions from the STOMP simulator are generated from the numerical solution of partial differential equations that describe subsurface environment transport phenomena. Description of the contaminated subsurface environment is founded on governing conservation equations and constitutive functions. Governing coupled flow equations are partial differential equations for the conservation of water mass, air mass, volatile organic compound mass, salt or surfactant mass, and thermal energy. Constitutive functions relate primary variables to secondary variables. Solution of the governing partial differential equations occurs by the integral volume finite difference method. The governing equations that describe thermal and hydrogeological flow processes are solved simultaneously using Newton-Raphson iteration to resolve the nonlinearities in the governing equations. Governing transport equations are partial differential equations for the conservation of solute mass. Solute mass conservation governing equations are solved sequentially, following the solution of the coupled flow equations, by a direct application of the integral volume finite difference method. The STOMP simulator is written in the FORTRAN 77 and 90 languages, following American National Standards Institute (ANSI) standards. The simulator utilizes a variable source code configuration, which allows the execution memory and speed be tailored to the problem specifics, and essentially requires that the source code be assembled and compiled through a software maintenance utility.

KEYWORDS: subsurface, porous media, multiple phase, groundwater, nonaqueous phase liquid (NAPL), volatile organic compound (VOC), variably saturated, brines, nonequilibrium kinetics, solute transport, radioactive chain decay, hysteresis, fluid entrapment, finite-difference, Newton-Raphson, nonlinear, modeling, TVD transport, banded linear system solver, conjugate gradient solver, relative permeability, CO_2 sequestration,

Acknowledgements

This work was partly supported by the Groundwater/Vadose Zone Integration Project funded through the U.S. Department of Energy's Richland Operations Office. In addition to programmatic support, the continued development of the STOMP simulator in its sequential and parallel implementations has been funded by the Laboratory Directed Research and Development (LDRD) program at the Pacific Northwest National Laboratory. In particular development of a scalable implementation has been funded through the Computational Science and Engineering Initiative and development of new operational modes for modeling carbon dioxide sequestration has been supported through the Carbon Management Initiative. Laboratory Directed Research and Development (LDRD) at Pacific Northwest National Laboratory is a productive and efficient program that develops technical capabilities for solving complex technical problems that are important to the Department of Energy and to the nation. DOE Order 413.2A sets forth the Department's LDRD policy and guidelines for DOE multiprogram laboratories and authorizes the national laboratories to allocate up to six percent of their operating budgets to fund the program.

Table of Contents

List of Figures

List of Tables

1. Introduction

The initial development of the simulator was supported by the VOC-Arid Soils Integration Demonstration Program, abbreviated as Arid-ID and funded by the U.S. Department of Energy (DOE), Office of Technology Development (OTD). The program was directed at the cleanup of volatile organic compounds and associated radionuclides and heavy metals in soils and groundwater at arid sites. The initial demonstration site is located within the 200 West Area on the Hanford Site near Richland, Washington. The site contains a volume of soil contaminated with the Dense Nonaqueous Phase Liquid (DNAPL) carbon tetrachloride, which includes approximately 200 vertical feet of contaminated unsaturated sediments underlying inactive disposal sites and overlying a 7-square-mile plume of contaminated groundwater. A critical component of the Arid-ID program involved assessing the impact of spatial heterogeneity of subsurface materials on remediation processes and evaluating the effectiveness and efficiency of demonstrated remedial technologies. Because of the complexity of subsurface flow and heat transport phenomena, these assessments and evaluations required complex numerical tools for their completion. Numerical tools allow scientists and engineers to integrate the current knowledge of contaminant behavior in the subsurface environment to predict and evaluate the performance of proposed remediation methods against established technologies.

A principal subtask of the Arid-ID program involved the development of an engineering simulator (a numerical tool), which is capable of numerically simulating proposed remediation processes. The design goals were that the engineering simulator would: 1) be accessible and exploitable to scientists and engineers familiar with subsurface environment phenomena, but not necessarily numerical modeling technicalities, 2) have enough general applicability to recruit a user group that is capable of supporting training, maintenance, and enhancement activities, 3) be verified by comparisons to analytical solutions and benchmarked against existing simulators, 4) be validated against germane laboratory and field experiments, and 5) have controlled configuration and documentation under an appropriate quality assurance program. An engineering simulator named STOMP, an acronym for Subsurface Transport Over Multiple Phases, has been developed by the Pacific Northwest National Laboratory[a] which achieves the

---

(a) Pacific Northwest National Laboratory is operated by Battelle Memorial Institute for the U.S. Department of Energy under Contract DE-AC06-

five design goals described above. This document, the STOMP V3.0 User's Guide, has been written to provide users of the STOMP simulator with necessary information for selecting an appropriate operational mode, understanding the code flow path and design, creating input files, dimensioning the executable, compiling and executing, and interpreting simulation outputs.

This guide has been organized in a manner that duplicates the normal approach to solving a subsurface flow and transport problem with the STOMP simulator. The STOMP simulator has been designed with a variable source code, where source code configurations are referred to as operational modes. Operational modes are classified according to the solved governing flow and transport equations and constitutive relation extensions. Therefore, prior to creating an input file or assembling the source code the user must choose the appropriate operational mode for the particular subsurface system of interest. The selection of an operational mode requires that the user conceptualize the physical system as a computational system, which will always require making simplifying assumptions about the physical system. The complexity and execution speed of an operational mode is generally inversely related to the number of simplifying assumptions the user can justify about the physical system. Regardless of the operational mode complexity, the simulation of physical subsurface systems with the STOMP simulator always involves assumptions inherent to the founding governing equations and constitutive relations. The operational modes supported in STOMP V3.0 are:

STOMP1	Water or Fluid Mode
STOMP2	Water-Air Mode
STOMP3	Water-Air-Energy Mode
STOMP4	Water-Oil Mode
STOMP5	Water-Air-Oil Mode
STOMP6	Water-Air-Oil-Energy Mode
STOMP8	Water-Oil-Dissolved Oil Mode
STOMP11	Water-Salt Mode
STOMP12	Water-Air-Salt Mode
STOMP13	Water-Air-Salt-Energy
STOMP32	Water-CO2-NaCl
STOMP33	Water-CO2-NaCl-Energy

Having chosen an operational mode, the user should then set about creating an *input* file. Input files, simply stated, are translations of the physical system description into a computational system. The user communicates information about the physical system to the STOMP simulator through the *input* file. The *input* file, operational mode, and linear system solver determine the source code configuration and required dimension of the executable. Executable dimensions are controlled through parameters, which are communicated to the STOMP simulator through the *parameters* file. It is the user's responsibility to ensure that the parameters listed in the *parameters* file are sufficiently dimensioned for the *input* file, operational mode, and selected linear system solver. Having created *input* and *parameters* files that describe the physical system to be modeled, the next step in the simulation process involves assembling the source code, creating an executable, and executing the code.

As previously mentioned, the STOMP simulator has a variable source code configuration, which is dependent on the selected operational mode. The source coding for each operational mode comprises mode specific coding and global coding from a library of routines. Assembly and compilation of the source coding for a particular operational mode has been facilitated through a UNIX utility called *make* (Talbott, 1988). Instructions for using these utilities are described in this document. The STOMP simulator offers the user considerable flexibility in executing and controlling the execution of simulations. The user can create numerous execution periods within a single simulation, which differ with respect to time step advancement and convergence control. Controls of this sort are beneficial for simulations involving complex time-varying boundary conditions or sources. Moreover, the user can produce intermediate *restart* files which are essentially "snapshots" of the primary variables. Simulations can be restarted from intermediate *restart* files using the altered or unaltered descriptions of the computational system. Various techniques and approaches for executing the simulator are also described in this document.

The STOMP simulator generates results in a number of different formats, which can be controlled by the user. This guide describes these formats and the output capabilities of the simulator. Output formats include text files of simulation results and printing to the standard input/output device, which is typically a terminal or workstation screen. All *input* and *output* files generated by the simulator including *restart* files are ASCII text files, which can be read or modified by conventional text editors. Output data has been grouped in two basic formats, historic and snapshot. Historic data are records of selected variables over the simulation period. These data records are essential for creating graphs or plots showing the progression of variable values over simulation time. Snapshot data are records of selected variables at a particular moment in time over the computation domain. These data records are essential for generating images of variable values over the computational space. Snapshot data records can also be linked in a time sequence to show the evolution of variable values in both computational time and space.

Beyond providing operational type instructions, this document also contains useful information for executing the simulator with a debug utility or making modifications to the source code. The user may often be able to isolate input errors or convergence problems by tracing the execution of the simulator with a debug utility. To assist the user in performing debug executions or in making modifications to the code, critical information about the operational flow path, subroutine functions, and variable descriptions have been provided in this guide. The source coding for the STOMP simulator is suitably documented with comment statements, however none of the variable or parameter names are defined nor is a general flow path for simulator outlined. Therefore, this guide provides essential information about subroutines and variables. Indexing of primary and secondary field variables and flux variables in the STOMP simulator is complex and depends on the operational mode. A description of the indexing patterns developed for the simulator is also provided in this guide.

2. Fundamentals

2.1 Introduction

The STOMP simulator has been designed to solve a wide variety of nonlinear, multiple-phase, flow and transport problems for variably saturated geologic media. Partial differential conservation equations for component mass, energy, and solute mass comprise the fundamental equations for the simulator. Coefficients within the fundamental equations are related to the primary variables through a set of constitutive relations. The conservation equations for component mass and energy are solved simultaneously, whereas the solute transport equations are solved sequentially after the coupled flow solution. The variable source code configuration allows the user to select the combination of solved fundamental equations. The current version of the STOMP simulator recognizes ten coupled flow equation combinations. Each coupled flow equation combination is referred to as an operational mode and may additionally include the solution of a number of transported solutes. The associated constitutive relations for each recognized operational mode are automatically incorporated into the source code as required.

In reading this document, it is important to distinguish between phases and components. Phases are composed of components. The terms aqueous, gas, nonaqueous phase liquid (NAPL), ice, and solid will be used exclusively in referring to phases. The terms water, air, CO_2, oil, salt, and refer to components. The aqueous phase is primarily composed of liquid water with lesser amounts of dissolved oil, CO_2, air, and salt. The gas-phase composition can be highly variable, and may contain air, water vapor and oil vapor. The NAPL phase is assumed to be composed only of liquid oil with no dissolved air or water. The current version of the STOMP simulator permits only a single oil component within the NAPL phase (i.e., a noncompositional model). Dissolved oil and transported solutes can sorb to the solid phase (i.e., the solid rock/soil matrix).

The STOMP simulations are limited in application scope according to the solved fundamental equations, the associated constitutive theory, inherent assumptions, computer execution speed and memory, and the user's creativity. A critical component to correct application of the STOMP simulator and comprehension of output results requires an understanding by the user of the assumptions taken to develop the various flow and transport algorithms. The simulator is capable of predicting flow and transport behavior for a variety of subsurface systems, however, application of the simulator to problems which violate an inherent assumption in the simulator's design or fundamental equations could yield incorrect results. STOMP is principally limited to flow through variably saturated porous media, which can be characterized with an extended form of Darcy's law. Additionally, the simulator is limited to a maximum of three immiscible phases referred to as the aqueous, nonaqueous liquid, and gas phases. Low solubilities are assumed for the liquid phases and interphase mass transfer assumes equilibrium conditions in most modes. The Water-Oil-Dissolved Oil Mode allows kinetic dissolution of the NAPL. Specific to the energy equation, the principal assumptions are that

heat transport by gas-phase conduction and the kinetic nature of thermodynamic processes are neglected. Solute transport solutions are computed sequentially to the coupled flow equations. This approach requires the assumption that solutes are passive scalars with respect to the flow equations, which is equivalent to assuming solute concentrations are dilute.

The STOMP simulator solves transient flow and transport problems in the subsurface environment in one, two, or three dimensions. Coordinate systems must be orthogonal and currently are limited to Cartesian, tilted Cartesian, and cylindrical, where the vertical coordinate of the cylindrical system must be aligned with the gravitational vector. The STOMP simulator solves steady-state problems either directly or through false-transients starting from a user specified initial state. Direct solutions to steady-state problems are possible for initial conditions sufficiently close to the solution, therefore, transient solutions to steady-state conditions are the recommended approach. All boundary conditions, sources, and sinks are time variant and allow the user considerable control over transient simulations. Hydrogeologic properties can be spatially varied throughout the computational domain, within the resolution of a node volume. Hydraulic and thermal transport properties for the porous medium can be anisotropic, where the diagonal tensor elements are assumed to be aligned with the principal axes of the coordinate system. Coupled flow solutions can be obtained for selected one-, two-, or three-phase systems under isothermal or nonisothermal conditions. Transport of radioactive solutes with chain-decay tracking can be coupled to all transport solutions with the assumption of dilute solute concentrations. A variety of boundary conditions are available for each operational mode, which may be applied selectively over the boundary surfaces. Unspecified boundary surfaces are always assumed to be zero flux surfaces. Inactive nodes can be specified by the user and boundary conditions can be applied to surfaces separating active (computational) and inactive (noncomputational) nodes. Selected operational modes allow the user to invoke a dynamic domain option which temporarily sets quiescent nodes to an inactive set, thus removing them from the computational domain.

The fundamental coupled flow equations are solved following a integral volume finite-difference approach with the nonlinearities in the discretized equations resolved through Newton-Raphson iteration. Linear systems which result from the Newton-Raphson linearization or the solute transport solution can be solved with a direct banded matrix solver and an indirect conjugate gradient based solver. These linear system solvers are commercially available software products that have been adapted for the particulars of the STOMP simulator. The STOMP simulator allows considerable control over simulation parameters related to convergence, time stepping, solution techniques, and execution limits. A single simulation can be divided into multiple execution periods, each with a different set of solution control parameters. Restart capabilities have also been included in the simulator, which can resume a simulation from user define points with or without alterations to input parameters. Output from the simulator can be totally controlled by the user and is written both to files and to the standard input/output device (e.g., screen). Output forms included time histories of selected variables, time “snap shots” of selected variables across the computational domain, and variable integrals for sources and fluxes across boundary and internode surfaces.

The dominant nonlinear functions within the STOMP simulator are the relative permeability-saturation-capillary pressure (k–S–P) relations. The STOMP simulator allows the

user to specify these relations through a large variety of popular and classic functions. Two-phase (water-air) *k–S–P* relations can be specified with hysteretic or nonhysteretic functions or nonhysteretic tabular data. Entrapment of air with imbibing water conditions can be modeled with the hysteretic two-phase *k–S–P* functions. Two-phase *k–S–P* relations span both saturated and unsaturated conditions. The aqueous phase is assumed to never completely disappear through extensions to the *S–P* function below the residual saturation and a vapor pressure lowering scheme. Three-phase (aqueous-NAPL-gas) *k–S–P* relations can be specified with hysteretic or nonhysteretic functions. Entrapment of nonwetting fluids (i.e., gas in aqueous, gas in NAPL, and NAPL in aqueous) with imbibing wetting fluid conditions is the only hysteretic process that is currently supported with the hysteretic three-phase *k–S–P* functions. The wettability order for fluids in the STOMP simulator is assumed to follow the descending order aqueous to NAPL to gas. Three-phase *k–S–P* relations span both total-liquid saturated and unsaturated conditions and aqueous saturated and unsaturated conditions. Phase appearances and disappearances are possible for the gas and NAPL phases, however, the aqueous phase is assumed to never completely disappear through extensions to the *S–P* function below the residual saturation and a vapor pressure lowering scheme.

2.2 Operational Modes

The STOMP simulator operational modes do not comprise all of the possible combinations of coupled governing equations. The combination sets of coupled governing equations selected for inclusion in the list of operational modes represent those with the greatest utility for physical systems. For example, a two-phase nonvolatile nonisothermal operational mode could be envisioned, which solved the water mass and energy conservation equations. This operational mode would invoke the assumption of a nonparticipating gas phase. Two-phase flow and transport through porous media under thermal gradients, however, strongly depends on the diffusion, dispersion, and advection transport through gas phase, even for low thermal gradients. Therefore, this operational mode while capable of functioning and producing converged solutions would have limited utility, because its associate premise that gas phase transport could be neglected. Other combinations of governing equation sets with associated assumptions have considerable utility for specific systems or problems and have been coded, but have not been selected for inclusion in the STOMP guides. For example, an operational mode has been created that solves the water mass, air mass, oil mass, and/or energy equation with the assumption that oil concentrations remain below the aqueous solubility limit. This operational mode differs from one which models the oil as a dilute solute, because the oil mass equation is solved simultaneously with the other coupled flow equations and fluid properties have a dependence on the oil concentration. This operational mode has utility in the investigation of the remediation of oil contaminants in dissolved plumes.

Operational modes are indentified according to the coupled conservation equations that are solved. All operational modes support solute transport calculations. The solved conservation equations and primary assumptions are summarized in Table 2-1 for each operational mode.

Table 2-1 Operational Mode Summary

Operational Mode w/ Options	Solved Coupled Equations	Primary Assumptions
Water (STOMP1)	water mass	isothermal conditions passive gas phase no NAPL phase no dissolved oil no brine local thermodynamic equilibrium
Water-Air (STOMP2)	water mass air mass	isothermal conditions no NAPL phase no dissolved oil no brine local thermodynamic equilibrium
Water-Air-Energy (STOMP3)	water mass air mass thermal energy	no NAPL phase no dissolved oil no brine local thermodynamic equilibrium
Water-Oil (STOMP4)	water mass oil mass	isothermal conditions single component NAPL phase passive gas phase no brine local thermodynamic equilibrium
Water-Air-Oil (STOMP5)	water mass oil mass air mass	isothermal conditions single component NAPL phase no brine local thermodynamic equilibrium
Water-Air-Oil-Energy (STOMP6)	water mass oil mass air mass thermal energy	single component NAPL phase no brine local thermodynamic equilibrium
Water-Oil-Dissolved Oil (STOMP8)	water mass oil mass dissolved oil mass	isothermal conditions single component NAPL phase passive gas phase no brine kinetic oil dissolution or local thermodynamic equilibrium

Table 2-1 (Contd.)

Operational Mode w/ Options	Solved Coupled Equations	Primary Assumptions
Water-Salt (STOMP11)	water mass salt mass	isothermal conditions passive gas phase no NAPL phase no dissolved oil local thermodynamic equilibrium
Water-Air-Salt (STOMP12)	water mass air mass salt mass	isothermal conditions no NAPL phase no dissolved oil local thermodynamic equilibrium
Water-Air-Energy-Salt (STOMP13)	water mass air mass salt mass thermal energy	no NAPL phase no dissolved oil local thermodynamic equilibrium
Water-CO_2-Salt (STOMP32)	water mass CO_2 mass NaCl mass	isothermal conditions no NAPL phase no dissolved oil local thermodynamic equilibrium
Water-CO_2-Energy-Salt (STOMP33)	water mass CO_2 mass NaCl mass thermal energy	no NAPL phase no dissolved oil local thermodynamic equilibrium

3. Code Design

3.1 Introduction

The primary design guides for the STOMP simulator have been modularity, computational efficiency, and readability. Modular code architecture is beneficial because of the ease of reading, maintaining, and modifying the algorithms and is essential to the variable configuration source code. Computational efficiency refers to both memory requirements and execution speed. The STOMP simulator has been designed with a variable configuration source code which allows the memory requirements and code algorithms to be partially customized to the computational problem. This approach offers considerable advantages with respect to achieving a computationally efficient code design. Within this source code framework, however, many design choices have been made that affect computational efficiency. Algorithm design often offers options between memory and speed. For example, to lessen memory requirements a code designer may opt to repeatedly compute commonly used variables. Conversely, execution speed may be increased at the cost of increased memory requirements, by storing commonly used variables after their initial computation. Generally, the approach in the STOMP simulator has been to favor increased memory requirements to gain computational speed. This design approach has been chosen because of current state of computer architecture and capabilities. Because the STOMP simulator has been created as a scientific tool, algorithm readability has been a primary design guide. As a scientific tool, the simulator was never expected to remain unmodified, but rather a constantly changing package of software tools which could be applied to new or more complex problems. This design goal makes readability an essential feature of the code. Code readability has been achieved through an extensive use of comments, a modular design, a large group of common blocks, and minimal subroutine and function arguments.

3.2 Flow Path

The general flow path for all operational modes of the STOMP simulator comprises three components, initialization, iteration, and closure. A flow chart for the initialization, iteration, and closure components of the main program is shown in Figure 3.1, where the enclosing boxes indicate either a single or group of routines. The initialization component of the program is executed once during a simulation. The routines in the initialization component are executed in sequence shown in Figure 3.1, from the program start to the start of the first time step. The iteration component of the program contains a pair of nested loops, an outer loop for time stepping and an inner loop for Newton-Raphson linearization. Termination of the Newton-Raphson loop occurs with a successful convergence or after an iteration limit violation.

Termination of the time-stepping loop after a simulation limit or a time-step reduction limit violation has occurred. Regardless of the cause for termination during the time-stepping loop the closure routines are executed at the simulation completion. The transport solution is shown as a single routine on the STOMP flow diagram. It comprises, however, several transport routines within a solute loop. The flow diagram for the solute transport portion of the iteration component is shown in Figure 3.2.

The initialization component (Initialize Variables) of the program flow path begins by initializing all common variables. All common variables, those in the common blocks of the *commons* file, are initialized at this point either to zero or with default values. Unless specifically defaulted, integers are initialized to 0, real variables are initialized to 0.D+0, and character strings are initialized to blank. During the variable initializations the *input* and *output* files are opened. The next routine (Print File Banners) prints the welcome statement, disclaimer, and banner to the standard output device (screen) and the *output* file. This is followed with the procedures (Read Input File) for reading the *input* file. The *input* file is read using a predefined card order. After each card has been read, the *input* file is rewound and searched from the beginning for the next card to be read. This approach allows the user to sequence input cards randomly within the *input* file. The card read order defined within the code software is critical and should not be altered. When appropriate the *restart* file is read for input data information during these procedures. Input data are then checked for saturation or thermodynamic state consistency (Check Physical States). If an error is found in these routines the simulation will be terminated with an associated error message. If no errors in state conditions are noted then the program continues the initializing process with the Jacobian matrix pointers (Set Matrix Pointers).

The Jacobian matrix structure varies with operational mode and grid geometry. The reader is referred to the STOMP Theory Guide (White and Oostrom, 2000) for a description of the numerical methods and linear system solvers. Jacobian matrix pointers are integer arrays that relate primary variables for a particular node to a location in the Jacobian matrix. If the simulation does not involve dynamic domains then these pointers are constants during the simulation and need to be computed only once. The next three initialization routines (Saturation Properties, Physical Properties and Solute Concentrations) compute initial values for the secondary variables from the initial conditions specified through the *input* or *restart* files. These routines have been divided into three components: those for computing phase saturation related variables, thermodynamic properties, and solute concentrations.

The iteration component of the program flow path contains a pair of nested loops. The outer loop increments time and represents a single time step and the inner loop increments iterations of the Newton-Raphson linearization technique. During a single time increment loop (time step) both the flow and transport governing equation sets are solved. The Newton-Raphson linearization loop is applicable only to the solution of the coupled mass and heat flow governing equations. The solute transport governing equations are solved directly (without iteration) and sequentially to the iterative flow solution (Start Time Step). Each time step loop starts with a computation of the new time step and increments to the time and time step counter. Time steps are computed with an algorithm based on the most previous complete time step, the time step acceleration factor, and the time to a transition point. Transition points occur with

changes in execution period times, boundary condition times, source times, and output times. Time steps always conclude on transition points, which commonly require the time step to be temporarily reduced. The time step following a temporary reduction to meet a transition point time will resume the prior time stepping levels, except in the case of execution period transitions where a new initial or maximum time step has been declared. The next procedure (Load Old Time Arrays) involves loading the previous time step arrays for field variables. Field variables from the array location for the current field variable value are loaded into the array location for the previous time step field variable value. The next three routines (Reference Node Output, Write Plot File and Write Restart File) prior to starting a Newton-Raphson iteration loop involves writing previous time step results to output. Simulation results are written to *output* file the standard input/output device (screen), a *plot.n* file, and a *restart.n* file, depending on the directives made by the user on the Output Control Card.

The Newton-Raphson iteration loop solves the coupled governing flow equations for component mass and energy. The first procedure (Boundary Properties) within this loop involves the calculation of boundary surface properties. Each declared boundary surface, has associated field variables which are computed with the same algorithms as the node field variables, but only when the boundary condition is active. This approach eliminates the necessity for computing field variable values for disabled boundary surfaces. The next group of routines (Interior Fluxes) computes fluxes across interior surfaces (those surfaces between active nodes). Fluxes which are computed within these routines are dependent on the operational mode and include Darcy phase velocities, component diffusion-dispersion fluxes, thermal conductive flux, thermal advective flux, and thermal diffusion-dispersion fluxes. The same group of flux variables is computed in the next procedure (Boundary Fluxes) for each active boundary surface. As with the procedure for computing field variables on boundary surfaces the approach of computing flux variables only for active boundary surfaces eliminates computing unused boundary flux variables. The next procedure (Source Contributions) computes source and/or sink contributions to each of the governing flow equations from the user specified inputs on the Source Card. Source contributions are stored in arrays and subsequently used in computing the Jacobian matrix coefficients. At this point in the Newton-Raphson iteration loop, all field and surface flux variables have been computed, along with the source contributions. These variables compose the primary components of the governing flow equations.

The Jacobian matrix coefficients and solution vector are computed in a multiple stage sequence. The first stage involves setting all of the previous coefficient arrays to zero. This stage is necessary because the nonzero elements of the Jacobian matrix will change with time step as various boundary condition transition between active and inactive states. The second stage (Load Jacobian Matrix) involves computing the Jacobian matrix and solution vector with the assumption of zero flux boundary conditions for all boundary surfaces, including surfaces between active and inactive nodes. The Jacobian matrix loading procedure depends on the operational mode, but is sequenced according to governing partial differential equations. The resulting system of equations represents the discretized and linearized system of governing flow equations with zero flux boundary conditions imposed, where the source contributions have been incorporated. The final stage (Boundary Matrix Modify) modifies this linear system according to the active user imposed boundary conditions. Boundary conditions will alter both the

coefficient matrix and solution vector. With the Jacobian matrix and solution vector elements computed, the next procedure (Solve Linear System) involves solving the linear system of equations. The linear system is solved either with a direct banded matrix solver or a iterative conjugate gradient solver. Both routines return corrections to the primary variables in the solution vector array.

Corrections to the primary variables, computed from the linear system solvers, are used to update the primary variables and determine convergence. The Newton-Raphson procedure computes corrections to the primary variable set with each iteration. The starting values for primary variables for each new time step are the previous time step values of the primary variables, as these values represent reasonable estimates of the future values. For a convergent iteration scheme, each successive iteration yields diminishing corrections to the primary variables. Phase transitions and primary variable switching schemes, however, can yield temporary increases in the correction to a particular primary variable. The two procedures (Update Primary Variables and Compute Convergence) that immediately follow the linear system solver procedures update the primary variables and determine convergence. Convergence occurs if the normalized values of the primary variable corrections for all unknowns falls below a user defined value (typically 1.0×10^{-6}). The next three (Primary Increments, Saturation Properties and Physical Properties) are executed independent of the convergence result. If convergence occurs (Convergence Switch) then these computed values represent the current values of secondary variables at the conclusion of the time step; otherwise they represent the current iterate values of the secondary variables. If convergence occurs then the solution procedure continues with solute transport procedures. At the conclusion of nonconvergent iterations two additional checks are made. If the iteration count does not exceed the user specified limit then program proceeds with a new Newton-Raphson iteration loop. If the iteration count exceeds the limit (Iteration Limit), then a check is made on the count of successive time step reductions. If convergence has failed and resulted in a time step reduction four times in succession (Time Reduction Limit), then the simulation aborts and program execution is transferred to the closure routines. Otherwise, the time step is reduced (Reduce Time Step), the program execution is transferred to the beginning of the time increment loop, and another attempt is made reach a converged solution for the time step.

The transport solution procedure follows the iterative solution of the mass and heat flow equations and, although a direct solution scheme, involves looping over the number of solutes. A procedure flow diagram for the transport solution routines is shown in Figure 3.2. Prior to entering the solute loop the interior-surface and boundary-surface flux procedures are called to obtain values of all flux variables at the conclusion of a time step. Flux values at the conclusion of a time step will typically vary from those computed during the last iteration of the current time step, because the primary and secondary variables will have been updated near the bottom of the last Newton-Raphson iteration loop. The surface flux calculations within the Newton-Raphson loop differ from these surface calculations in that only the current unincremented value of the surface flux is computed. Refer to Section 3.4 for a description of surface flux variable arrays.

The transport solution procedure loops over the number of solutes in the reverse order that they are defined on the Solute/Fluid Interactions Card. A reverse looping order is used to

compute progeny solutes prior to computing parent solutes. This approach allows sequential coupling between solutes that decay radioactively or chemically with first order reaction rates yielding solute products. The first step (Solute Partitioning) of the transport solution loop involves computing the equilibrium distribution of solute between the fluid and solid phases. The second step (Zero Jacobian Matrix) is to initialize the coefficient matrix and solution vector elements to zero. As with the flow solution scheme the possibility for boundary conditions and sources that transition makes initializing the linear system elements mandatory. The third step (Solute Sources) of the transport solution loop is to compute the solute source and/or sink contributions. Solute source contributions are incorporated directly into the coefficient matrix and solution vector elements. The fourth step (Load Jacobian Matrix) involves loading the coefficient matrix and solution vector. As with the flow solution scheme, the coefficient matrix and solution vector elements are computed assuming zero-flux conditions on all boundary surfaces, then modified for the boundary conditions (Boundary Matrix Modify). Element loading depends on the operational mode and occurs sequentially by phases, where the aqueous phase contributions are loaded first followed by the gas and NAPL phases. Solid phase contributions are loaded with the first active fluid phase. The fifth step (Solve Linear System) of the transport solution procedure is the solution of linear system of equations. Algorithms for solving the transport linear system of equations are identical to those for the flow solution, except that the returned results for the transport solution are directly the volumetric solute concentrations. The sixth step (Update Solute Conc.) of the transport solution is to update the solute concentrations with their newly computed values. The concluding steps (Zero Solute Fluxes, Compute Solute Fluxes and Integrate Solute Sources) for the transport solution compute solute fluxes and integrate the solute sources. Once the transport solution loop has been executed for every solute the program execution returns to the flow solution procedures. Time steps for the transport solution are by default equal to those used for the flow solution. Although not currently an option with the STOMP simulator, the transport solution could be advanced in fractional values of the flow solution time steps.

The closure routines are only executed once at the successful or unsuccessful conclusion of a simulation. These routines generate final *plot.n* files, *restart.n* files, and close all opened files. Upon successful conclusion to the simulation the final *restart.n* file will contain a record of the primary variables at the conclusion of the final time step. Conversely, if a simulation concludes unsuccessfully, because of a convergence failure or otherwise, the *restart.n* file will contain a record of the primary variables at the conclusion of the previous converged time step.

3.3 Subroutines

Each operational mode of the STOMP simulator comprises global and mode dependent subroutines. Global subroutines are those subroutines that are generally included in more than one operational mode. Mode dependent subroutines, however, are associated with a single operational mode. Subroutine names are generally descriptive abbreviations. These abbreviations frequently contain the letters G, L, or N, which respectively correspond to the gas, aqueous, or NAPL phases. Other common letters are A, O, and W, which represent air or CO_2, oil and water

components, respectively. The letters C and T often refer to the solute transport and energy equations. A bold letter **B** refers to a group of similar subroutines written for the bottom, south, west, east, north, and top surfaces, respectively. Each surface will have a corresponding subroutine with the subroutine name having a B, S, W, E, N, or T in place of the bold letter **B** to represent the bottom, south, west, east, north, or top surface, respectively. A short description of a subroutine's function can generally be found in the heading portion of the source coding. Mode dependent subroutine names contain a numerical suffix, which corresponds to the operational mode. Refer to Table 3.1 for operational mode index definitions. Every subroutine and function begins with a series of comment statements which includes disclaimers, a short description of the coding, a creation date stamp, and a modification date stamp. The modification date stamp is used to track the last modification date for a particular subroutine. The STOMP simulator variable structure was primarily designed around large arrays held in common blocks, which are included in nearly every subroutine. Therefore, subroutines generally have few arguments. The subroutines associated with the hysteretic saturation functions were written prior to the STOMP simulator and generally have a different programming style than other portions of the code.

3.4 Variables

The STOMP simulator has been designed with the principal variables defined in common blocks, which are included in nearly all subroutines. The common variables are listed in Table 3.2. This approach reduces the number of arguments that are passed between routines and increases the readability of the code, because variable names remain unchanged between routines. The STOMP simulator has been coded without equivalence statements and no variables are temporarily overwritten to save memory. Variable names generally follow the intrinsic protocols for FORTRAN 77, where integer variables begin with letters in the range "I-N" and real valued variables begin with letters in the ranges "A-H" and "O-Z." Variables that begin with the letter "L" generally indicate integer parameters.

Field variables are defined at node points and are represented with two dimensional arrays. Examples of field variables include primary unknowns, saturation properties, rock/soil properties, and thermodynamic properties (e.g., temperature, pressure, phase saturation, porosity, density, and viscosity). The first index of a field variable indicates the time step or increment status. The second index of a field variable indicates the node number. The dimension of the first index depends on the operational mode or equivalently number of coupled flow equations; where the dimension will equal the number of unknowns plus two. Therefore, field variables at each node comprise a number of elements equal to the number of unknowns plus two. Each element for each field variable contains a variant on the field variable. For example, the first element or index contains the value of the field variable at the previous time step. A list of field variable variants is shown in Table 3.3.

The primary variable order and corresponding field variable variant indexing depend on the operational mode. However, primary variables are always ordered, regardless of the operational mode, according to the following equation sequence: energy, water mass, air mass, oil

mass, dissolved-oil mass, salt mass, and surfactant mass. The primary variable ordering system for each operational mode is listed in Table 3.4. Primary variables are referred to by equation in Table 3.4, because primary variables for a given equation and operational mode are dependent on the local phase condition. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a description of primary variables, phase conditions, and variable switching. Consider an example for the Water-Air Operational Mode, which solves the water mass and air mass conservation equations. For this operational mode there would be four field variable variant indices. Using the aqueous saturation for an unsaturated node as the example field variable, the following variant definitions would apply: the first index would refer to the aqueous saturation at the previous time step, the second index would refer to the aqueous saturation at the current iteration or time step value, the third index would refer to the aqueous saturation with the aqueous pressure incremented, and the fourth index would refer to the aqueous saturation with the gas pressure increments.

Flux variables are defined on node surfaces between node points or on boundary surfaces and are represented with two dimensional arrays. Examples of flux variables include heat fluxes, Darcy phase velocities, and component diffusion/dispersion fluxes. Flux variables are aligned with one of the primary orthogonal coordinate directions. Flux variables names which begin with the character "U" are aligned with the x- or radial-direction coordinate, those which begin with the character "V" are aligned with the y- or azimuthal-direction coordinate, and those which begin with the character "W" are aligned with the z-direction coordinate. The first index of a flux variable indicates the increment status and the second index indicates the surface number. Unlike field variables, previous time-step fluxes are not stored. Surface numbers do not correspond directly to node numbers. The number of surfaces in any given coordinate direction equals the number of nodes in that direction plus one.

A list of the flux variable variants is shown in Table 3.5. The primary variable order and corresponding field variable variant index depend on the operational mode. In Table 3.5 the upper node refers to the node in the east, north, or top direction (positive *x*-, *y*-, or *z*-direction) with respect to the surface. Likewise the lower node refers to the node in the west, south, or bottom direction (negative *x*-, *y*-, or *z*-direction) with respect to the surface. Consider an example for the Water-Air Operational Mode, which solves the water mass and air mass conservation equations. For this operational mode there would be five flux variable variant indices. Using the z-direction aqueous Darcy velocity for an unsaturated node as the example field variable, the following variant definitions would apply: the first index would refer to the aqueous Darcy velocity for the current iteration or time step value, the second and third indices would refer to the Darcy velocity with the aqueous pressure in the "top" and "bottom" nodes incremented, respectively; and the fourth and fifth indices would refer to the Darcy velocity with the gas pressure in the "top" and "bottom" nodes incremented, respectively.

Table 3.1. Glossary of Operational Mode Indices.

STOMP Mode Index	Operational Mode
1	Water
2	Water-Air
3	Water-Air-Energy
4	Water-Oil
5	Water-Air-Oil
6	Water-Air-Oil-Energy
8	Water-Oil-Dissolved Oil
11	Water-Salt
12	Water-Air-Salt
13	Water-Air-Salt-Energy
32	Water-CO_2-NaCl
33	Water- CO_2-NaCl-Energy

Table 3.2. Description of Global Variables (*commons* and *allo.f*)

AFX(LSX)	X-Direction Node Area, m^2
AFY(LSY)	Y-Direction Node Area, m^2
AFZ(LSZ)	Z-Direction Node Area, m^2
AGLF(LSV,LFDG)	Interfacial Area between Free Gas and Aqueous, m^2
AGLT(LSV,LFDG)	Interfacial Area between Trapped Gas and Aqueous, m^2
ALU(LJD,LJE)	Jacobian Coefficient Matrix
ANLF(LFDN)	Specific Aqueous-Free NAPL Interfacial Area, 1/m
ANLT(LFDN)	Specific Aqueous-Trapped NAPL Interfacial Area, 1/m
ASGT(LFD)	Apparent Trapped Gas Saturation
ASGTL(LFD)	Apparent Trapped Gas Saturation, Aqueous
ASGTN(LFD)	Apparent Trapped Gas Saturation, NAPL
ASL(LFD)	Apparent Aqueous Saturation
ASLMIN(LSU,LFD)	Minimum Apparent Aqueous Saturation
ASLSC(LSU,LFD)	Apparent Aqueous Saturation at Reversal Point
ASNMIN(LSU,LFD)	Minimum Apparent NAPL Saturation
ASNR(LFD)	Apparent Residual NAPL Saturation
ASNT(LFD)	Apparent Trapped NAPL Saturation
AST(LFD)	Apparent Total-Liquid Saturation
ASTMAX(LSU,LFD)	Maximum Apparent Total-Liquid Saturation
ASTMIN(LSU,LFD)	Minimum Apparent Total-Liquid Saturation
ATMC(5)	Atmospheric Condition Measurement Parameters
	(1) Wind Speed Measurement Height, m
	(2) Air Temperature/Relative Humidity Measurement Height, m
	(3) Local Longitude, deg
	(4) Local Latitude, deg
	(5) Local Meridian, deg
ATMOS(6,LATM)	Atmospheric Condition Parameters
	(2,LATM) Time, s
	(2,LATM) Temperature, C
	(3,LATM) Pressure, Pa
	(4,LATM) Relative Humidity, %
	(5,LATM) Incident Solar Radiation, W/m^2
	(6,LATM) Wind Speed, m/s
ATMST	Atmospheric Conditions Data Start Time, s
BAO	Gas-NAPL Scaling Factor
BAW	Gas-Aqueous Scaling Factor
BC(LBCV,LBTM,LBC)	Boundary Condition Variables
BCG(LBCV,LBTM,LBCGR)	LUGR Subgrid Boundary Condition Variables
BIG	Constant, 1020
BLU(LJF)	Solution Vector
BOW	NAPL-Aqueous Scaling Factor
BTGL(LSV,LFD)	Gas-Aqueous Scaling Factor
BTGLB(LSV,LBC)	Gas-Aqueous Scaling Factor at Boundary
C_OBDT(LOBDT)	Field-Observation Output Units* (64 characters)

C(LFDC,LSOLU)	Solute Concentration, 1/m^3
CA_AO	Gas-NAPL Interfacial Contact Angle, radians
CA_AW	Gas-Aqueous Interfacial Contact Angle, radians
CA_OW	NAPL-Aqueous Interfacial Contact Angle, radians
CARD	Card Name* (64 characters)
CB(LBCC,LSOLU)	Boundary Solute Concentration, 1/m^3
CCL_CRN(LSOLU)	Aqueous-Phase Cut-0ff Concentration for Courant-Limit Control
CHDATE	Date* (10 characters)
CHDF(LSOLU,LSOLU)	Solute Chain Decay Fraction
CHML(LRC)	Characteristic Matrix Length, m
CHMSG	Error Message* (132 characters)
CHREF(LOUPV)	Reference Node Variable Abbreviation* (4 characters)
CHSF(LOUPV,3)	Surface Flux Variable Abbreviation* (4 characters)
CHTIME	Time* (8 characters)
CIMTC(4)	Oil Interphase Mass Tranfer Coefficients
CLKMX	Maximum Clock Time
CLKSEC	Clock Time, s
CLU(LJA)	Solution Vector
CMP(3,LRC)	Rock/Soil Compressibility Parameters (1,LRC) Matrix Compressiblity, 1/Pa (2,LRC) Fracture Compressibility, 1/Pa (3,LRC) Compressibility Reference Pressure, Pa
CMPNY	Company Name*
CMTLN(4,LSOLU)	NAPL-Aqueous Interphase Mass Transfer Coefficients
CNL(LFDC,LSOLU)	Solute Concentration in NAPL
CNLB(LBCC,LSOLU)	Boundary Solute Concentration in NAPL
CNLO(LFDC,LSOLU)	Old Time Step Solute Concentration in NAPL
CO(LFDC,LSOLU)	Old-time-step Solute Concentration, 1/m^3
CPAC(4)	Alcohol Isobaric Molar Specific Heat Constants
CPOC(4)	Oil Isobaric Molar Specific Heat Constants
CPS(LRC)	Rock/Soil Specific Heat, J/kg K
CPUMX	Maximum CPU Time
CPUSEC	CPU Time, s
CRNTG(LFD)	Gas Courant Number
CRNTL(LFD)	Aqueous Courant Number
CRNTMX	Courant Number Upper Limit
CRNTN(LFD)	NAPL Courant Number
DETAF(LFD)	Generalized Curvilinear Coordinate Factor
DETAP(LSY)	Generalized Curvilinear Coordinate Factor
DFGO(LSV,LFDG)	Gas Oil Diffusion Coefficient, m^2/s
DFGOB(LSV,LBCG)	Boundary Gas Oil Diffusion Coefficient, m^2/s
DFGOC	Oil Vapor Diffusion Coefficient, m^2/s
DFGW(LSV,LFDG)	Gas Water Diffusion Coefficient, m^2/s
DFGWB(LSV,LBCG)	Boundary Gas Water Diffusion Coefficient, m^2/s
DFGWC	Water Vapor Diffusion Coefficient, m^2/s
DFGWW(LSV,LNWN)	Well Water-Vapor Diffusion Coefficient, m^2/s

DFGWWB(LSV,LNW)	Well Boundary Water-Vapor Diffusion Coefficient, m^2/s
DFLA(LSV,LFD)	Aqueous Air Diffusion Coefficient, m^2/s
DFLAB(LSV,LBC)	Boundary Aqueous Air Diffusion Coefficient, m^2/s
DFLAC	Dissolved Air Diffusion Coefficient, m^2/s
DFLAW(LSV,LNWN)	Well Dissolved Air Diffusion Coefficient, m^2/s
DFLAWB(LSV,LNW)	Well Boundary Dissolved Air Diffusion Coefficient, m^2/s
DFLO(LSV,LFD)	Aqueous Oil Diffusion Coefficient, m^2/s
DFLOB(LSV,LBC)	Boundary Aqueous Oil Diffusion Coefficient, m^2/s
DFLOC	Dissolved Oil Diffusion Coefficient, m^2/s
DFLOW(LSV,LNWN)	Well Dissolved Oil Diffusion Coefficient, m^2/s
DFLOWB(LSV,LNW)	Well Boundary Dissolved Oil Diffusion Coefficient, m^2/s
DFLS(LSV,LFD)	Aqueous Surfactant Diffusion Coefficient, m^2/s
DFLSB(LSV,LBC)	Bound. Aqueous Surfactant Diffusion Coefficient, m^2/s
DFLSC	Dissolved Salt Diffusion Coefficient, m^2/s
DFNA(LSV,LFDA)	Alcohol Diffusion Coefficient in NAPL, m^2/s
DFNAB(LSV,LBCA)	
DFNW(LSV,LFDA)	Water Diffusion Coefficient in NAPL, m^2/s
DFNWB(LSV,LBCA)	
DIMF	Generalized Curvilinear Coordinate Factor
DISPL(LRC)	Longitudinal Dispersivity, m
DISPT(LRC)	Transverse Dispersivity, m
DLU(LJB)	Jacobian Coefficients Matrix for SPLIB
DNR(LUK,LFD)	Primary Variable Increments
DNRW(LUK,LNWN)	Well Primary Variable Increments
DPLD(LRCN)	Dissolved Oil Longitudinal Dispersivity, m
DPLGS(LRCS)	Surfactant Longitudinal Dispersivity, m
DPMA	Alcohol Dipole Moment, debye
DPMO	Oil Dipole Moment, debye
DPMW	Water Dipole Moment, debye
DPTD(LRCN)	Dissolved Oil Transverse Dispersivity, m
DPTRS(LRCS)	Surfactant Transverse Dispersivity, m
DT	Time Step, s
DT_CRN	Courant-Limit Time Step, s
DTAF	Time Step Acceleration Factor
DTCF	Time Step Cut Factor
DTI	Inverse Time Step, 1/s
DTI_CRN	Courant-Limit Inverse Time Step, 1/s
DTMX	Maximum Time Step, s
DTO	Old Time Step, s
DTSO	Old Time Step for 2nd Order Time Differencing, s
DX(LFX)	X-Direction Grid Cell Dimension, m
DXI(LFX+1)	X-Direction Inverse Internodal Dimension, 1/m
DXIF(LFD)	Generalized Curvilinear Coordinate Factor
DXIP(LSX)	Generalized Curvilinear Coordinate Factor
DXP(LFX+1)	X-Direction Internodal Dimension, m
DY(LFY)	Y-Direction Grid Cell Dimension, m

DYI(LFY+1)	Y-Direction Inverse Internodal Dimension, 1/m
DYP(LFY+1)	Y-Direction Internodal Dimension, m
DZ(LFZ)	Z-Direction Grid Cell Dimension, m
DZI(LFZ+1)	Z-Direction Inverse Internodal Dimension, 1/m
DZP(LFZ+1)	Z-Direction Internodal Dimension, m
DZTAF(LFD)	Generalized Curvilinear Coordinate Factor
DZTAP(LSZ)	Generalized Curvilinear Coordinate Factor
ELC_DCF(4)	Electrolyte Density Function Parameters
ELC_DUN	Electrolyte Density Function Parameter
ELC_SOL	Electrolyte Name* (64 characters)
ELC_VCF(4)	Electrolyte Viscosity Function Parameters
ELC_VUN	Electrolyte Viscosity Function Parameter
EPSL	Machine Precision, 10-14
ETAS(LFD)	Sum of Squares of ETAX, ETAY, and ETAZ
ETAX(LFD)	Grid Metric along ETA Coordinate
ETAY(LFD)	Grid Metric along ETA Coordinate
ETAZ(LFD)	Grid Metric along ETA Coordinate
FLG_EXT	Logical Flag for Creating an ".ext" File for UCODE
FLG_UNI	Logical Flag for Creating an ".uni" File for UCODE
FNPL	Plot File Name* (64 characters)
FNRD	Input File Name* (64 characters)
FNRS	Restart File Name* (64 characters)
FNSF(LSF)	Surface Flux File Name* (64 characters)
FNSR	Screen Unit Name* (64 characters)
FNWR	Output File Name* (64 characters)
FUG_TA(LP_TA,LT_TA)	CO2 Fugacity Coefficient Table
GAMMA(13,LRC)	Scaling Factor Parameters
	(1,LRC) Matrix Permeability
	(2,LRC) Matrix Porosity
	(3,LRC) Saturation/Relative Permeability Functions Parameter
	(4,LRC) Saturation/Relative Permeability Functions Parameter
	(5,LRC) Saturation/Relative Permeability Functions Parameter
	(6,LRC) Fracture Permeability
	(7,LRC) Fracture Porosity
	(8,LRC) Saturation/Relative Permeability Functions Parameter
	(9,LRC) Saturation/Relative Permeability Functions Parameter
	(10,LRC) Saturation/Relative Permeability Functions Parameter
	(11,LRC) Mualem Anisotropy Parameter
	(12,LRC) Mualem Anisotropy Parameter
	(13,LRC) Mualem Anisotropy Parameter
GJI(LFD)	Generalized Curvilinear Coordinate Factor
GRAV	Acceleration of Gravity, m/s2
GRAVX	X-Direction Acceleration of Gravity, m/s2
GRAVY	Y-Direction Acceleration of Gravity, m/s2
GRAVZ	Z-Direction Acceleration of Gravity, m/s2
H_TA(LP_TA,LT_TA)	CO2 Enthalpy Table, J/kg

HCAW	Henry's Constant Air-Water, Pa
HCMWE(LRC)	Capillary Pressure Head for Water Entry (Triple-Curve Function)
HCOW	Henry's Constant Oil-Water, Pa
HDOD	Oven-dried Capillary Head, m
HDSC(LSU,LFD)	Apparent Capillary Head at Reversal Point, m
HFW_O(LFDI)	Old Ice Enthalpy, J/kg
HFW(LSV,LFDI)	Ice Enthalpy, J/kg
HFWB(LSV,LBCI)	Boundary Ice Enthalpy, J/kg
HGA(LSV,LFDT)	Air Enthalpy, J/kg
HGAB(LSV,LBCT)	Boundary Air Enthalpy, J/kg
HGO(LSV,LFDT)	Oil Vapor Enthalpy, J/kg
HGOB(LSV,LBCT)	Boundary Oil Vapor Enthalpy, J/kg
HGW(LSV,LFDT)	Water Vapor Enthalpy, J/kg
HGWB(LSV,LBCT)	Boundary Water Vapor Enthalpy, J/kg
HKL(LFDN)	Interfacial Mass Transfer Coefficient in Aqueous Phase
HKNF(LFDN)	Interfacial Mass Transfer Coefficient in Free NAPL
HKNT(LFDN)	Interfacial Mass Transfer Coefficient in Trapped NAPL
HL_O(LFDT)	Old Aqueous Enthalpy, J/kg
HL(LSV,LFDT)	Aqueous Enthalpy, J/kg
HLB(LSV,LBCT)	Boundary Aqueous Enthalpy, J/kg
HLF(LSOLU)	Solute Half-Life, s
HN_O(LFDT)	Old NAPL Enthalpy, J/kg
HN(LSV,LFDT)	NAPL Enthalpy, J/kg
HNB(LSV,LBCT)	Boundary NAPL Enthalpy, J/kg
HSP_O(LFDS)	Old Precipitated Salt Enthalpy, J/kg
HSP(LSV,LFDS)	Precipitated Salt Enthalpy, J/kg
HSPB(LSV,LBCS)	Boundary Precipitated Salt Enthalpy, J/kg
I_OBDT(9,LOBDT)	Index for Field Observation Data
	(1,LOBDT) Observation Data Type
	(2,LOBDT) Observation Data Variable
	(3,LOBDT) Observed Data Statistical Index
	(4,LOBDT) Observed Data I Index
	(5,LOBDT) Observed Data J Index
	(6,LOBDT) Observed Data K Index
	(7,LOBDT) Observed Data I Index
	(8,LOBDT) Observed Data J Index
	(9,LOBDT) Observed Data K Index
IATM_C	Index for Cyclic Atmospheric Conditions
IBCC(LBC)	Index for Cyclic Boundary Conditions
IBCCG(LBCGR)	LUGR Subgrid Index for Cyclic Boundary Conditions
IBCD(LBC)	Index for Boundary Condition Direction
IBCDG(LBCGR)	LUGR Subgrid Index for Boundary Condition Direction
IBCM(LBC)	Index for Number of Boundary Cond. Times
IBCMG(LBCGR)	LUGR Subgrid Index for Number of Boundary Cond. Times
IBCN(LBC)	Index for Adjacent Node to Boundary Cond.
IBCNG(LBCGR)	LUGR Subgrid Index for Adjacent Node to Boundary Cond.

IBCT(LUK+LPH*LSOLU*LC+1,LBC)	Index for Boundary Condition Type
IBCTG(LUK+LPH*LSOLU*LC+1,LBCGR)	LUGR Subgrid Index for Boundary Condition Type
IBIG	Constant, 32000
ICAIR(LFDG)	Initial Condition Index for Aqueous Dissolved CO2
ICBRN(LFDS)	Index Salt Concentration Initial Condition
ICD	Number of Characters in Card Name String
ICNO	Index for Reference Node Header Line
ICNS	Index for Surface Flux Header Line
ICNV	Index for Newton-Raphson Convergence
ICODE	Index for Code's Operational Mode
ICRNT	Index for Courant Number Computation
ICS	Index for Coordinate System Type
ICSN	Number of Characters in Subroutine String
ICT(LFDC,LSOLU)	Index for Aqueous Solute Concentration Initial Cond.
ICTN(LFDC,LSOLU)	Index for NAPL Solute Concentration Initial Cond.
ID(LFD)	X-Direction Index of Node
IDF_ELC	Index for Electrolyte Density Function
IDFLT	Index Indicating Default Value Specified
IDFLTD	Index Indicating Default Value Used
IDG(LFDGR,LGRL)	LUGR Subgrid X-Direction Index of Node
IDISP	Index for Hydraulic Dispersion Calculation
IDMN(20)	Index for Interfacial Averaging Scheme (1) thermal conductivity (2) aqueous density (3) gas density (4) napl density (5) aqueous viscosity (6) gas viscosity (7) napl viscosity (8) aqueous relative permeability (9) gas relative permeability (10) napl relative permeability (11) intrinsic permeability (12) water vapor diffusion (13) oil vapor diffusion (14) dissolved air diffusion (15) dissolved oil diffusion (16) solute diffusion (17) hydraulic dispersion (18) dissolved salt diffusion (19) effective permeability
IDP(LRC)	Index for Dual-Porosity Model
IDSPD	Index for Dissolved Oil Dispersion Calcu.
IDSPS	Index for Salt/Surfactant Dispersion Calcu.
IEDL(LSOLU)	Index for Empirical Solute Diffusion Coefficient
IEDLS	Index for Empirical Salt Diffusion Coefficient

IEO	Index for Execution Option
IEPD	Number of Execution Periods
IEQA	Index for Air Equation
IEQALC	Index for Alcohol Equation
IEQC	Index for Solute Transport Equation
IEQD	Index for Dissolved-Oil Equation
IEQDA	Index for Dissolved-CO2 Equation
IEQDO	Boundary Index for Dissolved-Oil Equation
IEQO	Index for Oil Equation
IEQS	Index for Salt/Solute Equation
IEQT	Index for Energy Equation
IEQW	Index for Water Equation
IFLD	Number of Nodes in the X-Direction
IFLDG(LGRL)	LUGR Subgrid Number of Nodes in the X-Direction
IFQO	Frequency of Refer. Node Prints (Output)
IFQS	Frequency of Refer. Node Prints (Screen)
IGAMMA(8)	(1) Saturated Hydraulic Conductivity Scaling Function (2) Diffusive Porosity Scaling Function (3) van Genuchten "alpha" or Brooks/Corey "psi" Scaling Function (4) van Genuchten "n" or Brooks/Corey "lambda" Scaling Function (5) Residual Saturation Scaling Function
IGL	Index for LUGR Grid Level
IHSF	Index for Surface Flux Header
IJFLD	Number of Nodes in the X-Y plane
ILES	Index for Linear Equation Solver
ILU(LJI)	Index for Jacobian Matrix Pivoting
IM(LUKW,LFD)	Jacobian Matrix Pointer Array
IMSG	Index for Error Message
IMTC	Index for Oil Interface Mass Trans. Corr.
IMTLN(LSOLU)	Index for Interface Mass Transfer Function
INPDAT	Input File Creation Date* (132 characters)
INPTIM	Input File Creation Time* (132 characters)
IOBDEF	Unit Number for an ".ext" File for UCODE
IOBDSF	Unit Number for an ".sto" File for UCODE
IOBDUF	Unit Number for an ".uni" File for UCODE
IOM	Index for Operational Mode
IP_TA	CO2 Property Table Pressure Index
IPCL(LSOLU)	Index for Solute Solid-Aqueous Partition Option
IPCLN(LSOLU)	Index for Solute Aqueous-NAPL Partition Option
IPCSL(LRC,LSOLU)	Index for Solute Dependent Solid-Aqueous Part.
IPCSLD(LRCN)	Index for Dissolved-Oil Sol.-Aqueous Part. Opt.
IPCSLS(LRCS)	Index for Salt/Surf. Sol.-Aqueous Part. Opt.
IPH(LSU,LFD)	Index for Wetting/Drying
IPL	Unit Number of the Plot File
IPLOT(LOUPV)	Index for Plot File Variables
IPRF(LRC)	Index for Permeability Reduction Factors

IPTPS(4)	Index for Surface Tension Correlation Function
IRD	Unit Number of the Input File
IREF(LOUPV)	Index for Reference Node Variables
IRGTBL(2,LRC)	Start/Stop Indices for Gas Relative Permeability Table
IRHOO	Index for Oil Density Function
IRHOS	Index for Alcohol Density Function
IRLTBL(2,LRC)	Start/Stop Indices for Aqueous Relative Permeability Table
IRLTBLT(2,LRC,3)	Start/Stop Indices for Aqueous Relative Permeability Tensor Table
IRNTBL(2,LRC)	Start/Stop Indices for NAPL Relative Permeability Table
IRPG(LRC)	Index for Gas Relative Permeability Function
IRPL(LRC)	Index for Aqueous Relative Permeability Function
IRPLT(3,LRC)	Index for Aqueous Relative Permeability Tensor Function
IRPN(LRC)	Index for NAPL Relative Permeability Function
IRS	Unit Number of the Restart File
ISALT	Index for Salt Type
ISC	Unit Number of the Screen (I.e., Standard Output)
ISCALE(LRC)	Index for Scaling Group
ISCHR(LRC)	Index for Saturation-Capillary Pressure Function
ISF(LSF)	Unit Number of the Surface File.
ISFC(6,LSF)	Surface Flux Domain Indices (1) Bottom (2) South (3) West (4) East (5) North (6) Top
ISFD(LSF)	Index for the Surface Flux Direction
ISFF(LSF)	Index for Surface Flux File
ISFGP(LSF)	Index for Surface Flux Group
ISFT(LSF)	Index for the Surface Flux Type
ISGNO	Number of Sign. Digits in the Output File
ISGNP	Number of Sign. Digits in the Plot File
ISGNS	Number of Sign. Digits on the Screen
ISIC	Index for Saturation Initial Condition
ISKP(LRC)	Index for Skipping Residual Checking on Rock/Soil Types
ISLC(40)	Indices for Solution Control (1) Index for Solute Transport Advection/Diffusion Scheme (2) Index for Water-Vapor Gas Diffusion Model (3) Index for Tortuosity Model (4) Index for Aqueous-Phase Diffusion Coefficient Model (5) Index for Soil Freezing Conditions (6) Index for Salt Transport Advection/Diffusion Scheme (7) Index for Osmotic Pressure and Surface Tension Effects (8) Index for Dissolved-Oil Transport Advection/Diffusion Scheme (9) Index for Fluid Properties (10) Index for Dissolved-Air Transport Advection/Diffusion Scheme

	(11) Index for Time-Differencing Order (12) Index for Diffusive-Dispersive Dissolved-Oil Transport Scheme (13) Index for Particle-Displacing Bubbles (14) Index for SPLIB Summary Output (15) Index for LUGR (16) Index for Density Dependent Solute Transport (17) Index for Courant Number Control (18) Index for Restart Files (19) Index for Scaling Factors (20) Index for Inverse (UCode) (21) Index for Operational Mode Restart (22) Index for IJK Indexing (23) Index for Advective Solute Transport (24) Index for Plants (Root-Uptake Evapotranspiration)
ISLTBL(2,LRC)	Start/Stop Indices for Saturation Function Table
ISM(LRC)	Index for Saturation Function Extension
ISMALL	Constant, -3200
ISNR	Index for NAPL Residual Saturation Model
ISRDM(13,LSR)	Source Domain Indices
ISRM(LSR)	Index for Source Time
ISRT(LSR)	Index for Source Type
ISVC	Number of Coupled Unknowns
ISVF	Number of Flux Variable Indices
ISVT	Index for Solute Transport Solution
IT_TA(LP_TA)	Index for Temperature Range in the CO2 Property Table
ITERDC	Index for Ternary Phase Diagram, Binodal Curve Type
ITHK(LRC)	Index for Thermal Conductivity Model
ITOR(LRC)	Index for Tortuosity Model
IUNK	Exponent on temperature units
IUNKG	Exponent on mass units
IUNM	Exponent on length units
IUNMOL	Exponent on molar units
IUNS	Exponent on time units
IVAPO	Index for Oil Saturated Vapor Pressure Function
IVAPS	Index for Alcohol Saturated Vapor Pressure Function
IVF_ELC	Index for Electrolyte Viscosity Function
IVISA	Index for Alcohol Viscosity Function
IVISO	Index for Oil Viscosity Function
IVISS	Index for Surfactant Viscosity Function
IVR	Number of Non-Blank Characters
IVRSN	Index for Version Number
IWCC(LNW)	Index for Cyclic Well Conditions
IWL(LNWN)	Index for Well Number from Well Node Number
IWLDM(2*LNWS+4,LNW)	
IWM(LNW)	Number of Well Condition Times
IWN(LNWN)	Index for Node Number form Well Node Number

IWR	Unit Number of the Output File
IWT(LNW)	Index for Well Type
IXF(LFD)	Index for Fixed Nodes
IXG(LFDGR,LGRL)	LUGR Subgrid Matrix Location of Node
IXGB(LFDGR,LGRL)	LUGR Subgrid Index for Boundary Nodes
IXP(LFD)	Matrix Location of Node
IXREF(3)	Index for Curvilinear Coordinate Factor
IXW(LFD)	Index for Well Node Number from Node Number
IZ(LFD)	Index for Rock/Soil Type
IZG(LFDGR)	LUGR Subgrid Index for Rock/Soil Type
JD(LFD)	Y-Direction Index of Node
JDG(LFDGR,LGRL)	LUGR Subgrid Y-Direction Index of Node
JFLD	Number of Nodes in the Y-Direction
JFLDG(LGRL)	LUGR Subgrid Number of Nodes in the Y-Direction
JKFLD	Number of Nodes in the Y-Z Plane
JLU(LJJ)	Unused Banded Matrix Solver Integer Array
JM(LUK,LUK)	Jacobian Matrix Pointer
KD(LFD)	Z-Direction Index of Node
KDG(LFDGR,LGRL)	LUGR Subgrid Z-Direction Index of Node
KFLD	Number of Nodes in the Z-Direction
KFLDG(LGRL)	LUGR Subgrid Number of Nodes in the Z-Direction
KIFLD	Number of Nodes in the Z-X Plane
KLU(LJG,LJH)	Coupled-Equations Conjugate Gradient Solver Pointer Array
KLUC(LJK,LJL)	Transport-Equations Conjugate Gradient Solver Pointer Array
MADJ(9)	Adjacent Node Index Pointer
MDC	Row of Diagonal Elements for Banded Sol.
MDIM(6)	Conjugate Gradient Solver Pointer Array
MDT	Matrix Band Width
MEPD	Index for Cyclic Execution Periods
MFLX(9)	Flux Variable Index Pointer
MKC	Number of Coupled-Equations Unkowns
MKT	Number of Transport-Equations Unknowns
MLC	Lower Half Band Width
MLT	Matrix Lower Half Band Width
MLU(LJB)	Coupled-Equations Pointer for SPLIB Solver
MLUC(LJM)	Transport-Equations Pointer for SPLIB Solver
MNEG(9)	Flux Variable Index Pointer
MNEGB(5)	Boundary Index Pointer
MNOD(9)	Flux Variable Index Pointer
MPOS(9)	Flux Variable Index Pointer
MPOSB(5)	Boundary Index Pointer
MUC	Upper Half Band Width
MUT	Matrix Upper Half Band Width
MXSTEP	Maximum Number of Time Steps
N_CRN(LSOLU)	Courant Number Indexed on Solute
N_DB	Node Number for Error Messages

NATM_T	Number of Atmospheric Condition Table Entries
NBC	Number of Boundary Conditions
NBCG	Number of Boundary Conditions at the Local Grid Level
NCHEM(LSOLU)	Number of Chemical Reactions
ND(LFX,LFY,LFZ)	Node Number of I, J, K Indices
NDG(LFXGR,LFYGR,LFZGR,LGRL)	Node Number of I, J, K Indices at the Local Grid Level
NDIM	Number of Active Dimensions
NDREF(LREF)	Reference Node Type
NEPD	Number of Execution Periods
NFLD	Number of Active Field Nodes
NFLDG(LGRL)	Number of Active Field Nodes at the Local Grid Level
NGC(LFDGR,LGRL)	Node Pointer from Finer Grid to Next Coarser Grid Level
NGF(LFDGR,LGRL)	Node Pointer from Coarser Grid to Next Finer Grid Level
NGL	Number of Grid Levels
NITER	Number of Iterations
NITR(LGRL)	Number of Iterations at the Local Grid Level
NLU(LJC)	Pointer for SPLIB Solver
NLUC(LJN)	Pointer for SPLIB Solver for Solutes
NOBDP	Observed Data Point Number
NOBDS(LOBDT)	Observed Data Sample Pointer
NOBDT	Number of Observed Data Types
NOTES(LNOTES)	Number of Lines of Simulation Notes
NPHAZ(LSU,LFD)	Phase Condition
NPHAZW(LSU,LNWN)	Phase Condition for Coupled Well Nodes
NPLANT	Number of Plants
NPRTM	Number of Print Times
NQA	Number of Carbon Atoms in Alcohol
NQO	Number of Carbon Atoms in Oil
NREF	Number of Reference Nodes
NRIM(LEPD)	Max. Number of Newton Raphson Iterations
NRIMX	Max. Number of Newton Raphson Iterations
NROCK	Number of Rock/Soil Types
NRST	Restart Time Step Number
NSCALE	Number of Scaling Groups
NSD(LUKW)	Node of Maximum Residual
NSF	Number of Surface Flux Domains
NSFGP	Number of Named Surface Flux Files
NSL_ELC	Number of Electrolyte Solutes
NSOLU	Number of Transported Solutes
NSR	Number of Source Domains
NSTEP	Number of Time Steps
NSX(LFD)	X-Direction Lower Surface Number of Node
NSY(LFD)	Y-Direction Lower Surface Number of Node
NSZ(LFD)	Z-Direction Lower Surface Number of Node
NSZW(LNWN)	Z-Direction Lower Surface Number of Coupled Well Node
NTBL	Number of Tables

NTSR	Num. of Successive Time Step Reductions
NVPLOT	Number of Plot File Variables
NVREF	Number of Reference Node Variables
NWLN	Number of Coupled Well Nodes
NWLS	Number of Coupled Wells
NXP	Number of Inactive Nodes
OEC(LSV,LFDS)	Osmotic Efficiency Coefficient
OECB(LSV,LBCS)	Boundary Osmotic Efficiency Coefficient
ONE	Constant, 1
P_TA(LP_TA)	Pressure Array for CO2 Property Data Table
PAF(LCMP)	Component Pitzer Acentric Factor
PAFA	Alcohol Pitzer Acentric Factor
PAFO	Oil Pitzer Acentric Factor
PAFS	Surfactant Pitzer Acentric Factor
PAFW	Water Pitzer Acentric Factor
PATM	Atmospheric Pressure, Pa
PC(LCMP)	Component Critical Pressure, Pa
PCCVA	Alcohol Pure Comp. Characteristic Volume, L/mol
PCCVO	Oil Pure Comp. Characteristic Volume, L/mol
PCGL(5,LSOLU)	Gas-Aqueous Partition Coefficient, m^3/m^3
PCLN(5,LSOLU)	Aqueous-NAPL Partition Coefficient, m^3/m^3
PCRA	Alcohol Critical Pressure, Pa
PCRO	Oil Critical Pressure, Pa
PCRS	Surfactant Critical Pressure, Pa
PCRW	Water Critical Pressure, Pa
PCSL(5,LRC,LSOLU)	Solid-Aqueous Partition Coefficient, m^3/kg
PCSLD(2,LRCN)	Dissolved-Oil Solid-Aqueous Part. Coefficient, m^3/kg
PCSLS(2,LRCS)	Surfactant Solid-Aqueous Part. Coefficient, m^3/kg
PERM(9,LRC)	Intrinsic Permeability, m^2
PERMRF(LSV,LFD)	Permeability Reduction Factor
PFW(LSV,LFD)	Ice Pressure, Pa
PFWB(LSV,LBC)	Boundary Ice Pressure, Pa
PG(LSV,LFD)	Gas Pressure, Pa
PGB(LSV,LBC)	Boundary Gas Pressure, Pa
PGW(LSV,LSR)	Well Gas Pressure, Pa
PHDL(2,LBC)	Aqueous Ponding Height, m
PHDN(2,LBC)	NAPL Ponding Height, m
PI	Constant, 3.1415926536
PL(LSV,LFD)	Aqueous Pressure, Pa
PLANT(LPLANT)	Plant Name
PLB(LSV,LBC)	Boundary Aqueous Pressure, Pa
PLNTP(25,LPLANT)	Plant Parameters
PLWB(LSV,LSR)	Well Bottom Pressure, Pa
PMDD(LRC)	Mean Particle Diameter, m
PMN	Minimum Gas Pressure, Pa
PMX	Maximum Gas Pressure, Pa

PN(LSV,LFD)	NAPL Pressure, Pa
PNB(LSV,LBC)	Boundary NAPL Pressure, Pa
POR(4,LRC)	Porosity
	(1) Total Porosity
	(2) Diffusive Porosity
	(3) Fracture Total Porosity
	(4) Diffusive Porosity
PORD_O(LFD)	Old Diffusive Porosity
PORD(LSV,LFD)	Diffusive Porosity
PORDB(LSV,LBC)	Boundary Diffusive Porosity
PORT_O(LFDT)	Old Total Porosity
PORT(LSV,LFD)	Total Porosity
PORTB(LSV,LBC)	Boundary Total Porosity
POSB(LSV,LBC)	Boundary Osmotic Pressure, Pa
POSM(LSV,LFD)	Osmotice Pressure, Pa
PRTM(LPTM)	Plot File Print Time
PSO(LSV,LFD)	Oil Saturated Vapor Pressure, Pa
PSOB(LSV,LBC)	Boundary Oil Saturated Vapor Pressure, Pa
PSW(LSV,LFD)	Water Saturated Vapor Pressure, Pa
PSWB(LSV,LBC)	Water Saturated Vapor Pressure, Pa
PTPS(5)	Surface Tension Function Parameters
PVA(LSV,LFD)	Air Vapor Pressure, Pa
PVAB(LSV,LBC)	Boundary Air Vapor Pressure, Pa
PVO(LSV,LFD)	Oil Vapor Pressure, Pa
PVOB(LSV,LBC)	Boundary Oil Vapor Pressure, Pa
PVW(LSV,LFD)	Water Vapor Pressure, Pa
PVWB(LSV,LBC)	Boundary Water Vapor Pressure, Pa
PW(LSV,LNWN)	Coupled Well Total Pressure, Pa
PWB(LSV,LNW)	Coupled Well Bottom Pressure, Pa
PWGW(LSV,LNWN)	Coupled Well Gas Pressure, Pa
PWLW(LSV,LNWN)	Coupled Well Aqueous Pressure, Pa
PWNW(LSV,LNWN)	Coupled Well NAPL Pressure, Pa
QL_W(4,LNW)	Coupled Well Aqueous Volumetric Flow Rate, m^3/s
QLW(3,LSR)	Well Aqueous Volumetric Flow Rate, m^3/s
QN_W(4,LNW)	Coupled Well NAPL Volumetric Flow Rate, m^3/s
QNW(3,LSR)	Well NAPL Volumetric Flow Rate, m^3/s
QT_W(4,LNW)	Coupled Well Total-Liquid Volumetric Flow Rate, m^3/s
QTW(3,LSR)	Well Total-Liquid Volumetric Flow Rate, m^3/s
R_OBDS(2,LOBDS,LOBDT)	Observed Data Time and Value
R_OBDT(6,LOBDS)	Observed Data Location and Weighting Factors
RCA	Air Gas Constant, 287.0 J/kg K
RCHDF(LSOLU,LSOLU,LCHEM)	Chemical Reaction Decay Factor
RCHDFL(LSOLU,LSOLU,LCHEM)	Chemical Reaction Decay Factor in Aqueous
RCHDFN(LSOLU,LSOLU,LCHEM)	Chemical Reaction Decay Factor in NAPL
RCO	Oil Gas Constant, J/kg K
RCS	Surfactant Gas Constant, J/kg K

RCU	Universal Gas Constant, 8314.34 J/kg K
RCW	Water Gas Constant, 461.52 J/kg K
RHLF(LSOLU,LCHEM)	First-Order Reaction Rate Constant, s
RHLFL(LSOLU,LCHEM)	First-Order Reaction Rate Constant in Aqueous Phase, s
RHLFN(LSOLU,LCHEM)	First-Order Reaction Rate Constant in NAPL, s
RHO_TA(LP_TA,LT_TA)	CO2 Property Table Values for Density, kg/m^3
RHOFW_O(LFDI)	Old Ice Density, kg/m^3
RHOFW(LSV,LFDI)	Ice Density, kg/m^3
RHOFWB(LSV,LBCI)	Boundary Ice Density, kg/m^3
RHOG_O(LFDG)	Old Gas Density, kg/m^3
RHOG(LSV,LFD)	Gas Density, kg/m^3
RHOGB(LSV,LBC)	Boundary Gas Density, kg/m^3
RHOGW_O(LNWN)	Old Well Gas Density, kg/m^3
RHOGW(LSV,LNWN)	Well Gas Density, kg/m^3
RHOGWB(LSV,LNW)	Well Boundary Gas Density, kg/m^3
RHOL_O(LFD)	Old Aqueous Density, kg/m^3
RHOL(LSV,LFD)	Aqueous Density, kg/m^3
RHOLB(LSV,LBC)	Boundary Aqueous Density, kg/m^3
RHOLI	Density for Invariant Fluids
RHOLW_O(LNWN)	Old Well Aqueous Density, kg/m^3
RHOLW(LSV,LNWN)	Well Aqueous Density, kg/m^3
RHOLWB(LSV,LNW)	Well Boundary Aqueous Density, kg/m^3
RHOMG(LSV,LFD)	Gas Molar Density, mol/m^3
RHOMGB(LSV,LBC)	Boundary Gas Molar Density, mol/m^3
RHOML(LSV,LFD)	Aqueous Molar Density, mol/m^3
RHOMLB(LSV,LBC)	Boundary Aqueous Molar Density, mol/m^3
RHOMN(LSV,LFDA)	NAPL Molar Density, kmol/m^3
RHOMNB(LSV,LBCA)	Boundary NAPL Molar Density, kmol/m^3
RHON_O(LFDN)	Old NAPL Density, kg/m^3
RHON(LSV,LFD)	NAPL Density, kg/m^3
RHONB(LSV,LBC)	Boundary NAPL Density, kg/m^3
RHONW_O(LNWN)	Old Well NAPL Density, kg/m^3
RHONW(LSV,LNWN)	Well NAPL Density, kg/m^3
RHONWB(LSV,LNW)	Well Boundary NAPL Density, kg/m^3
RHORA	Alcohol Liquid Reference Density, kg/m^3
RHORG	Gas Reference Density, kg/m^3
RHORL	Aqueous Reference Density, kg/m^3
RHORN	NAPL Reference Density, kg/m^3
RHORO	Oil Liquid Reference Density, kg/m^3
RHOS(LRC)	Rock/Soil Particle Density, kg/m^3
RHOSP_O(LFDS)	Old Precipitated Salt Density, kg/m^3
RHOSP(LSV,LFDS)	Precipitated Salt Density, kg/m^3
RHOSPB(LSV,LBCS)	Boundary Precipitated Salt Density, kg/m^3
RKG(LSV,LFDG)	Gas Relative Permeability
RKGB(LSV,LBCG)	Boundary Gas Relative Permeability
RKL(3,LSV,LFD)	Aqueous Relative Permeability

	(1) X-Direction
	(2) Y-Direction
	(3) Z-Direction
RKLB(3,LSV,LBC)	Boundary Aqueous Relative Permeability
	(1) X-Direction
	(2) Y-Direction
	(3) Z-Direction
RKN(LSV,LFDN)	NAPL Relative Permeability
RKNB(LSV,LBCN)	Boundary NAPL Relative Permeability
RLMSG	Real Number Component of Error Message
RLXF	Relaxation Factor
ROCK(LRC)	Rock/Soil Name* (64 characters)
RP(LFX)	Radial Coordinate Radius, m
RPG(LFXGR,LGRL)	Radial Coordinate Radius at the Local Grid Level, m
RPGC(LRPGC,LRC)	Gas Relative Permeability Function Coefficients
RPLC(LRPLC,LRC)	Aqueous Relative Permeability Function Coefficients
RPLT(4,LRPL,LRC)	Aqueous Relative Permeability Function Coefficients
RPNC(LRPNC,LRC)	NAPL Relative Permeability Function Coefficients
RSD(LUKW)	Maximum Residual of Equation
RSDL(LUKW,LFD)	Residual
RSDM(LEPD)	Maximum Residual of Execution Period
RSDMX	Maximum Residual
S_O(LFDS)	Old Salt Concentration, kg/m^3 node volume
S(LSV,LFD)	Salt Concentration, kg/m^3 node volume
SATAC(4)	Alcohol Saturated Vapor Pressure Function Coefficients
SATOC(4)	Oil Saturated Vapor Pressure Function Coefficients
SB(LSV,LBC)	Boundary Salt Concentration, kg/m^3 node volume
SCALNM(LRC)	Scaling Group Name
SCHR(LSCHR,LRC)	Soil Moisture Retention Charac. Coefficient
SDCL(3,LRC,LSOLU)	Solute Aqueous Molecular Diffusion Coefficient, m^2/s
SDCLS(3,LRC)	Salt/Surf. Aqueous Molecular Diffusion Coefficient, m^2/s
SDPF(LFD)	Fracture Saturation (Dual Porosity)
SDPFB(LBC)	Bound. Fracture Saturation (Dual Porosity)
SDPM(LFD)	Matrix Saturation (Dual Porosity)
SDPMB(LBC)	Bound. Matrix Saturation (Dual Porosity)
SF(2,LSF)	Surface Flux Rate and Integral
SFCSF(4)	Surfactant Critical Micellar Concentration, kg/m^3
SFW_O(LFDI)	Old Ice Saturation
SFW(LSV,LFD)	Ice Saturation
SFWB(LSV,LBC)	Boundary Ice Saturation
SG_O(LFDG)	Old Gas Saturation
SG(LSV,LFD)	Gas Saturation
SGB(LSV,LBC)	Boundary Gas Saturation
SGT(LSV,LFD)	Trapped Gas Saturation
SGTL(LFD)	Aqueous Trapped Gas Saturation
SGTN(LFD)	NAPL Trapped Gas Saturation

SL_O(LFD)	Old Aqueous Saturation
SL(LSV,LFD)	Aqueous Saturation
SLB(LSV,LBC)	Boundary Aqueous Saturation
SLSC(LSU,LFD)	Aqueous Scanning Path Saturation
SLW_O(LNWN)	Old Well Aqueous Saturation
SLW(LSV,LNWN)	Well Aqueous Saturation
SMALL	Constant, 1-20
SMDEF(LRC,LSOLU)	Macrodispersivity Enhancement Factor
SMDG(LSOLU)	Solute Gas Molecular Diffusion Coefficient, m^2/s
SMDL(LSOLU)	Solute Aqueous Molecular Diffusion Coefficient, m^2/s
SMDLS	Salt Aqueous Molecular Diffusion Coefficient, m^2/s
SMDN(LSOLU)	Solute NAPL Molecular Diffusion Coefficient, m^2/s
SN_O(LFDN)	Old NAPL Saturation
SN(LSV,LFD)	NAPL Saturation
SNB(LSV,LBC)	Boundary NAPL Saturation
SNR(LSV,LFD)	Residual NAPL Saturation
SNT(LSV,LFD)	Entrapped NAPL Saturation
SNW_O(LNWN)	Old Well NAPL Saturation
SNW(LSV,LNWN)	Well NAPL Saturation
SOLML(LSOLU)	Solute Aqueous Solubility Limit, 1/m^3
SOLUT(LSOLU)	Solute Name* (64 characters)
SPCMON(LFD)	Scaled Nodal Space Monitor for Local Grid Refinement
SPCTOL(LGRL)	Scaled Nodal Space Tolerance for Local Grid Refinement
SRC(8+LSOLU,LSTM,LSR)	Source Quantity
SRCA(LSV,LFDG)	Air Mass Source, kg/s
SRCD(LSV,LFDD)	Dissolved-Oil Mass Source, kg/s
SRCIA(LFDG)	Air Mass Source Integral, kg
SRCIC(LFDC,LSOLU)	Solute Source Integral, 1/s
SRCID(LFDD)	Dissolved-Oil Mass Source Integral, kg
SRCIO(LFDN)	Oil Mass Source Integral, kg
SRCIS(LFDS)	Salt/Surfactant Mass Source Integral, kg
SRCIT(LFDT)	Energy Source Integral, J
SRCIW(LFD)	Water Mass Source Integral, kg
SRCO_W(LSV,LNWN)	Well Oil Mass Source, kg/s
SRCO(LSV,LFDN)	Oil Mass Source, kg/s
SRCS(LSV,LFDS)	Salt/Surfactant Mass Source, kg/s
SRCT(LSV,LFDT)	Energy Source, W
SRCW_W(LSV,LNWN)	Well Water Mass Source, kg/s
SRCW(LSV,LFD)	Water Mass Source, kg/s
STW_O(LNWN)	Old Well Total-Liquid Saturation
STW(LSV,LNWN)	Well Total-Liquid Saturation
SUBNM	Subroutine Path* (512 characters)
SUBNMX	Subroutine Path* (512 characters)
SUFA	Surface Tension of Pure Alcohol @ 25 C, N/m
SUFO	Surface Tension of Pure Oil @ 25 C, N/m
SUFW	Surface Tension of Pure Water @ 25 C, N/m

SUGL(LSV,LFDA)	Gas-Aqueous Interfacial Tension, N/m
SUGN(LSV,LFDA)	Gas-NAPL Interfacial Tension, N/m
SUNL(LSV,LFDA)	NAPL-Aqueous Interfacial Tension, N/m
T_O(LFDT)	Old Temperature, C
T_TA(LP_TA,LT_TA)	Temperature Array for CO2 Property Data Table
T(LSV,LFD)	Temperature, C
TABS	Absolute Temperature, 273.15 K
TB(LSV,LBC)	Boundary Temperature, C
TBA	Air Normal Boiling Temperature, K
TBLDDX(LTBL)	Cubic Spline Interpolating Function Table
TBLDDY(LTBL)	Cubic Spline Interpolating Function Table
TBLX(LTBL)	Independent Variable of a Tabular Function
TBLY(LTBL)	Dependent Variable of a Tabular Function
TBO	Oil Normal Boiling Temperature, K
TBW	Water Normal Boiling Temperature, K
TC(LCMP)	Component Critical Temperature, K
TCRA	Alcohol Critical Temperature, K
TCRO	Oil Critical Temperature, K
TCRS	Surfactant Critical Temperature, K
TCRW	Water Critical Temperature, K
TDRA	Alcohol Density Reference Temperature, C
TDRO	Oil Density Reference Temperature, C
TENTH	Constant, 0.1
TERDC(11)	Water-Alcohol-Oil Ternary Equilibrium Coefficients
	(1) Binodal Curve Constant Ah
	(2) Binodal Curve Constant Bh
	(3) Binodal Curve Constant Eh
	(4) Binodal Curve Constant Fh
	(8) Solubility of Water in Alcohol-Free Oil
	(10) Solubility of Oil in Alcohol-Free Water
TFPA	Alcohol Freezing Point Temperature, C
TFPO	Oil Freezing Point Temperature, C
THKFW(LSV,LFDI)	Ice Thermal Conductivity, W/m K
THKFWB(LSV,LBCI)	Boundary Ice Thermal Conductivity, W/m K
THKG(LSV,LFDT)	Gas Thermal Conductivity, W/m K
THKGB(LSV,LBCT)	Boundary Gas Thermal Conductivity, W/m K
THKL(LSV,LFDT)	Aqueous Thermal Conductivity, W/m K
THKLB(LSV,LBCT)	Boundary Aqueous Thermal Conductivity, W/m K
THKN(LSV,LFDT)	NAPL Thermal Conductivity, W/m K
THKNB(LSV,LBCT)	Boundary NAPL Therm. Conductivity, W/m K
THKRA	Reference Alcohol Thermal Conductivity, W/m K
THKRW	Reference Water Thermal Conductivity, W/m K
THKS(9,LRC)	Effective Thermal Conductivity, W/m K
TITLE	Simulation Title* (132 characters)
TM	Time, s
TM_CRN	Cournant Number Limiting Time Step, s

TMMX	Maximum Time, s
TMN	Minimum Temperature, C
TMOB	Observed Date Output Time, s
TMPA(LEPD)	Time Step Acceleration Factor
TMPC(LEPD)	Time Step Cut Factor
TMPD(LEPD)	Execution Period Initial Time Step, s
TMPE(LEPD)	Execution Period Stop Time, s
TMPR	Plot/Restart File Print Time, s
TMPS(LEPD)	Execution Period Start Time, s
TMPX(LEPD)	Execution Period Maximum Time Step, s
TMX	Maximum Temperature, C
TOLS	Adaptive Grid Spatial Tolerance
TOR(6,LRC)	Tortuosity Function Constants
TORG(LSV,LFDG)	Gas Tortuosity
TORGB(LSV,LBCG)	Boundary Gas Tortuosity
TORL(LSV,LFD)	Aqueous Tortuosity
TORLB(LSV,LBC)	Boundary Aqueous Tortuosity
TORN(LSV,LFDN)	NAPL Tortuosity
TORNB(LSV,LBCN)	Boundary NAPL Tortuosity
TRPGL(LSU,LFD)	Gas-Aqueous Trapping Number
TRPNL(LSU,LFD)	NAPL-Aqueous Trapping Number
TSPRF	Solute Diffusion Reference Temperature, C
U_TA(LP_TA,LT_TA)	CO2 Property Table Values for Internal Energy, J/kg
UC(LSXC,LSOLU)	X-Direction Solute Flux, 1/m^2 s
UCN(LSXC,LSOLU)	X-Direction Solute Flux in NAPL, 1/m^2 s
UDGO(LSFV,LSX)	X-Direction Oil Vapor Diffusive Flux, kg/m^2 s
UDGW_W(LSFV,LNWN)	Well Radial Water Vapor Diffusive Flux, kg/m^2 s
UDGW(LSFV,LSX)	X-Direction Water Vapor Diffusive Flux, kg/m^2 s
UDLA_W(LSFV,LNWN)	Well Radial Dissolved-Air Diffusive Flux, kg/m^2 s
UDLA(LSFV,LSX)	X-Direction Dissolved-Air Diffusive Flux, kg/m^2 s
UDLO_W(LSFV,LNWN)	Well Radial Dissolved-Oil Diffusive Flux, kg/m^2 s
UDLO(LSFV,LSX)	X-Direction Dissolved-Oil Diffusive Flux, kg/m^2 s
UDO(LSFV,LSXN)	X-Direction Oil Diffusive Flux, kg/m^2 s
UDS(LSFV,LSX)	X-Direction Salt/Surfactant Diffusion Flux, kg/m^2 s
UEG_O(LFDT)	Old Gas Internal Energy, J/kg
UEG(LSV,LFDT)	Gas Internal Energy, J/kg
UEGB(LSV,LBCT)	Boundary Gas Internal Energy, J/kg
UG_W(LSFV,LNWN)	Well X-Direction Darcy Gas Flux, m/s
UG(LSFV,LSX)	X-Direction Darcy Gas Flux, m/s
UL_W(LSFV,LNWN)	Well X-Direction Darcy Aqueous Flux, m/s
UL(LSFV,LSX)	X-Direction Darcy Aqueous Flux, m/s
UN_W(LSFV,LSZW)	Well X-Direction Darcy NAPL Flux, m/s
UN(LSFV,LSXN)	X-Direction Darcy NAPL Flux, m/s
UNAR	Units for Radiants
UNLN	Output/Reference Node Length Units* (64 characters)
UNPLOT(LOUPV)	Plot File Variable Index

UNREF(LOUPV)	Reference Node Variable Index
UNSF(2,LSF)	Surface Flux Rate and Integral Units* (64 characters)
UNTM	Output/Reference Node Time Units* (64 characters)
UQ(LSFV,LSXT)	X-Direction Energy Flux, W/m^2
US(LSFV,LSXS)	X-Direction Salt/Surfactant Flux, kg/m^2 s
USER	User Name* (64 characters)
VARB	Input Variable Name* (64 characters)
VC(LSYC,LSOLU)	Y-Direction Solute Flux, 1/m^2 s
VCN(LSYC,LSOLU)	Y-Direction Solute Flux in NAPL, 1/m^2 s
VCRA	Air Critical Molar Volume, cm^3/mol
VCRO	Oil Critical Molar Volume, cm^3/mol
VCRS	Surfactant Critical Molar Volume, cm^3/mol
VCRW	Water Critical Molar Volume, cm^3/mol
VDGO(LSFV,LSY)	Y-Direction Oil Vapor Diffusive Flux, kg/m^2 s
VDGW(LSFV,LSY)	Y-Direction Water Vapor Diffusive Flux, kg/m^2 s
VDLA(LSFV,LSY)	Y-Direction Dissolved-Air Diffusive Flux, kg/m^2 s
VDLO(LSFV,LSY)	Y-Direction Dissolved-Oil Diffusive Flux, kg/m^2 s
VDO(LSFV,LSYN)	Y-Direction Oil Diffusive Flux, kg/m^2 s
VDS(LSFV,LSY)	Y-Direction Salt/Surfactant Diffusion Flux, kg/m^2 s
VG(LSFV,LSY)	Y-Direction Gas Darcy Flux, m/s
VISCA(4)	Alcohol Viscosity Function Constants
VISCO(4)	Oil Viscosity Function Constants
VISCS(4)	Surfactant Viscosity Function Constants
VISG(LSV,LFDG)	Gas Viscosity, Pa s
VISGB(LSV,LBCG)	Boundary Gas Viscosity, Pa s
VISGW(LSV,LNWN)	Well Gas Viscosity, Pa s
VISGWB(LSV,LNW)	Well Boundary Gas Viscosity, Pa s
VISL(LSV,LFD)	Aqueous Viscosity, Pa s
VISLB(LSV,LBC)	Boundary Aqueous Viscosity, Pa s
VISLI	Viscosity of Invariant Nodes
VISLW(LSV,LNWN)	Well Aqueous Viscosity, Pa s
VISLWB(LSV,LNW)	Well Boundary Aqueous Viscosity, Pa s
VISN(LSV,LFDN)	NAPL Viscosity, Pa s
VISNB(LSV,LBCN)	Boundary NAPL Viscosity, Pa s
VISNW(LSV,LNWN)	Well NAPL Viscosity, Pa s
VISNWB(LSV,LNW)	Well Boundary NAPL Viscosity, Pa s
VISRG	Gas Reference Viscosity, Pa s
VISRL	Aqueous Reference Viscosity, Pa s
VISRN	NAPL Reference Viscosity, Pa s
VL(LSFV,LSY)	Y-Direction Water Darcy Flux, m/s
VMC(LCMP)	Component Critical Molar Volume, cm^3/mol
VN(LSFV,LSYN)	NAPL Darcy Flux, m/s
VOL(LFD)	Node Volume, m^3
VQ(LSFV,LSYT)	Y-Direction Energy Flux, W/m^2
VS(LSFV,LSYS)	Y-Direction Salt/Surfactant Flux, kg/m^2 s
WBR(LNW)	Well Bore Radius, m

WBRS(LNW)	Well Bore Radius for Storage, m
WC(LSZC,LSOLU)	Z-Direction Solute Flux, 1/m^2 s
WCN(LSZC,LSOLU)	Z-Direction Solute Flux in NAPL, 1/m^2 s
WDGO(LSFV,LSZ)	Z-Direction Oil Vapor Diffusive Flux, kg/m^2 s
WDGW_W(LSFV,LSZW)	Well Z-Direction Water Vapor Diffusive Flux, kg/m^2 s
WDGW(LSFV,LSZ)	Z-Direction Water Vapor Diffusive Flux, kg/m^2 s
WDLA_W(LSFV,LSZW)	Well Z-Direction Dissolved-Air Diffusive Flux, kg/m^2 s
WDLA(LSFV,LSZ)	Z-Direction Dissolved-Air Diffusive Flux, kg/m^2 s
WDLO_W(LSFV,LSZW)	Well Z-Direction Dissolved-Oil Diffusive Flux, kg/m^2 s
WDLO(LSFV,LSZ)	Z-Direction Dissolved-Oil Diffusive Flux, kg/m^2 s
WDO(LSFV,LSZN)	Z-Direction Oil Diffusive Flux, kg/m^2 s
WDS(LSFV,LSZ)	Z-Direction Salt/Surfactant Diffusion Flux, kg/m^2 s
WFMN(20)	Interfacial Average Weighting Factor
WG_W(LSFV,LSZW)	Well Z-Direction Gas Darcy Flux, m/s
WG(LSFV,LSZ)	Z-Direction Gas Darcy Flux, m/s
WHBTA	Alcohol HBT Accentric Factor
WHBTO	Oil HBT Accentric Factor
WHP(LNW)	Initial Well-Head Pressure, Pa
WIDA(LNW)	Initial Well Dissolved-Air Saturation
WIDO(LNW)	Initial Well Dissolved-Oil Saturation
WL_W(LSFV,LSZW)	Well Z-Direction Aqueous Darcy Flux, m/s
WL(LSFV,LSZ)	Z-Direction Aqueous Darcy Flux, m/s
WLVR(LNWV,LNWT,LNW)	Well Head Parameters (1) Well Time, s (2) Well-Head Pressure, Pa (3) Well-Head Relative Humidity
WN_W(LSFV,LSZW)	Well Z-Direction NAPL Darcy Flux, m/s
WN(LSFV,LSZN)	Z-Direction NAPL Darcy Flux, m/s
WQ(LSFV,LSZT)	Z-Direction Energy Flux, W/m^2
WRH(LNW)	Initial Well Relative Humidity
WS(LSFV,LSZS)	Z-Direction Salt/Surfactant Flux, kg/m^2 s
WTM(LCMP)	Component Molecular Weight, kg/kgmol
WTMA	Air Molecular Weight, kg/kgmol
WTMO	Oil Molecular Weight, kg/kgmol
WTMS	Surfactant Molecular Weight, kg/kgmol
WTMW	Water Molecular Weight, kg/kgmol
WWD(LNW)	Well Total-Liquid Depth, m
WWDL(LNW)	Well Aqueous Depth, m
WWDN(LNW)	Well NAPL Depth, m
X(LFX+1,LFY+1,LFZ+1)	X-Direction Lower Node Surface Position, m
XETA(LFD)	Spatial Differential of X Along Eta Coordinate
XGA_O(LFDG)	Old Air Mass Fraction in Gas
XGA(LSV,LFD)	Air Mass Fraction in Gas
XGAB(LSV,LBC)	Boundary Air Mass Fraction in Gas
XGAW_O(LNWN)	Old Well Air Mass Fraction in Gas
XGAW(LSV,LNWN)	Well Air Mass Fraction in Gas

XGAWB(LSV,LNW)	Well Boundary Air Mass Fraction in Gas
XGO_O(LFDG)	Old Oil Mass Fraction in Gas
XGO(LSV,LFD)	Oil Mass Fraction in Gas
XGOB(LSV,LBC)	Boundary Oil Mass Fraction in Gas
XGW_O(LFDG)	Old Well Oil Mass Fraction in Gas
XGW(LSV,LFD)	Water Mass Fraction in Gas
XGWB(LSV,LBC)	Boundary Water Mass Fraction in Gas
XGWW_O(LNWN)	Old Well Water Mass Fraction in Gas
XGWW(LSV,LNWN)	Well Water Mass Fraction in Gas
XGWWB(LSV,LNW)	Well Boundar Water Mass Fraction in Gas
XIS(LFD)	Sum of Squares of XIX, XIY, and XIZ
XIX(LFD)	X Grid Metric Along XI Coordinate
XIY(LFD)	Y Grid Metric Along XI Coordinate
XIZ(LFD)	Z Grid Metric Along XI Coordinate
XLA_O(LFD)	Old Dissolved-Air Mass Fraction in Aqueous
XLA(LSV,LFD)	Dissolved-Air Mass Fraction in Aqueous
XLAB(LSV,LBC)	Boundary Dissolved-Air Mass Fraction in Aqueous
XLAW_O(LNWN)	Well Old Dissolved-Air Mass Fraction in Aqueous
XLAW(LSV,LNWN)	Well Dissolved-Air Mass Fraction in Aqueous
XLAWB(LSV,LNW)	Well Boundary Dissolved-Air Mass Fractin in Aqueous
XLO_O(LFD)	Oil Dissolved-Oil Mass Fraction in Aqueous
XLO(LSV,LFD)	Dissolved-Oil Mass Fraction in Aqueous
XLOB(LSV,LBC)	Boundary Dissolved-Oil Mass Fraction in Aqueous
XLOW_O(LNWN)	Old Dissolved-Oil Mass Fraction in Aqueous
XLOW(LSV,LNWN)	Well Dissolved-Oil Mass Fraction in Aqueous
XLOWB(LSV,LNW)	Well Boundary Dissolved-Oil Mass Fraction in Aqueous
XLS_O(LFDS)	Old Dissolved-Salt Mass Fraction in Aqueous
XLS(LSV,LFDS)	Dissolved-Salt Mass Fraction in Aqueous
XLSB(LSV,LBCS)	Boundary Dissolved-Salt Mass Fraction in Aqueous
XLW_O(LFD)	Old Well Dissolved-Salt Mass Fraction in Aqueous
XLW(LSV,LFD)	Water Mass Fraction in Aqueous
XLWB(LSV,LBC)	Boundary Water Mass Fraction in Aqueous
XLWW_O(LNWN)	Old Well Water Mass Fraction in Aqueous
XLWW(LSV,LNWN)	Well Water Mass Fraction in Aqueous
XLWWB(LSV,LNW)	Well Boundary Water Mass Fraction in Aqueous
XMGA(LSV,LFD)	Air Mole Fraction in Gas
XMGAB(LSV,LBC)	Boundary Air Mole Fraction in Gas
XMGO(LSV,LFD)	Oil Mole Fraction in Gas
XMGOB(LSV,LBC)	Boundary Oil Mole Fraction in Gas
XMGW(LSV,LFD)	Water Mole Fraction in Gas
XMGWB(LSV,LBC)	Boundary Oil Mole Fraction in Gas
XMGWW(LSV,LNWN)	Well Water-Vapor Mole Fraction in Gas
XMGWWB(LSV,LNW)	Well Boundary Water-Vapor Mole Fraction in Gas
XMLA(LSV,LFD)	Air Mole Fraction in Aqueous
XMLAB(LSV,LBC)	Boundary Air Mole Fraction in Aqueous
XMLAW(LSV,LNWN)	Well Dissolved-Air Mole Fraction in Aqueous

XMLAWB(LSV,LNW)	Well Boundary Dissolved-Air Mole Fraction in Aqueous
XMLO(LSV,LFD)	Oil Mole Fraction in Aqueous
XMLOB(LSV,LBC)	Boundary Oil Mole Fraction in Aqueous
XMLOW(LSV,LNWN)	Well Dissolved-Oil Mole Fraction in Aqueous
XMLOWB(LSV,LNW)	Well Boundary Dissolved-Oil Mole Fraction in Aqueous
XMLS(LSV,LFDS)	Salt/Surfactant Mole Fraction in Aqueous
XMLSB(LSV,LBCS)	Bound. Salt/Surfactant Mole Frac. in Aqueous
XMLW(LSV,LFD)	Water Mole Fraction in Aqueous
XMLWB(LSV,LBC)	Bound. Water Mole Fraction in Aqueous
XMNA(LSV,LFDA)	Alcohol Mole Fraction in NAPL
XMNAB(LSV,LBCA)	Boundary Alcohol Mole Fraction in NAPL
XMNO(LSV,LFDA)	Oil Mole Fraction in NAPL
XMNOB(LSV,LBCA)	Boundary Oil Mole Fraction in NAPL
XMNW(LSV,LFDA)	Water Mole Fraction in NAPL
XMNWB(LSV,LBCA)	Boundary Water Mole Fraction in NAPL
XNA_O(LFDG)	Old Alcohol Mass Fraction in NAPL
XNA(LSV,LFDA)	Alchohol Mass Fraction in NAPL
XNAB(LSV,LBCA)	Boundary Alcohol Mass Fraction in NAPL
XNO_O(LFDG)	Old Oil Mass Fraction in NAPL
XNO(LSV,LFDA)	Oil Mass Fraction in NAPL
XNOB(LSV,LBCA)	Boundary Oil Mass Fraction in NAPL
XNW_O(LFDG)	Old Water Mass Fraction in NAPL
XNW(LSV,LFDA)	Water Mass Fraction in NAPL
XNWB(LSV,LBCA)	Boundary Water Mass Fraction in NAPL
XP(LFD)	X-Direction Node Centroid Position, m
XPG(LFDGR,LGRL)	X-Direction Node Centroid Position at Local Grid Level, m
XREF(3)	Generalized Curvilinear Coordinate Factor
XREFU(3)	Generalized Curvilinear Coordinate Factor
XXI(LFD)	Spatial Differential of X Along XI Coordinate
XZTA(LFD)	Spatial Differential of X Along Zeta Coordinate
Y(LFX+1,LFY+1,LFZ+1)	Y-Direction Lower Node Surface Position, m
YCN(LFDC,LSOLU)	Fraction of Solute in NAPL
YETA(LFD)	Spatial Differential of Y Along Eta Coordinate
YG(LFDC,LSOLU)	Fraction of Solute in Gas
YGB(LBCC,LSOLU)	Boundary Fraction of Solute in Gas
YL(LFDC,LSOLU)	Fraction of Solute in Aqueous
YLB(LBCC,LSOLU)	Boundary Fraction of Solute in Aqueous
YLO_O(LFDN)	Old Fraction of Dissolved-Oil in Aqueous
YLO(LSV,LFDN)	Fraction of Dissolved-Oil in Aqueous
YLS_O(LFDS)	Old Fraction of Dissolved-Salt in Aqueous
YLS(LSV,LFDS)	Fraction of Dissolved-Salt in Aqueous
YLSB(LSV,LBCS)	Boundary Fraction of Dissolved-Salt in Aqueous
YMLOW(LSV,LNWN)	Well Dissolved-Oil Mole Fraction in Aqueous
YNB(LBCC,LSOLU)	Boundary Fraction of Solute in NAPL
YP(LFD)	Y-Direction Node Centroid Position, m
YPG(LFDGR,LGRL)	Y-Direction Node Centroid Position at Local Grid Level, m

YXI(LFD)	Spatial Differential of Y Along XI Coordinate
YZTA(LFD)	Spatial Differential of Y Along Zeta Coordinate
Z(LFX+1,LFY+1,LFZ+1)	Z-Direction Lower Node Surface Position, m
ZCRA	Alcohol Critical Compressibility Factor
ZCRO	Oil Critical Compressibility Factor
ZCRS	Surfactant Critical Compressibility Factor
ZCRW	Water Critical Compressibility Factor
ZERO	Constant, 0.
ZETA(LFD)	Spatial Differential of Z Along Eta Coordinate
ZP(LFD)	Z-Direction Node Centroid Position, m
ZPG(LFDGR,LGRL)	Z-Direction Node Centroid Position at Local Grid Level, m
ZRA(LCMP)	Component Critical Compressibility Factor
ZRAA	Alcohol Rackett Compressibility Factor
ZRAO	Oil Rackett Compressibility Factor
ZTAS(LFD)	Sum of Squares of ZTAX, ZTAY and ZTAZ
ZTAX(LFD)	Grid Metric along ZETA Coordinate
ZTAY(LFD)	Grid Metric along ZETA Coordinate
ZTAZ(LFD)	Grid Metric along ZETA Coordinate
ZXI(LFD)	Spatial Differential of Z Along XI Coordinate
ZZTA(LFD)	Spatial Differential of Z Along Zeta Coordinate

Table 3.3. Primary Variable Sequencing Scheme

	Primary Variable Sequence			
STOMP Mode Index	**Equation 1**	**Equation 2**	**Equation 3**	**Equation 4**
1	Water			
2	Water	Air		
3	Energy	Water	Air	
4	Water	Oil		
5	Water	Air	Oil	
6	Energy	Water	Air	Oil
8	Water	Oil	Dissolved-Oil	
11	Water	Salt		
12	Water	Air	Salt	
13	Energy	Water	Air	Salt
32	Water	CO_2	NaCl	
33	Energy	Water	CO_2	NaCl

Table 3.4. Field Variable Indexing Scheme

Index	Description
1	Previous time step value
2	Current iteration or time step value
3	First primary variable incremented
4	Second primary variable incremented
5	Third primary variable incremented
6	Fourth primary variable incremented

Table 3.5. Flux Variable Indexing Scheme

Index	Description
1	Current iteration or time step value
2	First primary variable in the upper node incremented
3	First primary variable in the lower node incremented
4	Second primary variable in the upper node incremented
5	Second primary variable in the lower node incremented
6	Third primary variable in the upper node incremented
7	Third primary variable in the lower node incremented
8	Fourth primary variable in the upper node incremented
9	Fourth primary variable in the lower node incremented

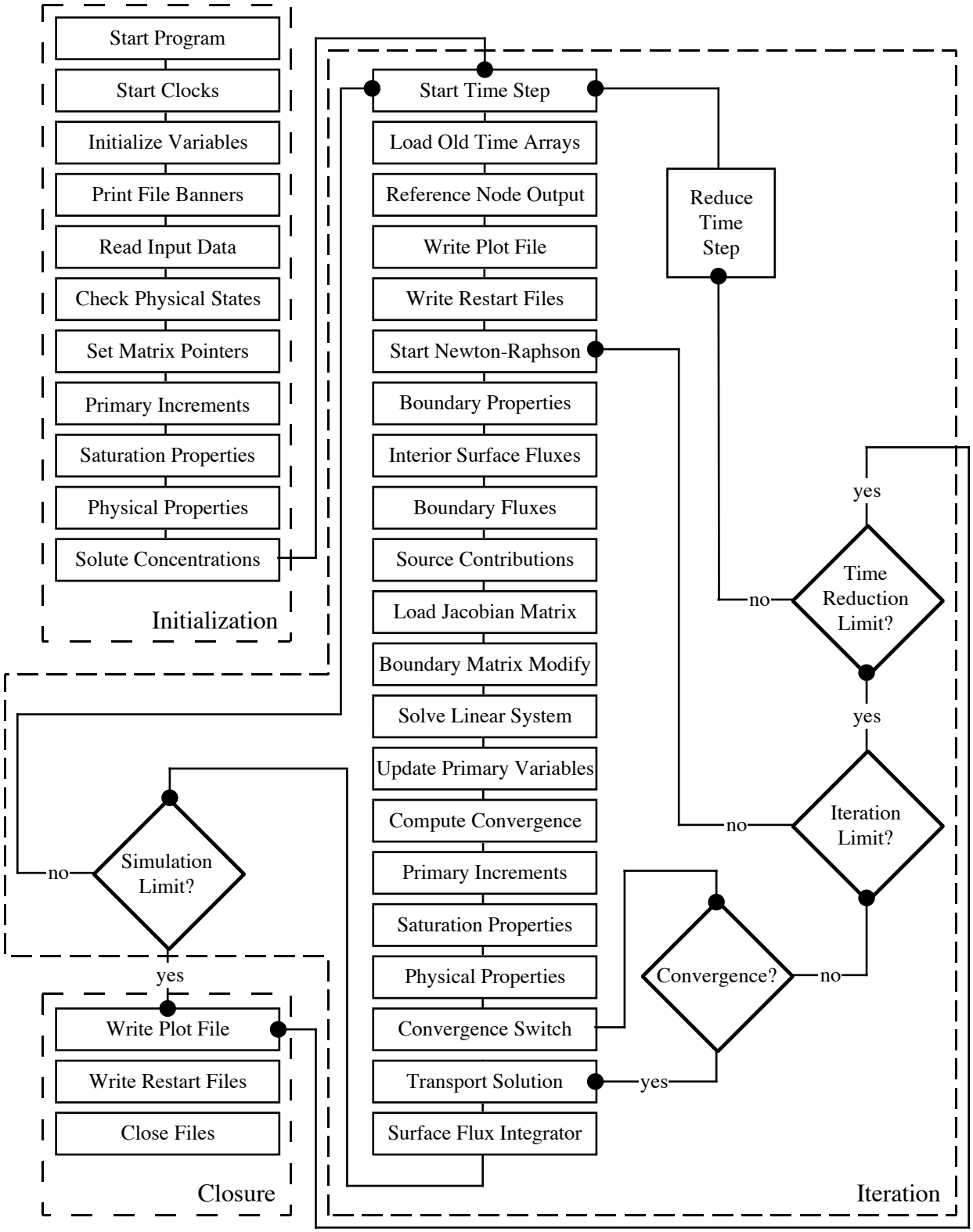

Figure 3.1. STOMP Flow

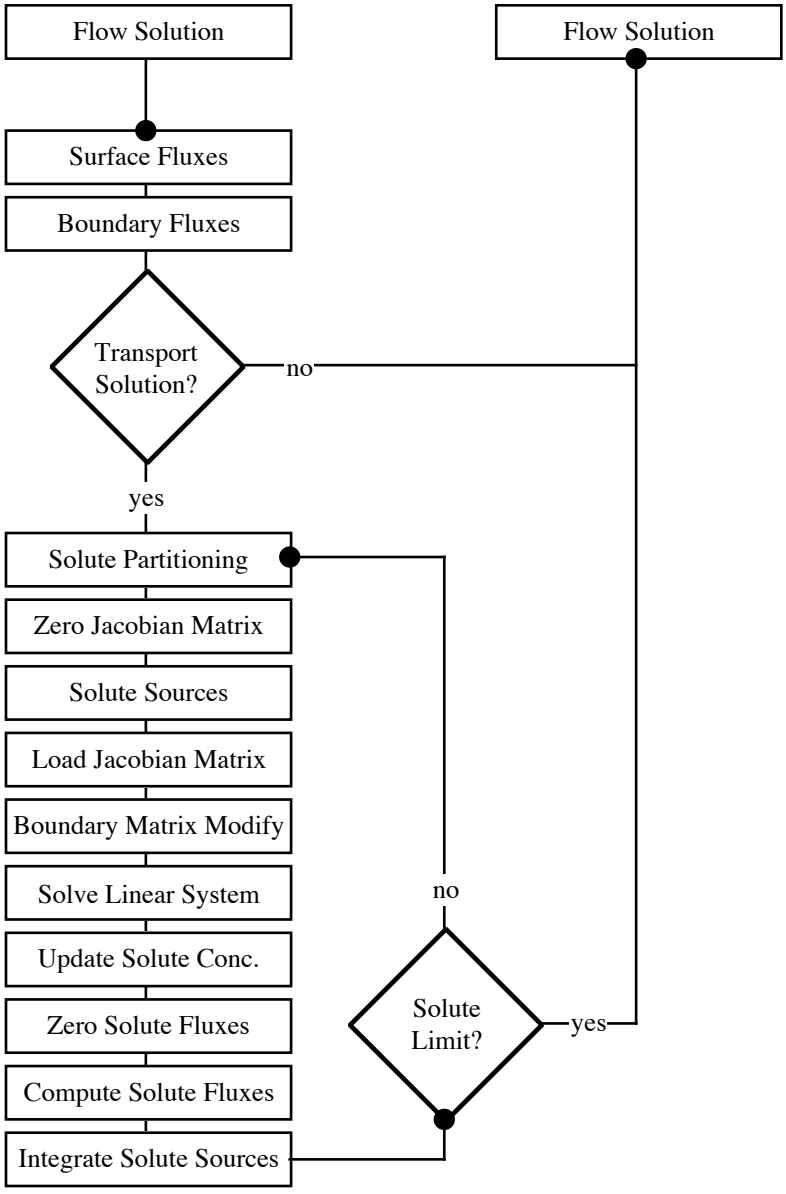

Figure 3.2. Transport Solution Flow Diagram

4. Input File

4.1 Introduction

The STOMP simulator is controlled through a text file, which must be entitled *input* for proper execution. This input file has a structured format composed of cards, which contain associated groups of input data. Depending on the operational mode, input cards maybe required, optional, or unused. Required cards must be present in an input file. Optional cards are not strictly required to execute the simulator, but may be required to execute a particular problem. Unused cards are treated as additional text that is unrecognized by the simulator, but will not hinder a proper execution. Cards may appear in any order within the input file; however, the data structure within a card is critical and must follow the formatting directives, shown in Appendix A and B. Data structures within cards vary with operational mode, which requires the user to follow a series of logic type statements to construct a readable input file. The simulator contains logic to capture and report some input errors. These capabilities are primarily limited to indicating syntax or formatting type errors and will generally not reveal errors such as those associated with ill-posed problems, atypical parameters, or inappropriate grid structures, for example. Because of its text format, STOMP input files may be generated with text editors, word processors, spread sheet programs, or graphical user interfaces and is portable between computing environments. The simulator's read routines are case insensitive (e.g., SAND and sand are equivalent) and allow considerable flexibility in specifying the simulation directives; however, close attention to the formatting instructions in this section will be necessary to prepare an executable input file.

4.2 Input File Structure

A STOMP input file is composed of cards, some of which are required and others which are optional or unused. The number of required cards depends on the operational mode. If an attempt is made to execute the simulator on an *input* file with an incomplete set of required cards, an error message will be generated and the code execution will stop. Optional cards are used to specify STOMP capabilities that may be required to execute a particular problem or generate desired output data. These cards are considered optional, because the capabilities accessed through these cards are not necessarily required to execute the code. Execution of the simulator on input files with an incomplete set of optional cards yields messages, which will note the missing optional cards but allow the execution to continue. A summary of the required and optional cards which compose a STOMP input file for each operational mode is shown in Table 4.1. Each card begins with a header, which must contain a tilde symbol in the first column followed by the card name (e.g., ~Simulation Title Card. Cards may be arranged in any order

within an input file; however, the input format within a card is structured. If a card type is repeated within an input file only the first card will be read by the simulator; the other cards of similar type will be ignored. Blank lines or additional comment lines may be included in the input file outside of the card structures.

4.3 Formatting and Notation

Every input card has formatting specifications, which must be followed to create an input file that is readable by the simulator. The format structure of certain cards varies with the operational mode and/or other input options, and the user must strictly follow the formatting instructions for the particular operational mode of interest. As may be expected, the complexity of the input increases with the number of solved equations. Guides for formatting input cards and card lines are given in Appendix B. The formatting guides shown in the Appendix were written in a compact format using special notations. Definitions of the formatting notation used in Appendix are given in Appendix A.

Input files are organized into three hierarchical structures, cards, lines, and data. Cards are delimited by a tilde (~) in the first column of the card title. Lines are delimited by hard returns and data are delimited by commas. The simulator recognizes three different types of data: integers, reals and character strings. Integers are used primarily to indicate indices or integral numbers. Integer data must be entered without decimal points or exponential notation. Real data are used to indicate dimensional parameters and can contain decimal points and/or exponential notation. Character string data are primarily used to indicate names, options and units, and are limited to 64 characters unless otherwise noted. Card lines comprise a series of input data items delimited by commas. A comma at the end of the card line is required to close the last data item. The format structure for each line in a card is shown in Appendix B using a format guide (e.g., **Format:** *Integer*^{a}*, Real*^{b}*, Char*^{c}*,*). Data types in the format statements are indicated as *Integer, Real* and *Char* (i.e., integer, real and character string) with lowercase letter superscripts. The lettered superscripts are used to make correspondence between the data item and it's position in the input line. The total length a single input line cannot exceed 132 characters. Additional text (comments or notes) may appear after the closing comma of an input line. Many input variables contain default values (indicated with an underline in Appendix B). Default values can be accessed by using a *null* entry. The *null* entry requires that the closing comma for the input data item immediately follow the previous closing comma or only blank spaces fill the space between the closing commas. To start a line with a *null* entry, the closing comma can occur in column 1 or after a number of blank spaces. Example input cards are shown in Appendix B after the formatting instructions for each card. A considerable portion of the input file formatting and creation work can be eliminated by developing input files from previously generated files or through an input generator.

4.4 Units

The simulator offers considerable flexible in specifying units. The user can declare units for both input or output data. Unspecified units will be assumed to be in standard Systeme Internationale (SI) units for the data item. The simulator operates internally almost exclusively in SI units, with pressures expressed in gauge, relative to 1 atm. Unless specifically stated, all input and output pressures are expressed in absolute values. Unit variables are read by the simulator as character strings, translated into primary unit form, and compared against the standard unit form for each data item. During the translation to primary unit form a conversion factor to SI units is generated. A unit character string comprises a combination of the recognized units delimited by spaces and/or a single divisor symbol (i.e., /). Only one divisor may appear in a unit character string. Spaces should not be used to separate the units immediately prior to or following the divisor symbol. The unit strings prior to the divisor symbol are considered as part of the numerator, and conversely the unit strings following the divisor symbol are considered as part of the denominator. Units recognized by the simulator are listed in Table 4.2. The units listed under the subtitle *Miscellaneous Units* are primarily descriptive units that may be included in a unit character string to increase its readability. The *hc* unit, however, is uniquely reserved for specifying rock/soil hydraulic conductivities, which are normally expressed in velocity units. Without the *hc* unit the input for rock/soil permeability will be interpreted as an intrinsic permeability value. The simulator does not apply hydraulic conductivity values directly, but instead first converts hydraulic conductivity values into intrinsic permeability values using the density and viscosity of water at 20 C. Therefore, when rock/soil permeability values are read as hydraulic conductivities the associated unit character string should include the *hc* unit (e.g., hc cm/hr). The unit *wh* indicates water equivalent head and can be combined with a length unit for a pressure unit (e.g., wh ft).

As an example, the standard units for thermal conductivity are W/K m, however a user may prefer to specify thermal conductivity in the standard English units of Btu in/hr ft^2 F. In this case the user would enter the character string "Btu in/hr ft^2 F" for the units data item. Note that the carat (i.e., ^) symbol is used to indicate an exponential. During simulator execution, the string of English units for thermal conductivity would be translated into the primary unit form "m/s^3 kg K" and compared against the standard unit form. If the primary unit form does not agree with the standard unit form, then an error message is generated by the simulator and the program execution stops. Otherwise if the comparison were successful, then the input data for thermal conductivity, which precedes its unit string, would be converted to SI units.

4.5 Card Descriptions

Formatting instructions for the input cards are provided in the Appendix. This section provides a brief synopsis of each input card with emphasis on its purpose and application. Italicized words refer to specific files, cards, options and data entries shown in the card formats in the Appendix. Input cards are listed in alphabetical order in the Appendix. In this section, input cards are listed in a sequence most user's find convenient when developing an input file.

4.5.1 Simulation Title Card

This card primarily provides a means to document a simulation. Information recorded in this card is rewritten on the *output* file, which then serves as a permanent record of the simulation. The user is encouraged to use descriptive titles and to briefly record the specifics and purpose of the simulation in the *Simulation Notes* section of the card. This becomes especially valuable, when the user is making repeated simulations with small modifications to the input parameters. The time required to indicate these changes in either the *Simulation Title* or *Simulation Notes* will be invaluable when reviewing archived *output* files.

4.5.2 Solution Control Card

This card controls many general operational aspects of a simulation. Three *Execution Modes* are recognized: *Normal, Restart,* or *Initial Conditions*. In the *Normal* and *Initial Conditions* modes initial state conditions are declared through the *Initial Conditions Card*. In the *Restart* mode initial state conditions are assigned via a restart file from a previous execution or declared through the *Initial Conditions Card*, using the special *overwrite* option on selected parameters. Unless specified through the *Output Control Card*, restart files (i.e., *restart.n*) are generated at each *plot.n* write event, and have name extensions that correspond to the generating time step (e.g., the file restart.28 would have been generated at the conclusion of time step 28). Restart files are text files that contain simulation time and control information, and a collection of field variables needed to redefined the simulation state for the operational mode. In the *Normal* mode the simulator executes from a declared start time, using an initial state declared through the *Initial Conditions Card*, until: the declared stop time, the declared number of time steps, an execution error, or a sequence of convergence failures. In the *Initial Conditions* mode the simulator reads the input deck, checking for formatting errors and stops; no time steps are executed . In the *Restart* mode the simulator executes from either a declared start time or the start time specified in the restart file, using an initial state defined by a previous execution, until: the declared stop time, the declared number of time steps, an execution error, or a sequence of convergence failures. In the *Restart* mode, the initial state defined by the restart file can be modified using the special *"overwrite"* option on selected parameters, declared through the *Initial Conditions Card*.

By default the *Normal* and *Restart* modes use Euler time differencing (i.e., first order backward). The *Second Order* option switches the time-differencing scheme to second order backward. The *Restart* mode offers an option to read named restart files, by including the keyword *"file"* in the *Execution Mode Option* character string. This option triggers the code to read an additional character string, which is the name of the restart file. For example, the input line: re*start using second order time differencing from file,restart.3456,* would start the simulator using second-order backward time differencing with the initial state declared through the field parameters in the file *restart.3456.* In addition to this option, the *Restart* mode offers an option to read restart files, generated by other operational modes by including the key word *"mode"*. For example, the input line: *restart mode file,restart.3456,1* would start the simulator with the initial state declared through the field parameters in the file *restart.3456,* which was

created from a previous *Water* operational mode (i.e., *Mode 1*). This is considered an advanced option, as the user is responsible for defining the initial system state, for the current operational mode, from a combination of parameters from the restart file and *Initial Conditions Card*.

The *Water* operational mode has two additional options associated with the *Execution Mode Option*: *Dynamic Domain* and *No Flow*. The *Dynamic Domain* option conserves computations by temporarily removing nodes from the computational domain where changes in the flow field are insignificant. The active computational domain is updated every time step. The *No Flow* option (*Water w/ Transport* operational mode only) is used for transporting solutes through a steady-state flow field. For these simulations the flow field is computed once during the initial time step and then remains unaltered.

The governing flow and transport equations to be solved are declared through the *Operational Mode Option*, which has two components: coupled flow equations and solute transport equations. The solved coupled flow equations are declared by choosing an operational mode from those listed in Table 2-1 (e.g., *Water*, *Water-Air-Energy*, *Water-Oil*, *Water-CO2-NaCl*). Operational modes are referred to by name or number; see Table 2-1 for a correspondence between operational mode name and numbers (e.g., *Water* or *Mode 1*, *Water-Air-Energy* or *Mode 3*, *Water-Oil* or *Mode 4*, *Water-CO2-NaCl* or *Mode 32*). Because execution performance is indirectly proportional to the number of solved governing equations, the user should select an operational mode which is most appropriate for the problem of interest. For example, if the physical system to be modeled does not contain thermal nor gas pressure gradients and is void of oil, then the most appropriate operational mode would be the *Water* operational mode, where only the water mass conservation equation is solved. Executing the described physical system under the *Water-Air* operational mode would yield nearly identical results, however, the execution time would be significantly increased. Considerable attention should be given to the selection of an appropriate operational mode for the physical system of interest. In terms of efficient and representative simulations, it is equally important not to eliminate critical physical phenomena through erroneous assumptions, as it is not to solve superfluous governing equations. Each operational mode allows the solution of solute transport equations, which are indicated by including the keyword *Transport* in the operational mode (e.g., *Water-Salt w/ Transport* or *Water-Air and Solute Transport*). Solute transport is solved using the Patankar method, unless the keyword *TVD* appears, which indicates the TVD method. The TVD method for coupled salt, dissolved oil or surfactant transport is indicated with the keyword *LFL*.

Execution periods refer to a period of simulation time. The simulator allows the user to specify a single or multiple execution periods. For each execution period the use can control the initial time step, maximum time step, time step acceleration factor, maximum number of Newton-Raphson iterations and convergence criterion. Recommended values for the *Time Step Acceleration Factor*, *Maximum Number of Newton-Raphson Iterations* and *Convergence Criterion* are 1.25, 8, and 1.e-06, respectively. Simulations involving complex phase transitions often require more Newton-Raphson iterations to reach convergence, because of the design of the phase transition algorithms. For these types of simulations a value of 16 is recommended for the *Maximum Number of Newton-Raphson Iterations*. Except under special circumstances, it is not recommended to change the value for the *Convergence Criterion* from its recommended value.

This value has proven through numerous applications to achieve a good balance between accuracy and execution speed.

Field variables, which include physical, thermodynamic, and hydrologic properties are defined in the finite-difference formulation at the node centers. Conversely, flux variables are defined at node interfaces. Computation of flux variables requires knowledge of field variables at node interfaces. Values of flux variables at node interfaces are evaluated by averaging the field values for the two nodes adjoining an interfacial surface. Interfacial averaging schemes may be declared individually for each field variable through the *Interfacial Averaging Variables* input. The default interfacial averaging schemes for the simulator are shown in Table 4.3. For simulations of physical systems involving heat transfer it should be noted that convergence problems might arise if the density properties are not averaged with upwind weighting. Likewise infiltration problems typically demonstrate strong dependencies on the relative permeability of the infiltrating fluid.

4.5.3 Grid Card

The simulator's finite-difference formulation is based on orthogonal grid systems. Currently two orthogonal grid systems are recognized, the Cartesian and cylindrical coordinate systems. The Cartesian coordinate system is a "right-handed" system with the longitudinal axis (z–direction) aligned with the negative gravitational vector. Cartesian coordinate systems may be defined that are tilted with respect to the gravitational vector. The cylindrical coordinate system has the longitudinal axis (z-direction) aligned with the negative gravitational vector. The radial (r–direction) and azimuthal ($\square$-direction) axes are constrained to a horizontal plane. For the Cartesian coordinate system the terms west, south, and bottom refer to the negative *x-, y-,* and *z*-directions, respectively, and the terms east, north, and top refer to the positive *x-, y-,* and *z*-directions, respectively. For the cylindrical coordinate system the terms west, south, and bottom refer to the negative *r-,* $\square$*-,* and *z*-directions, respectively, and the terms east, north, and top refer to the positive *r-,* $\square$*-,* and *z*-directions, respectively. Negative dimensional values are not recognized and axes are defined positive towards increasing node numbers. The grid dimensions which are specified on the *Grid Card* refer to node surfaces; therefore, for grids with nonuniform spacing one plus the number of nodes entries are required for each grid direction. Node volumes are defined by their bounding surfaces. Cylindrical coordinates systems are restricted to azimuthal axes which are less than or equal to 360 degrees. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for graphical descriptions of the Cartesian and cylindrical coordinate systems.

4.5.4 Rock/Soil Zonation Card

In the STOMP simulator, hydrologic flow and transport properties are associated with a rock/soil types. The zonation or distribution of rock/soil types across the computational domain is declared via the *Rock/Soil Zonation Card*. Several options are available through the this card for declaring the distribution of rock/soil types: explicit declaration, external files, or *IJK Indexing*. Explicit declaration is a good choice for simulations involving a limited number of

rock/soil types. Under this option the user associates rock/soil type names with portions of the computational domain. Rock/soil names must be unique and contain no more than 64 characters. The following key words have special meanings when used within a rock/soil name: *"skip conv", "dp", "dual porosity", "fractured"*; where, "*skip conv"* is used to skip convergence checks on nodes with associated rock/soil types, and *"dp", "dual porosity", and "fractured"* is used to declare the rock/soil type as having dual porosity characteristics. When explicitly declaring the rock/soil zonations, rock/soil names can be repeatedly applied to a grid cells or domain of grid cells, with only the last definition being applied. For example, to simplify the zonation of a problem with a dominant rock/soil type and isolated pockets or bands of another rock/soil type, the user should initially declare the dominant rock/soil type as covering the entire problem domain. With subsequent input lines, the user can overwrite the original rock/soil type for selected nodes with another rock/soil type. This layering approach is recommended and will often avoid leaving nodes of undeclared rock/soil types. External files can be useful when the rock/soil zonation data are automatically generated (e.g., geologic visualization programs). *IJK Indexing* should be used when every grid cell is associated with a unique rock/soil type, as common with statistically generated rock/soil distributions. With *IJK Indexing* the concept of rock/soil types is abandoned for a system where hydrologic flow and transport properties are associated with the grid cell. The rock/soil type of a grid cell is extended to boundary surfaces adjacent to the grid cell (i.e., boundary surface hydrologic properties are computed using the rock/soil type property or *IJK Indexing* property descriptions for the node adjacent to the boundary surface. Hydrogeologic properties for rock/soil types are specified through the *Mechanical Properties Card*, *Hydraulic Properties Card*, *Thermal Properties Card*, *Saturation Function Card*, *Aqueous Relative Permeability Function Card*, *Dissolved-Oil Transport Card, Salt Transport Card, Surfactant Transport Card, Gas Relative Permeability Function Card*, *NAPL Relative Permeability Function Card* and *Solute/Porous Media Interaction Card*.

4.5.5 Scaling Factor Card

The *Water* and *Fluid* operational modes include an option for lumping rock/soil types into scaling groups; where hydraulic parameters for individual rock/soil types are computed from those declared for the scaling group and scaling factors declared for the rock/soil type. The premise behind the scaling factor card is that like rock/soil types can be scaled to a common set of hydrologic parameters. When applied to inverse calculations, this concept can reduce the number of unknown hydrologic parameters by a factor equal to the number of rock/soil types in the scaling group. For example, an inverse problem requiring 5 hydrologic parameters per rock/soil type in a domain with 7 different rock/soil types would have 35 unknowns. Using the scaling factor approach, the number of unknowns could be reduced to 5 hydrologic parameters for the scaling group. The association between rock/soil types and scaling groups is made on the *Rock/Soil Zonation Card* and this card is used to declare scaling factors. If an association is made between a rock/soil type and a scaling group, then the hydrologic parameters for the scaling group and not the individual rock/soil types are declared through the hydrologic property and function cards (i.e., *Mechanical Properties Card*, *Hydraulic Properties Card*, *Saturation Function Card*, and *Aqueous Relative Permeability Function Card*). An input file can simultaneously contain

scaling groups and individual rock/soil types. When scaling groups are applied, individual rock/soil parameters for *Saturated Hydraulic Conductivity, Diffusive Porosity, van Genuchten "alpha" and "n," Brooks-Corey "psi" and "lambda,"* and *Residual Saturation* are computed from the scaling group parameters and scaling factors through a scaling function. Scaling factors are dependent on the scaling group, the rock/soil type, and hydrologic parameter. Scaling functions are either linear or logarithmic.

4.5.6 Inactive Nodes Card

This card allows the user to declare nodes, within the computational domain, as inactive or noncomputational. Inactive nodes are those nodes that will remain permanently excluded from the computational domain. Because inactive nodes reduce the number of unknowns, therefore increase execution speed, their use is recommended. Inactive nodes can be used to simulate irregular boundaries or define impervious zones. Inactive nodes can also be used to define internal boundary surfaces. Boundary conditions may be applied to any surface between an active and inactive node. Boundary condition definitions, however, must refer to an active node. Examples of appropriate uses for inactive nodes include representation of the atmosphere above a sloping surface, or conversely a nonhorizontal water table along the bottom of a domain. To facilitate user input a number of features have been incorporated into this card: explicit declaration, external files, rock/soil type association, and multiple entries. Using explicit declaration the user declares domains of nodes as being inactive. Declaring inactive nodes via external files can be useful when the inactive nodes domains are automatically generated . As with the external files option, the rock/soil type association can be useful when the rock/soil zonation has been entered via external files. As no rock/soil types are defined for *IJK Indexing*, rock/soil type association is not recognized in conjunction with *IJK Indexing*. The multiple entries option allows a combination of input options on a single *Inactive Nodes Card* (e.g., explicit declaration and rock/soil type association).

4.5.7 Mechanical Properties Card

This card allows the user to assign values to the particle density, porosity, specific storativity, compressibility, and tortuosity function for each defined rock/soil type. Every rock/soil type defined on the *Rock/Soil Zonation Card* must be referenced. With the *IJK Indexing* option, node dependent parameters are entered via external files and node independent parameters are entered directly on the card. *Particle Density* represents the rock grain density. This value can be defaulted to 2650 kg/m^3 by using a *null* entry for both the particle density and its associated unit. *Total Porosity* refers to total connected and unconnected pore volumes. *Diffusive Porosity* refers to only the connected pore volume. As with *Particle Density*, the *Specific Storativity* can be defaulted by using a *null* entry for both the *Specific Storativity* and its associated *Units*. Default specific storativity is computed from the *Diffusive Porosity* and a default value of $1.x10^{-7}$ 1/Pa for the compressibility. Alternatively, the *Pore Compressibility* can be read directly along with a *Pore Compressibility Reference Pressure*. Tortuosity functions are required for simulations that involve solute transport or diffusion of components through phases

(e.g., water vapor diffusing through the gas phase or dissolved oil diffusing through the aqueous phase). Tortuosities can be computed either as constants, which require input values, or as functions of the phase saturation and diffusive porosity through the Millington and Quirk function. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a description of the Millington and Quirk tortuosity function. The key words "dp", "dual porosity", or "fractured" in the rock/soil name indicate a dual porosity medium, which triggers the reading of both matrix and fracture properties (e.g., *Fracture and Matrix Diffusive Porosity*, *Fracture and Matrix Specific Storativity*).

4.5.8 Hydraulic Properties Card

This card allows the user to assign values to the intrinsic permeability or each defined rock/soil type. Every rock/soil type defined on the Rock/Soil Zonation Card must be referenced. With the *IJK Indexing* option, node dependent parameters are entered via external files and node independent parameters are entered directly on the card. Intrinsic permeability can be declared directly or through entering the hydraulic conductivity at reference conditions, where reference conditions refer to atmospheric pressure and 20 C. By default the simulator reads the permeability values on this card as intrinsic permeabilities, unless the character string *hc* is included in the associated units. Default units of m^2 are applied to null entries for the units associated with permeability values. A primary assumption with the simulator is that principal components of the intrinsic permeability tensor are aligned with the principal coordinate directions. For cylindrical coordinate systems the radial, azimuthal, and vertical permeabilities correspond with the *x-, y-,* and *z*-direction values, respectively. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a description of the conversion of hydraulic conductivity at reference conditions to intrinsic permeability. The key words "dp", "dual porosity", or "fractured" in the rock/soil name indicate a dual porosity medium, which triggers the reading of both matrix and fracture properties (e.g., *Fracture and Matrix Intrinsic Permeability*). Reduction in intrinsic permeability with precipitation of salt, as computed in the water-CO2-NaCl and water-CO2-NaCl-Energy operational modes requires two parameters, which are entered via this card: *Pore-Body Fractional Length* and *Fractional Critical Porosity*.

4.5.9 Saturation Function Card

This card allows the user to declare and define a saturation-capillary pressure function for each defined rock/soil type. Every rock/soil type defined on the *Rock/Soil Zonation Card* must be referenced. With the *IJK Indexing* option, node dependent parameters are entered via external files and node independent parameters are entered directly on the card. Saturation function types and the required input items vary greatly among the operational modes. Functional forms for the saturation-capillary pressure functions are preferred; however, for some operational modes, tabular input is acceptable. By default, tabular data will be interpolated using linear interpolation; where, values beyond the table limits will be assigned either the table minimum or maximum values appropriately. The available function forms and form options varies among the operational modes, with all operational modes offering options to use van Genuchten and

Brooks and Corey functions. For the van Genuchten function, the *"m" Parameter* can be defaulted (with a *null* entry) or assigned a value. Default values will depend on which porosity distribution model (Mualem or Burdine) is chosen on the relative permeability function cards for the aqueous phase. Common options to these functional forms include: nonwetting fluid entrapment and nonwetting fluid residual formation. Fluid entrapment functions consider the hysteresis between wetting and draining paths due only to entrapment of the nonwetting fluids by imbibing wetting fluids. Fluid entrapment functions allow the specification of initial trapped nonwetting fluid saturations without specifying detailed saturation path histories. These functions additionally allow dissolution of the nonwetting phase into the wetting phase (e.g., dissolution of air and oil into the aqueous phase). Nonhysteretic functions require fewer input items than their hysteretic counterparts and use a single scanning path for both drainage and imbibition events. Operational modes that consider NAPL flow, require interfacial tension parameters (i.e., gas-NAPL, NAPL-aqueous, and gas-aqueous) for scaling the saturation-capillary pressure functions. Allowances are made for spreading and nonspreading NAPLs. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a complete description of allowed saturation-capillary pressure functions.

4.5.10 Aqueous Relative Permeability Function Cards

This collection of cards allows the user to declare and define a relative permeability-saturation function for the aqueous phase for each defined rock/soil type. Every rock/soil type defined on the *Rock/Soil Zonation Card* must be referenced. With the *IJK Indexing* option, node dependent parameters are entered via external files and node independent parameters are entered directly on the card. Aqueous relative permeability function types and the required input items are dependent on the operational mode. The Mualem and Burdine relative permeability functions are also dependent on the saturation function type and are strictly applicable to the van Genuchten and Brooks and Corey functions. For these functions, either the van Genuchten *"m"* parameter or the Brooks and Corey *"λ"* parameter can be defaulted to the values entered or defaulted with the saturation function. Functional forms for the relative permeability-saturation functions are preferred. Tabular input is acceptable for some operational modes. By default, tabular data will be interpolated using linear interpolation; where, values beyond the table limits will be assigned either the table minimum or maximum values appropriately. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a complete description of aqueous relative permeability-saturation functions. For some operational modes aqueous relative permeability functions can be specified as a diagonal tensor (i.e., unique aqueous relative permeability functions can be declared for the principal grid-coordinate directions). With this option, a default aqueous relative permeability can be declared for all coordinate directions via the *Aqueous Relative Permeability Card*, then individual directional components can be specified via the directional cards (i.e., *X Aqueous Relative Permeability Card, Y Aqueous Relative Permeability Card,* or *Z Aqueous Relative Permeability Card*).

4.5.11 Gas Relative Permeability Function Card

This card is used to declare and define a relative permeability-saturation function for the gas phase for each defined rock/soil type. Every rock/soil type defined on the *Rock/Soil Zonation Card* must be referenced. With the *IJK Indexing* option, node dependent parameters are entered via external files and node independent parameters are entered directly on the card. This card is required only for simulations involving flow and/or transport through the gas phase. Gas relative permeability function types and the required input items are dependent on the operational mode. The Mualem and Burdine relative permeability functions are also dependent on the saturation function type and are strictly applicable to the van Genuchten and Brooks and Corey functions. For these functions, either the van Genuchten *"m"* parameter or the Brooks and Corey *"□"* parameter can be defaulted to the values entered or defaulted with the saturation function. Functional forms for the relative permeability-saturation functions are preferred, however tabular input is acceptable. By default, tabular data will be interpolated using linear interpolation; cubic spline interpolation is optional; where, values beyond the table limits will be assigned either the table minimum or maximum values appropriately. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a complete description of gas relative permeability-saturation functions.

4.5.12 NAPL Relative Permeability Function Card

This card allows the user to declare and define a relative permeability-saturation function for the NAPL for each defined rock/soil type. Every rock/soil type defined on the *Rock/Soil Zonation Card* must be referenced. With the *IJK Indexing* option, node dependent parameters are entered via external files and node independent parameters are entered directly on the card. This card is required only for simulations involving flow and/or transport through the NAPL phase. The Mualem and Burdine relative permeability functions for the NAPL phase are dependent on the saturation function type and are strictly applicable to the van Genuchten and Brooks and Corey functions. For these functions, either the van Genuchten *"m"* parameter or the Brooks and Corey *"□"* parameter can be defaulted to the values entered or defaulted with the saturation function. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a complete description of the gas relative permeability-saturation functions.

4.5.13 Thermal Properties Card

This card allows the user to assign values to the thermal conductivity and specific heat for each defined rock/soil type. Every rock/soil type defined on the *Rock/Soil Zonation Card* must be referenced. With the *IJK Indexing* option, node dependent parameters are entered via external files and node independent parameters are entered directly on the card. This card is required only for simulations involving the solution of the energy conservation equation. Declaration of the thermal conductivity, depends on the operational mode and function option. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a description of thermal conductivity functions. The *Constant* option fixes the thermal conductivity to a constant value, independent

of temperature or saturation. The *Parallel* option requires the thermal conductivity of the soil grains and models thermal conductivity with an equivalent parallel path model dependent on porosity, phase saturations, and temperature. The *Linear* and *Somerton* options scale the thermal conductivity between the unsaturated and saturated values depending on phase saturation. A primary assumption with the simulator is that principal components of the thermal conductivity tensor are aligned with the principal coordinate directions. For cylindrical coordinate systems the radial, azimuthal, and vertical permeabilities correspond with the *x-, y-,* and *z*-direction values, respectively.

4.5.14 Oil Properties Card

This card allows the user to declare functions and assign values required to compute the physical properties for an oil or dissolved oil. This card is required only for simulations involving flow and/or transport of an oil or dissolved oil. Properties for oils are computed from critical properties and functional parameters following the Corresponding States Method as presented by Reid et al. (1987). Because the oil property relations within the simulator were primarily based on the formulations in the reference book by Reid et al., the user is encouraged to obtain or borrow a copy of this reference book. Many of the requested input parameters for numerous oils may be found in the Appendix of this reference. Unless specifically stated below as an optional input, all parameters requested by this card should be considered as required. The *Freezing Point Temperature* is an optional input and can be defaulted with a *null* entry for the parameter and its associated units. The *Boiling Point Temperature* is used in computing oil component diffusion coefficients, oil vapor enthalpy, and liquid oil thermal conductivity; therefore, its value and associated units can be defaulted with *null* entries for simulations which do not involve oil component diffusion nor solution of the energy conservation equation. The *Critical Molar Volume* and its associated units are used only to calculate oil component diffusion coefficients and can be defaulted with *null* entries for simulations not involving oil component diffusion. The *Critical Compressibility* and *Dipole Moment* are used only to calculate the gas viscosity and can be defaulted with *null* entries for simulations that do not solve the gas flow equations. The *Pitzner Accentric Factor* and *Isobaric Molar Specific Heat Constants* are used only to calculate the liquid oil enthalpy and can be defaulted with *null* entries for simulations which do not involve heat transfer or solution of the energy conservation equation. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a complete description of the Corresponding States Method functions used to compute the oil property functions.

4.5.15 Dissolved-Oil Transport Card

This card allows the user to declare functions and assign values required for dissolved oil transport. If the operational mode is Water-Oil-Dissolved Oil or Water-Oil-Dissolved Oil-Surfactant, then an *Interphase Mass Transfer Function Option* must be declared using either the Welty or Parker function and the associated parameters specified. Properties for dissolved oil dispersion and adsorption must be specified with respect to each rock/soil type. With the *IJK Indexing* option, node dependent parameters are entered via external files and node independent

parameters are entered directly on the card. For each, the *Longitudinal Dispersivity* and *Transverse Dispersivity* of dissolved oil must be specified with respect to each rock/soil type, and a *Dissolved-Oil Adsorption Function* declared from among the choices of Linear, Linear Kd, Freundlich, or Langmuir functions, with associated parameters for the chosen function specified. Refer to Appendix B for details on the *Interphase Mass Transfer Function* and *Dissolved-Oil Adsorption Function* options and required input parameters specified through this card.

4.5.16 Salt Transport Card

This card allows the user to define porous media dependent dispersivities for salt. This card is required only for simulations involving salt transport. For operational modes involving salt transport, the salt transport equation is fully coupled with the flow and energy transport equations. For every porous media defined on the *Rock/Soil Zonation Card* dispersivities must be referenced. With the *IJK Indexing* option, node dependent parameters are entered via external files and node independent parameters are entered directly on the card. Computation of effective diffusivity may be either by a conventional or empirical (Kemper and van Schaik) equation. If the empirical equation is selected, a *Molecular Diffusion Rate* must also be specified for each porous media. *Longitudinal Dispersivity* is defined with respect to dispersion along the flow path and is assumed to be independent of the flow direction with respect to the porous media structure. Likewise, *Transverse Dispersivity* is defined with respect to dispersion transverse to the flow path, independent of the flow direction. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a more complete description of the salt mass conservation equation.

4.5.17 Atmospheric Conditions Card

This card allows the user to specify time-varying atmospheric conditions for temperature, pressure, relative humidity, net solar radiation, and wind speed. At present, data from this card is only used in conjunction with the *Plant Card* and the *Water-Air-Energy* operational mode. This card comprises two input sections: reference data and time-varying atmospheric conditions data. The reference data inputs include: atmospheric conditions start time; wind speed, air temperature, and relative humidity measurement heights, and local longitude and meridian. The *Net Solar Radiation* input should be total solar radiation incident on a horizontal ground surface. The *Atmospheric Start Time*, *Local Longitude*, and *Local Meridian* inputs are used to split total horizontal solar radiation into direct-beam and indirect components. The *Wind Speed Measurement Height* input is used to convert wind speed at the measurement height to wind speed near the ground surface and plant canopy heights. As with the wind speed, the *Air Temperature* and *Relative Humidity Measurement Height* inputs are used to convert their respective data to ground surface and plant canopy heights. The time-varying inputs are read in tabular form; where, each line of input represents the atmospheric conditions at that point in time. The *Atmospheric Condition Time* input is the first entry in each time-varying input line and is relative to the *Atmospheric Start Time* declared in the reference data input section. The time entry is followed in succession by the *Atmospheric Condition Temperature*, *Atmospheric*

Condition Pressure, Atmospheric Conditions Relative Humidity, *Atmospheric Conditions Net Solar Radiation*, and *Atmospheric Conditions Wind Speed* inputs. The time-varying inputs can be read directly from the input file or from an external file.

4.5.18 Plant Card

This card is used to specify plant varietals and characteristics growing on the ground surface. Under the current configuration of the simulator, data from this card is only used in conjunction with the *Atmospheric Conditions Card* and the *Water-Air-Energy* operational mode. With this card the user defines plant varietals and then enters characteristics about each varietal, including its root distribution. The areal location and ground-coverage density of plants is defined through the *Boundary Conditions Card*, using the *Shuttleworth-Wallace* boundary condition. The *Shuttleworth-Wallace* boundary condition computes transpiration and evaporation losses from the ground surface and surface vegetation in response to atmospheric conditions, specified on the *Atmospheric Conditions Card*. Plant varietal names must be unique and contain no more than 64 characters. For each defined *Plant Varietal*, the user must specify characteristics about the plant, including: vertical root extent, canopy height, stress characteristics, and crop coefficients. The extent and spatial distribution of plant roots in the vertical direction is defined through three input parameters: *Max. Root Depth*, *Null Root Depth*, *Root Depth Fit Parameter*. The *Max. Root Depth* input represents the maximum rooting depth, and the *Null Root Depth* and *Root Depth Fit Parameter* inputs are empirical parameters. These parameters provide for zero root water uptake at the maximum rooting depth, account for asymmetrical root water uptake with depth, and allow for maximum root water uptake at any depth from the ground surface to the maximum rooting depth. The *Plant Canopy Height* input is used to compute shading of taller plants on shorter plants, and used to compute canopy height wind speed and temperature. A stress function is available to reduce the root water uptake as a function of capillary pressure and uses four coefficient inputs: *Water Stress Point 1*, *Water Stress Point 2*, *Water Stress Point 3*, and *Water Stress Point 4*. Variations in the transpiration rate, root water uptake, and foliage on a yearly basis are handled via a crop-coefficient function, which requires four crop coefficients and associated times: *Crop Coefficient (Start)*, *Crop Coefficient (Mature Stage 1)*, *Crop Coefficient (Mature Stage 2)*, and *Crop Coefficient (Die-Off)*; where, linear interpolation based on the time of year is used between crop coefficients.

4.5.19 Solute/Fluid Interactions Card

This card allows the user to define solutes, solute-fluid interactions, and solute radioactive decay paths. This card is required only for simulations involving transport of solutes. The simulator is capable of simulating any number of solutes with the assumption that solute concentrations remain dilute (solute concentrations do not vary the physical properties of the transporting fluid phases). Solutes can decay radioactively to produce other solutes. For the loose coupling between parent solutes and progeny solutes to function properly, parent solutes must be defined on this card prior to their progeny. The simulator actually solves the transport equation for each solute sequentially in reverse order from the definition list on this card. Solutes

are defined by a unique solute name, which must contain no more than 64 characters. *Chain Decay Fraction* relates a parent decay member with a progeny and refers to the fraction of the decaying parent which produces a particular progeny. *Chain Decay Fraction* inputs should be fractional values between 0.0 and 1.0, inclusively. Solute-fluid interaction parameters such as molecular diffusion coefficients and interface partition coefficients depend on the operational mode. Molecular diffusion coefficients refer to the diffusion of the solute through the transporting fluid phase outside of the porous media. Corrections for transport through porous media are handled within the simulator. The interface partition coefficients define the equilibrium distribution of solute between the active liquid phases. Partition coefficients which define the equilibrium adsorption of solute onto the solid phase are defined on the *Solute/Porous Media Interaction Card*. The user should carefully note the definitions and requested units for each interphase partition coefficient. The *Aqueous-Gas Partition Coefficient* is the ratio of the concentration of solute in the aqueous phase per unit mass of aqueous phase to the concentration of solute in the gas phase per unit volume of gas phase. The *Aqueous-NAPL Partition Coefficient* is the ratio of the concentration of solute in the aqueous phase per unit mass of aqueous phase to the concentration of solute in the NAPL phase per unit mass of NAPL. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a more complete description of the interface partition coefficients, solute diffusion coefficients, radioactive decay rate equations, and radioactive chain decay fractions.

4.5.20 Solute/Porous Media Interaction Card

This card allows the user to define solid-aqueous phase partition coefficients and porous media dependent hydraulic dispersivities. This card is required only for simulations involving transport of solutes. This card differs from the *Solute/Fluid Interaction Card* because the input parameters declared are dependent on both the solute and rock/soil type. For every solute defined on the *Solute/Fluid Interaction Card*, every rock/soil type defined on the *Rock/Soil Zonation Card* must be referenced. With the *IJK Indexing* option, node dependent parameters are entered via external files and node independent parameters are entered directly on the card. The *Solid-Aqueous Partition Coefficient* defines the interface equilibrium of a solute adsorbed on the solid and dissolved in the aqueous phase, and refers to the concentration of solute adsorbed on the solid phase per unit mass of solid phase over the concentration of solute dissolved in the aqueous phase per unit mass aqueous phase. The longitudinal and transverse hydraulic dispersivities are properties only of the rock/soil type. *Longitudinal Dispersivity* is defined with respect to dispersion along the flow path and is assumed to be independent of the flow direction with respect to the porous media structure. Likewise, *Transverse Dispersivity* is defined with respect to dispersion transverse to the flow path, independent of the flow direction. Refer to the STOMP Theory Guide (White and Oostrom, 2000) for a more complete description of the solid-aqueous interface partitioning and hydraulic dispersion of transported solutes.

4.5.21 Initial Conditions Card

This card allows the user to assign starting values to both primary and secondary field variables. The current version of the simulator does not allow the user to initialize flux and/or surface variables. Such initializations would be useful for solving transport problems with steady flow fields. For *restart* simulations, initial conditions are obtained from the *restart* file, unless the word *"Overwrite"* appears with the initial condition variable name. Not all initial condition variables can be overwritten. Those variables that can be over written are indicated in the Appendix with the optional text string [Overwrite] show in the *Variable Name Option.* Transport problem simulations with the simulator require the solution of the flow fields. The variables which may be initialized are dependent on the operational mode. For all operational modes, default values for the initial conditions have been specified within STOMP as shown in Table 4.4. For the two-phase, gas-aqueous operational modes (e.g., *Water-Air, Water-CO2-NaCl*), STOMP allows the user to specify the initial aqueous saturation by assigning initial conditions for any two of the following three variables: gas pressure, aqueous pressure, and aqueous saturation. The unassigned variable will be computed from the other two through the declared saturation-capillary pressure functions.

No initial saturation options are allowed for the three-phase, aqueous-NAPL-gas, operational modes (e.g., *Water-Oil, Water-Oil-Air*). For these three-phase operational modes, initial saturations must be declared by specifying initial gas, aqueous, and NAPL pressures. If the initial NAPL pressure is assigned a value below the critical point, which signifies no NAPL, then the initial NAPL pressure is reset within the simulator to the critical pressure. Therefore, conditions without NAPL can be specified by initializing the NAPL pressure to any value below the aqueous pressure. Initial conditions may be declared repeatedly for a node with the last definition being applied. Initial conditions for solutes are expressed in terms of solute per unit volume where the volume can refer to the total node volume (*Volumetric*), the aqueous-phase volume (*Aqueous Volumetric*), the gas-phase volume (*Gas Volumetric*), or the NAPL volume (*NAPL Volumetric*). Solute units are undefined and can be expressed as the user chooses (e.g., Ci, pCi, gm, kg, mol, kgmol). Although units for expressing solute quantity may vary among solutes, units must be consistent for a single solute among all input data entries.

The gradient utilities of the initial condition card are invaluable and should be used to advantage when possible. Gradients to the initial conditions allow the user to specify that the initial value of field variables varies along one or more directions in the physical domain. The initial condition variable assigned to an initial condition domain applies to the node with the lowest *x-, y-,* and *z*-direction indices. If nonzero gradient values are specified then the initial condition values will vary according to the gradients specified for each physical direction. Default values for the initial condition gradients are zero, indicating no variation. Gradient values are applicable only over the state initial condition domain. An example application of the initial condition gradient utility occurs for problems which start with hydrostatic conditions. For these problems the pressure at the lowest *z*-direction node positions would be entered with a *Z-Dir. Gradient* that equaled minus the fluid's density times the acceleration of gravity. For water at 20 C, a *z*–direction gradient of -9793.5192 1/m will yield constant head conditions, which would be equivalent to hydrostatic conditions. This approach could be used to locate the position of a

water table under no-recharge equilibrium conditions given the rock/soil saturation function properties and knowledge of a single pressure using an *Initial Condition* execution mode simulation.

4.5.22 Boundary Conditions Card

This card allows the user to control the simulation by defining time varying boundary conditions. This card is optional, but is generally necessary to simulate a particular problem. Boundary conditions may be applied to any boundary surface or surface dividing active and inactive nodes. By default all undeclared boundary surfaces have zero flux boundary conditions for both flow and transport. Boundary conditions may be applied only to surfaces of active nodes. To apply a boundary condition to a boundary surface, the surface is referenced by the adjacent active node and a direction with respect to the adjacent node. To apply a boundary condition to a surface dividing an active and inactive node, the surface is referenced by the active node and the direction to the inactive node with respect to the active node. Boundary conditions are time varying. The user is not allowed to assign multiple boundary conditions to a boundary surface during the same time period, but multiple boundary conditions can be applied to a boundary surface over different time periods. The simulator controls time steps to agree with time transitions in boundary conditions.

Application of boundary conditions requires an appropriate conceptualization of the physical problem and translation of that conceptualization into boundary condition form. The variety of boundary condition types available in the simulator should afford the user with the flexibility to solve most subsurface flow and transport problems. The boundary condition card reader within the simulator performs limited error checking on the boundary condition inputs. An error free boundary condition card does not guarantee the user has not created an ill-posed problem or an execution that will successfully converge. For example, a mistake frequently made by users is to specify infiltration rates at the top of a column with positive fluxes. While this input would be perfectly acceptable to the boundary condition input reader, the specified condition would actually withdraw flux from the top of the column since the *z*-axis and *z*-direction flux are positive in the upward direction.

The *Boundary Surface Direction* is specified with respect to the active node adjacent to a boundary surface. For the Cartesian coordinate system the terms west, south, and bottom refer to the negative *x-, y-,* and *z*-directions, respectively, and the terms east, north, and top refer to the positive *x-, y-,* and *z*-directions, respectively. For the cylindrical coordinate system the terms west, south, and bottom refer to the negative *r-,* □*-,* and *z*-directions, respectively, and the terms east, north, and top refer to the positive *r-,* □*-,* and *z*-directions, respectively.

Boundary condition inputs depend on the operational mode, with the required inputs becoming more complex with increasing number of solved equations. Generally, the number of boundary type that must be declared for each boundary equals the number of solved governing flow and transport equations. The *Dirichlet* boundary type is used to specify a field value (e.g., pressure, temperature, or solute concentration) at the boundary surface. The *Neumann* boundary type allows the user to specify a flux (e.g., liquid phase flux, heat flux, or solute flux) at the boundary surface. The *Zero Flux* boundary type is used to impose no flow and/or transport

conditions across the boundary. The *Saturated* boundary type is available only for two-phase conditions and imposes total-liquid saturation conditions (e.g., water table) at the boundary surface. The *Unit Gradient* boundary type imposes hydrostatic conditions across the boundary surface for the specified phase. The *Hydraulic Gradient* boundary type should be applied only to a column or plane of vertical surfaces. With this boundary type the user specifies a fluid phase pressure at the lowest surfaces of a column or row and the simulator then computes fluid phase pressure for the remaining boundary surfaces assuming hydrostatic conditions for the fluid phase. The *Seepage Face* boundary type is similar to a *Hydraulic Gradient* boundary, but is limited to pressure boundaries of the local gas pressure. This boundary type is designed to model an exposed vertical face that seeps liquids. Liquid can enter a seepage face only for phase pressures that exceed the local gas pressure. The *Initial Conditions* boundary type fixes the boundary field variables (e.g., pressure, temperature, or solute concentration) to the initial value of the field variables of the node adjacent to the boundary surface. This boundary type is invariant with time. *Inflow* and *Outflow* boundary types are applicable only to solute and energy boundary conditions. These boundary types consider only advectively transported solute or energy; diffusion transport across the boundary surface is neglected. The solute concentration boundary types (e.g., *Volumetric Concentration*, *Aqueous Conc.*, *Gas Conc.*, and *NAPL Conc.*) are equivalent to Dirichlet boundary types for solute transport. These boundary types differ by their definitions of solute concentration.

Time variations of the boundary conditions are controlled through declaring multiple boundary times. All *Boundary Time* inputs are referenced against the *Initial Time* specified in the *Solution Control* card or obtained from a *restart* file. A boundary condition declared with a single *Boundary Time* implies that the boundary condition is time invariant and the specified *Boundary Time* represents the start time for the boundary condition. Prior to the start time the boundary surface will be assumed to be of type *Zero Flux*. The specified boundary condition will remain in effect from the start time until the execution completion. If a boundary condition is declared with multiple *Boundary Times*, then the first time listed equals the start time, the last time listed equals the stop time, and the intermediate times are transition points. For simulation times outside of the start and stop time limits, *Zero Flux* boundary conditions apply. For simulation times between two *Boundary Times*, linear interpolation of the boundary conditions is applied. Step boundary condition changes can be simulated by defining duplicate *Boundary Times.* The first time would indicate the completion of the previous boundary condition and the second time would indicate the start of the new boundary condition. At the completion of the step boundary condition another set of duplicate *Boundary Time* declarations would be used. Step boundary conditions are convenient methods for introducing slugs of fluids, heat, or solute in conjunction with the *Neumann* boundary type.

Regardless of the boundary type, the boundary condition inputs are used to compute phase saturations, phase relative permeabilities, and physical properties at the boundary surfaces. For the more complex operational modes, especially those modes involving three-phase conditions, it is critical for the user to specify boundary conditions that yield appropriate secondary field variables at the boundary surface. For example, a three-phase system with infiltrating NAPL under ponded conditions could be declared as type *Zero Flux* for the aqueous

phase, *Zero Flux* for the gas phase, and *Dirichlet* for the NAPL. For this boundary condition scenario, the aqueous and gas pressures requested for the *Zero Flux* type boundary are used with the specified NAPL pressure to compute liquid saturation conditions at the boundary surface. If the pressures specified yield zero NAPL saturation, then the NAPL relative permeability would equal zero and no NAPL would infiltrate across the surface, in spite of the ponding conditions.

4.5.23 Source Card

This card allows the user to control sources and/or sinks of mass, energy, and solutes by defining time-varying sources. By definition sinks are negative sources, and sources refer to an influx of mass, energy, or solute into a node. Sources can be specified for interior or boundary nodes and are functionally analogous to *Neumann* type boundary conditions. Sources applied to inactive nodes are not recognized. Sources are time varying; however, unlike boundary conditions multiple sources may be applied to a node during the same time period. The simulator controls time steps to agree with time transitions in sources. Source inputs depend on the operational mode, with the required inputs becoming more complex with increasing number of solved equations. Sinks withdraw mass, energy, or solutes from a node. The physical properties for the fluids withdrawn through sinks equal the properties of the fluids of the node. Sources inject mass, energy, or solutes into a node. The physical properties for fluids injected through sources are computed from the specified input parameters.

Sources of type *Aqueous Volumetric* and *Aqueous Mass* inject aqueous fluid. If specified, dissolved air and oil will be injected with the aqueous fluid. For nonisothermal problems the injected fluid would also transport an amount of heat into the node based on the enthalpy of the entering aqueous fluid. Sources of type *Gas Volumetric w/ Mass Fraction*, *Gas Volumetric w/ Relative Humidity*, *Gas Mass w/ Mass Fraction*, and *Gas Mass w/ Relative Humidity* are similar to the aqueous sources, in that mass and heat injected into the node would include contributions from water vapor, air, and oil vapors. The source type suffix *w/ Mass Fraction* indicates that water vapor, air, and vapor concentrations in the gas phase will be specified through inputs of mass fractions for these quantities. Similarly, the source type suffix *w/ Relative Humidity* indicates that water vapor, air, and oil vapor concentrations in the gas phase will be specified through inputs of relative humidity for the water and oil. Sources of type *NAPL Volumetric* and *NAPL Mass* inject NAPL. Because dissolution of air and water in the NAPL phase is neglected by the simulator, these sources represent pure NAPL. Energy sources are specified as type *Power* or *Power Density*, where *Power* type sources inject energy per unit time and *Power Density* type sources inject energy per unit time per node volume. *Solute* sources inject solute mass in the assumed units for solute (e.g., Ci, pCi, gm, kg, mol, kgmol).

Well sources model production or injection wells using a nonlinear well model. Positive flow rates are considered as injection wells and negative flow rates are considered as production wells. The well model essentially computes the depth of liquid (water) in the well through a Newton-Raphson iteration scheme that balances flux across the well casing with the source pumping rate. For injection wells, the depth of water in the well is limited by a well bottom pressure constraint, expressed in pressure units. For pumping wells the contribution from

seepage flow is ignored. Wells are vertical features in the STOMP simulator and must be defined over a contiguous array of vertical nodes. By default well bore storage is considered in the well model and the well volume, as defined by the *Borehole Radius* is removed from the node volume. Both of these features can be ignored by including the key words *No Storage* and *No Volume* in the *Source Type Option*.

Time variations of sources are controlled through declaring multiple source times. All *Source Time* inputs are referenced against the *Initial Time* specified in the *Solution Control* card or obtained from a *restart* file. A source declared with a single *Source Time* implies that the source is time invariant and the specified *Source Time* represents the start time for the source. Prior to the start time the source will be zero, and from the start time to execution completion the source will be as specified. If a source is declared with multiple *Source Times*, then the first time listed equals the start time, the last time listed equals the stop time, and the intermediate times are transition points. For simulation times outside of the start and stop time limits, zero source conditions apply. For simulation times between two *Source Times*, linear interpolation of the sources is applied. Step source changes can be simulated by defining duplicate *Source Times* the first time would indicate the completion of the previous source and the second time would indicate the start of the new source. At the completion of the step source another set of duplicate *Source Time* declarations would be used. Step sources are convenient methods for introduction slugs of fluids, heat, or solute into an interior node.

4.5.24 Output Control Card

This card allows the user to control output written to the *output* file, *plot* file, and *screen* (i.e., standard input/output device (STDIO)). The *output* file contains an interpreted and reformatted version of the *input* and simulation results for selected variables at selected *reference nodes* over the simulation period. The *plot.n* file contains values of geometric parameters and selected variables for the entire computational domain (both active and inactive nodes) at selected simulation times. A *plot.n* file will always be generated at the conclusion of an execution. The output to the STDIO primary comprises the reference node variable results versus simulation time and/or time step. It is recommended that the user request *screen* output, because well chosen output maybe invaluable in tracking the simulation progress and identifying possible input errors. If a suite of repetitive simulations is being performed, then *screen* output can be reduced to minimum values. As with other input cards, output options are dependent on the operational mode. With respect to this card, output options primarily refer to computed field and flux variables.

Reference node output is generated by selecting reference nodes and output variables. The user may request any number of reference nodes, but reference node output was primarily designed for tracking the time evolution of selected variables at key nodes of interest. Reference nodes are defined with three indices, which indicate the *x-, y-,* and *z*-direction coordinates of the node. Node numbering in the simulator increments in the order i, j, and k, where the indices refer to the *x-, y-,* and *z*-directions, respectively, for Cartesian coordinate systems and *r-, □-,* and *z*-directions, respectively, for cylindrical coordinate systems. *Reference Node Screen Output Frequency* and *Reference Node Output File Frequency* are parameters which indicate how often

with respect to time step reference node output will be written to STDIO and the *output* file, respectively. A frequency value of 1 indicates reference node output occurs every time step, whereas a frequency value of 10 indicates reference node output occurs every 10 time steps. The user has control over the output time and length units and the number of significant digits reported to the various output media. Unless declared through the *Output Time Units* or *Output Length Units* input items, values for time and lengths recorded to the output media will be expressed in units of seconds and meters, respectively. These inputs allow the user customize time- and length-scale units to those most appropriate to the solved problem. *The Screen Significant Digits*, *Output File Significant Digits*, and *Plot File Significant Digits* input items allow the user to customize the number of significant digits which appear in field and flux variable results written to the STDIO, *output* file, and *plot.n* file, respectively. Default values for the number of significant digits is 5, and the minimum number of significant digits is 4. Field and flux variables output for the reference nodes are selected from the list shown for each operational mode in the Appendix. The same list of variables for each operational is available for output to the *plot.n* files. Output units for all variables with units can be specified immediately following the variable name. Variables without units require a *null* entry for the variable units. *Null* entries for variables with units yield default output units, which are expressed in SI units.

Plot files are written at the conclusion of an execution, by default, and at each requested *Plot File Output Tim*e. Plot files contain geometry data and selected field and flux variable results for every node in the computational domain. These files represent a "snapshot" of the simulation at a certain point in time. Requests for *Plot File Output Times* can be specified with user defined units. A *restart.n* file is generated with every *plot.n* file. Both *restart.n* and *plot.n* files are suffixed with an file name extension of a dot followed by an integer (e.g., plot.567, restart.32). The extension integer corresponds with the time step for which the file was written. Both *plot.n* and *restart.n* files are written at the conclusion of a time step. Field and flux variables recorded to *plot.n* files are selected from the list shown for each operational mode in the Appendix (refer to the reference node variable options). Output units for all variables with units can be specified in the input item immediately following the variable name. Variables without units require a *null* entry for the variable units. *Null* entries for variables with units yield default output units, which are expressed in SI units.

4.5.25 Surface Flux Card

This card allows the user to define surfaces to track fluxes of fluid mass, fluid volume, heat, and/or solutes. A surface defined with this card can be composed of rectangular areas of coplanar surfaces on exterior boundaries or between interior nodes. Output from the surface flux integration routines is written to the *surface* file and contain flux rate and integral data for each defined surface at every time step. The types of fluxes that can be tracked depend on the operational mode. Declaration of surfaces is similar to defining boundary condition surfaces. Surfaces are defined by referencing a group of coplanar nodes and a surface direction with respect to the nodes. For the Cartesian coordinate system the terms west, south, and bottom refer to the negative *x-, y-,* and *z*-directions, respectively, and the terms east, north, and top refer to the positive *x-, y-,* and *z*-directions, respectively. For the cylindrical coordinate system the terms

west, south, and bottom refer to the negative *r*-, □-, and *z*-directions, respectively, and the terms east, north, and top refer to the positive *r*-, □-, and *z*-directions, respectively. For example, a surface to track the flux rate and integral of a particular solute entering the water table could be defined for a simulation with saturated conditions along the bottom boundary surface by referencing the node group along the bottom of the computational grid and defining the *Surface Flux Orientation* as *Bottom*. One surface flux rate and integral value is computed for each defined surface and represents the summation of surface flux contributions from the individual surfaces in the coplanar group of surfaces.

4.5.26 Observed Data Card

Parameter estimation with the STOMP simulator is executed using inverse modeling techniques in conjunction with UCODE (Poeter and Hill, 1998), a computer code for universal inverse modeling. In the current version of the STOMP simulator inverse modeling is only an option for the *Water* or *Fluid* operational modes. Inverse modeling involves repetitive forward modeling; where, each forward realization differs by increments in the parameters of interest. Simulation results are compared against observed data to compute error. This card is divided into two sections: one for specifying observed data parameters and the other for entering field (laboratory) observation data. Observed data can be field parameters (i.e., aqueous pressure, aqueous saturation, aqueous moisture content, solute concentrations), flux rate parameters (i.e., aqueous volumetric flux, solute flux), or flux integral parameters (i.e., aqueous volumetric flux integral, solute flux integral). Observed data is defined by the parameter type, physical location, statistical index, and time and space weighting factors. Physical locations for field parameters can be specified as *x*, *y*, *z* (*r*, □, *z*) coordinates or an *i*, *j*, *k* node index; where, the node indexing refers to the centroid of the grid cell. Flux rate and integral parameters can only be specified by defining surfaces as with the *Surface Flux Card* (i.e., a surface direction and a domain of *i*, *j*, *k* indexed nodes). As with the *Surface Flux Card*, surfaces can span over a range of nodes and the reported surface flux values will be the summed quantity (i.e., the flux rate or integral across the entire surface). In the first section of the card observed data parameters are defined and located within the computational domain. In the second section of the card, experimental data for the defined parameter is read, either directly from the input file or from an external file. The observed experimental data includes a time and observed data value; where, the time corresponds with the simulation time. STOMP does not compute or assess the error between the computed observed data and the experimental observed data, it simply records these data to a file. For reference observed data, specified as *x*, *y*, *z* (*r*, □, *z*) coordinates, tri-linear interpolation is used to compute the observed data parameter at the specified physical location.

4.5.27 UCODE Control Card

This card allows users to control UCODE executions through the STOMP input file. Parameters and options read from this card are written to external files that are then read and used by UCODE. This card is offered as a convenience to users executing UCODE in conjunction with STOMP, but is not required if the user prefers to externally created the controlling files for

UCODE. The control parameters read on this card are written to the external file *out_uc.sto*. As with other STOMP input, this card uses comma delimited fields. Output written to *out_uc.sto*, however, is properly formatted for UCODE input.

Table 4.1. Required and Optional Input Cards

Operational Mode	*Card Name*	*Card Status*
All	Aqueous Relative Permeability	Required
	Boundary Conditions	Optional
	Grid	Required
	Hydraulic Properties	Required
	Inactive Nodes	Optional
	Initial Conditions	Optional
	Mechanical Properties	Required
	Output Control	Optional
	Rock/Soil Zonation	Required
	Saturation Function	Required
	Simulation Title	Required
	Surface Flux	Optional
	Solute/Fluid Interactions	Optional
	Solute/Porous Media Interactions	Optional
	Solution Control	Required
	Source	Optional
	X-Direction Aqueous Permeability	Optional
	Y-Direction Aqueous Permeability	Optional
	Z-Direction Aqueous Permeability	Optional
Operational Mode	*Card Name*	*Card Status*
Water	Observed Data	Optional
	UCODE Control	Optional
Operational Mode	*Card Name*	*Card Status*
Water-Air	Gas Relative Permeability Function	Required
Operational Mode	*Card Name*	*Card Status*
Water-Air-Energy	Atmospheric Conditions	Optional
	Gas Relative Permeability Function	Required
	Plant	Optional
	Thermal Properties	Required
Operational Mode	*Card Name*	*Card Status*
Water-Oil	Dissolved-Oil Transport	Optional
	NAPL Relative Permeability Function	Required
	Oil Properties	Required
Operational Mode	*Card Name*	*Card Status*
Water-Air-Oil	Dissolved-Oil Transport	Optional
	Gas Relative Permeability Function	Required
	NAPL Relative Permeability Function	Required
	Oil Properties	Required
Operational Mode	*Card Name*	*Card Status*
Water-Air-Oil-Energy	Dissolved-Oil Transport	Optional
	Gas Relative Permeability Function	Required
	NAPL Relative Permeability Function Oil	Required
	Properties	Required
	Thermal Properties	Required

Table 4.1. Cont'd

Operational Mode	*Card Name*	*Card Status*

Water-Oil-Dissolved Oil	Dissolved Oil Transport NAPL Relative Permeability Function Oil Properties	Required Required Required
Operational Mode	*Card Name*	*Card Status*
Water-Salt	Salt Transport	Required
Operational Mode	*Card Name*	*Card Status*
Water-Air-Salt	Gas Relative Permeability Function Salt Transport	Required Required
Operational Mode	*Card Name*	*Card Status*
Water-Air-Salt-Energy	Gas Relative Permeability Function Salt Transport Thermal Properties	Required Required Required
Operational Mode	*Card Name*	*Card Status*
Water-CO2-NaCl	Gas Relative Permeability Function Salt Transport	Required Required
Operational Mode	*Card Name*	*Card Status*
Water-CO2-NaCl-Energy	Gas Relative Permeability Function Salt Transport Thermal Properties	Required Required Required

Table 4.2. Recognized Units

Notation	Description	SI Equivalent	Conversion to SI	Base Units
1	one	N/A	N/A	N/A
aqu	aqueous phase	N/A	N/A	N/A
aqueous	aqueous phase	N/A	N/A	N/A
atm	atmosphere	Pa	101325.0	$M / L T^2$
bar	bar	Pa	$1. \times 10^5$	$M / L T^2$
btu	BTU	J	1054.4	$M L^2 / T^2$
c	Celcius	C	1.	K
cal	calorie	J	4.184	$M L^2 / T^2$
ci	Curie	N/A	N/A	N/A
cm	centimeter	m	$1. \times 10^{-2}$	M / L T
cp	centipoise	Pa s	$1. \times 10^{-3}$	M / L T
d	day	s	1/86400.	T
darcy	darcy	m^2	9.8697×10^{-13}	L^2
day	day	s	1/86400.	T
debyes	Debyes	N/A	N/A	$(M/L T^2)^{1/2}$
deg	degrees	rad	0.017453	N/A
degree	degree	rad	0.017453	N/A
dyn	dynes	N	$1. \times 10^{-5}$	$M L / T^2$
dynes	dynes	N	$1. \times 10^{-5}$	$M L / T^2$
f	Farhenheit	C	(f-32) x (5/9)	K
ft	foot	m	0.3048	L
furlong	furlong	m	2.01168×10^2	L
g	gm	kg	$1. \times 10^{-3}$	M
gal	gallon	m^3	3.7854×10^{-3}	L^3
gas	gas phase	N/A	N/A	N/A
gm	gm	kg	$1. \times 10^{-3}$	M
gram	gm	kg	$1. \times 10^{-3}$	M
hc	hydraulic cond.	N/A	1.03910×10^{-7}	N/A
hour	hour	s	1/3600.	T
hp	horsepower	W	745.7	$M L^2 / T^3$
hr	hour	s	1/3600.	T
in	inch	m	2.54×10^{-2}	L
j	Joule	J	1.	$M L^2 / T^2$
k	kelvin	C	1.	K
kg	kilogram	kg	1.	M
kgmol	kilogram-mole	kgmol	1.	Mo
l	liter	m3	$1. \times 10^{-3}$	L^3
langely	Langley	J/m^2	4.186×10^{-6}	M / T^2

Table 4.2. Cont'd

Notation	Description	SI Equivalent	Conversion to SI	Base Units

lb	pound	kg	0.45359	M
lbf	pound-mass force	N	4.4482	$M L / T^2$
lbm	pound mass	kg	0.45359	M
lbmol	pound-mol	kgmol	0.45359	Mo
liq	liquid	N/A	N/A	N/A
liter	liter	m^3	$1. \times 10^{-3}$	L^3
m	meter	m	1.	L
mg	milligram	kg	1.0×10^{-6}	M
min	minute	s	1/60.	T
ml	milliliter	m^3	1.0×10^{-6}	L^3
mm	millimeter	m	1.0×10^{-3}	L
mol	mole	kgmol	1.0×10^{-3}	Mo
mole	mole	kgmol	1.0×10^{-3}	Mo
n	Newton	N	1.	$M L / T^2$
napl	NAPL	N/A	N/A	N/A
newton	Newton	N	1.	$M L / T^2$
oil	oil	N/A	N/A	N/A
p	Poise	Pa s	1.0×10^{-1}	M / L T
pa	Pascal	Pa	1.0	$M / L T^2$
pci	picoCuries	N/A	N/A	N/A
plant	plant	N/A	N/A	N/A
psf	lbs/ft^2	Pa	4.7880556×10^1	$M / L T^2$
psi	lbs/in^2	Pa	6894.8	$M / L T^2$
r	Rakine	K		K
rad	radian	rad	1.	N/A
radian	radian	rad	1.	N/A
rod	rod	m	5.0292	L
s	second	s	1.	T
sec	second	s	1.	T
slug	slug	kg	14.594	M
sol	solid phase	N/A	N/A	N/A
solid	solid phase	N/A	N/A	N/A
voc	VOC phase	N/A	N/A	N/A
w	Watt	W	1.	$M L^2 / T^3$
water	water	N/A	N/A	N/A
week	week	s	1/604800.	T
wh	water head	Pa	9.79353×10^3	$M / L T^2$
wk	week	s	1/604800.	T
yd	yard	m	0.9144	L
year	year	s	1/31557600.	T
yr	year	s	1/31557600.	T

Table 4.3. Default Interfacial Averaging Options

Field Variable	Interfacial Averaging
Aqueous Density	Upwind
Aqueous Relative Permeability	Upwind
Aqueous Viscosity	Harmonic
Dissolved Air Diffusion	Harmonic
Dissolved Oil Diffusion	Harmonic
Dissolved Salt Diffusion	Harmonic
Gas Density	Upwind
Gas Relative Permeability	Upwind
Gas Viscosity	Harmonic
Hydraulic Dispersion	Harmonic
Intrinsic Permeability	Harmonic
NAPL Density	Upwind
NAPL Relative Permeability	Upwind
NAPL Viscosity	Harmonic
Oil Vapor Diffusion	Harmonic
Solute Diffusion	Harmonic
Thermal Conductivity	Harmonic
Water Vapor Diffusion	Harmonic

Table 4.4. STOMP Initial Conditions Default Values

Initial Field Variable	Symbol	Default Value
temperature	T	20.D+0
pressure of phase j	P_j	101325.D+0
saturation of phase j	s_j	0.D+0
mole fraction of component i in phase j	$\square_j$	0.D+0
solute concentration in phase $\square$	$C_\square$	0.D+0
salt concentration in phase $\square$	$S_\square$	0.D+0

5. Compilation and Execution

5.1 Introduction

In its native form, the STOMP simulator is a collection of files, which contain either global routines or those associated with a particular operational mode. For users outside of the Pacific Northwest National Laboratory, the STOMP simulator is distributed as assembled source coding for a particular operational mode, with associated include files, modules, example input files and required external packages (e.g., SPLIB). Assembly of the source coding occurs through the *make* utility (Talbott, 1988) before the code is distributed. For users within the Laboratory, the *make* utility additionally can be used to directly create an executable. Except for external packages, the STOMP simulator is coded in Fortran. Distributing an open source allows users to read and modify the simulator; hopefully, promoting an open exchange of scientific ideas. The penalty of an open source; however, is that the user is responsible for compiling and linking the source code to generate an executable. This inherently assumes that the user has a Fortran compiler and is familiar with its use for generating code. Advanced users, interested in modifying the code, should additionally be familiar, if not skilled with using a symbolic debugger (often provided with the Fortran compiler). The unassembled STOMP source is coded in a combination of Fortran 77 and Fortran 90. With respect to memory allocation, the assembled source code can be configured in two forms: static and dynamic memory.

For those users with only access to a Fortran 77 compiler, (e.g., g77, f77, pgf77) the code must be configured in static memory form. In this form the source code will include a *parameters* and a *commons* file. As Fortran 77 is unable to dynamically allocate memory, the user is responsible for editing the *parameters* file to define array dimensions to statically allocate memory during compilation. For users with access to a Fortran 90 compiler (e.g., f90, pgf90, ifc), the source code can be configured for dynamic memory allocation. In the dynamic memory form, both the *parameters* and *commons* files are replaced with a series of Fortran 90 modules, in the file *allo.f*. As described below this module file must be compiled prior to compiling the source code. When configured under the dynamic memory option a utility named *step* is included in the source code. When incorporated, the *step* utility becomes the first subroutine called and reads a STOMP *input* file to determine dimensioned array requirements. These values (i.e., parameters) are then used to allocate memory for the dimensioned arrays, via a call to the subroutine *alloc*. Subroutine *alloc* makes a series of memory allocations and memory checks. If STOMP attempts to allocate more memory than available on the computer, the simulation stops and an error message is printed. Memory allocation under the dynamic memory option only occurs during the execution startup, (i.e., memory is never deallocated until the execution stops). The dynamic memory option is generally preferred as it allows the user to execute problems without having to create a *parameters* file and recompile the code with changes in the *input* file. Table 5.1 summarizes the memory allocation options.

Table 5.1. Memory Options

Memory Option	Fortran Compiler	Include Files	Module File	Notes
static	f77	*commons* *parameters*		• user generated *parameters* • recompilation w/ input change
dynamic	f90		*allo.f*	• no recompilation w/ input change • greater 1GB memory on Linux

5.2 Compilation

Compiling the source code into an executable differs between operating systems, compilers, and memory options. This section will describe the differences between memory options, using the UNIX[1] as an example operating system. For the static memory option, the following files will be provided with the assembled source coding: *stomp#_[sp,bd].f*, *commons*, and *parameters*; where the # in the filename *stomp#_[sp,bd].f* refers to the operational mode number and the solver options *sp* and *bd* refer to the conjugate gradient or banded solvers, respectively, (e.g., *stomp4_bd.f* is the source code for the *Water-Oil* operation mode with the banded solver. To create an executable on a UNIX system, assuming the *commons* and *parameters* files were in the same directory as the source code, the user would issue the following command:

```
f77 -I . -o stomp4_bd.e stomp4_bd.f
```

For the conjugate gradient solver, the compiler must link to the *splib* library, and the compilation command would be:

```
f77 -I . -o stomp4_sp.e stomp4_sp.f $SPLIB_PATH/splib.a
```

where, `$SPLIB_PATH` is the path to the *splib* library. For the dynamic memory option, the following files will be provided with the assembled source coding: *stomp#_[sp,bd].f*, and *allo.f*. To create an executable on a UNIX system, assuming the *commons* and *parameters* files were in the same directory as the source code, the user would issue the following command:

```
f90 -c allo.f
f90 -c stomp4_bd.f
f90 -o stomp4_bd.e allo.o stomp4_bd.o
```

[1] UNIX is a registered trademark of AT&T Information Systems

For the conjugate gradient solver, the compiler must link to the *splib* library, and the compilation command would be:

```
f90 -c allo.f
f90 -c stomp4_bd.f
f90 -o stomp4_bd.e allo.o stomp4_bd.o $SPLIB_PATH/splib.a
```

where, `$SPLIB_PATH` is the path to the *splib* library.

5.3 Parameters

For the static memory option, the user is responsible for creating a *parameters* file. This section describes the format and contents of the *parameters* file, which is required to compile and execute the STOMP simulator. Parameters are used by the FORTRAN programming language and compilers to statically allocate memory for storage of variables. The FORTRAN 77 language is unable to allocate memory dynamically, therefore all memory storage requirements must be defined at compilation time. No execution errors will occur if the memory allocated is greater that required by the simulation, unless the memory requirements exceed the computer's capabilities. Unless necessary, the user should avoid executing simulations which require the use of virtual memory. The time required to swap data between the virtual memory storage device and the active memory typically yields poor execution speeds. The STOMP simulator requires two types of parameters (declared and computed) to be defined, prior to compilation. The user is responsible for properly assigning all of the declared parameters. Declared parameters are assigned by modifying the *parameters* file supplied with the STOMP simulator using a text editor (word processor) or by creating a new *parameters* file. The equations for the computed parameters must be included in each *parameters* file after the declared parameters. The parameter definitions given in this manual represent minimum acceptable values. All declared parameters, except for switch type parameters, must have minimum values of 1. Undersized parameters will generally yield execution errors, which may or may not be detected by the system. Oversized parameters are permissible, but can result in excessive memory allocation.

5.3.1 Solution/Simulation Parameters

The *LNOTES* parameter equals the number of lines in the *Simulation Notes* field and should equal the variable *Number of Simulation Note Lines* on the *Simulation Title Card*. The *LEPD* parameter equals the number of execution periods and should equal the variable *Number of Execution Periods* on the *Solution Control Card*. A simulation with 7 lines of simulation notes and 4 execution periods would have a solution/simulation parameter set that appeared as

```
PARAMETER(LNOTES=7, LEPD=4)
```

5.3.2 Computational Domain Parameters

The *LFX*, *LFY*, and *LFZ* parameters equal the number of nodes in the x-, y- and z-directions, respectively, for Cartesian coordinate systems or the number of nodes in the radial-, -azimuthal, and z-directions, respectively, for cylindrical coordinate systems. These parameters should equal the variables *X-Dir. Nodes*, *Y-Dir. Nodes*, and *Z-Dir. Nodes* on the *Grid Card*. The parameter *LAN* equals the number of active nodes a should equal the total number of nodes minus the number of inactive nodes. The parameter *LAD* equals the number of active dimensions, where active dimensions are coordinate directions with more than 1 node. The number of active dimensions should be a number between 1 and 3, inclusive. The parameter *LMNP* equals the minimum plane size and should equal the minimum of the following three products: *LFX*LFY*, *LFY*LFZ*, and *LFZ*LFX* for the banded solver and 1 for the conjugate gradient solver. A simulation, using the banded solver with two active dimensions, having 22, 1, and 103 nodes in the x-, y-, and z- directions, respectively, and 51 inactive nodes would have a computational domain parameter set that appeared as

```
PARAMETER(LFX=22, LFY=1, LFZ=103)
PARAMETER(LAN=2215, LAD=2, LMNP=22)
```

Whereas, for the conjugate gradient solver the computational domain parameters would appear as

```
PARAMETER(LFX=22, LFY=1, LFZ=103)
PARAMETER(LAN=2215, LAD=2, LMNP=1)
```

5.3.3 Operation Mode Switch Parameters

The parameters *LT*, *LL*, *LG*, *LN*, and *LC* are on/off switches for the energy, water mass, air mass, oil mass, and solute conservation equations. For on/off switch type parameters, a value of 1 indicates the equation is solved and a value of 0 indicates the equation is unsolved. The parameter *LFW* is an on/off switch for freezing conditions. The parameter *LS* is an on/off switch for coupled salt- or NaCl-mass transport. The parameter *LD* is an on/off switch for the dissolved-oil mass conservation equation. The parameter *LPC* is an on/off switch for the kinetic partitioning tracer equation. The parameter *LALC* is an on/off switch for the alcohol mass balance equation. The parameter *LWELL* is an on/off switch for the fully coupled well models. The parameter *LDCO2* is an on/off switch for the CO_2 mass conservation equation. The switch combinations set with these parameters should agree with the operational mode. A simulation using the *Water-Air w/ Transport* operational mode would have a operational mode switch parameter set that appeared as

```
PARAMETER(LT=0, LL=1, LG=1, LN=0, LC=1, LFC=0, LS=0, LD=0)
PARAMETER(LPC=0, LALC=0, LWELL=0, LDCO2=0)
```

5.3.4 Linear System Solver Switch Parameters

The parameters *LBD* and *LSP* are on/off switches for the banded and conjugate gradient linear system solvers. For these switch type parameters, a value of 1 indicates the solver is active and value of 0 indicates the solver is inactive. Only one solver can be active for a simulation, either the banded matrix or the conjugate gradient solver. In the static memory mode, these parameters switches are the only means the user has for declaring a linear equation solution scheme. A simulation that used the banded matrix linear system solver would have a linear system solver parameter set that appeared as

```
PARAMETER(LBD=1, LSP=0)
```

5.3.5 Boundary Condition Parameters

The parameters *LBC* and *LBTM* equal the number of boundary surfaces and maximum number of boundary condition times. A boundary condition domain as declared on the *Boundary Conditions Card* may contain more than one boundary surface. The parameter *LBC* indicates the total number of boundary surfaces, not the number of boundary condition domains. Although they may refer to the same computational domain surface, boundary surfaces declared in one boundary condition domain differ from those declared in another. The maximum number of boundary condition times equals the maximum number of times declared for all boundary condition domains. A simulation with three boundary condition domains, the first with 6 boundary surfaces and 13 boundary condition times, the second with 2 boundary condition surfaces and 26 boundary condition times, and the third with 34 boundary condition surfaces and 2 boundary condition times, would have a boundary parameter set that appeared as

```
PARAMETER(LBC=42, LBTM=26)
```

5.3.6 Source and Well Parameters

The parameters *LSR* and *LSTM* equal the number of source domains and maximum number of source times. Unlike the boundary condition parameters, a single source may contain more than one node, but is only counted as once with respect to the parameter *LSR*. The maximum number of source times equals the maximum number of times declared for all sources. The parameters *LNW*, *LNWT*, and *LNWS* refer to fully-coupled wells and are used to define the number of possible wells, the number of well times and the number of well-screen intervals, respectively. A simulation with three source domains: the first with 6 nodes and 13 source times, the second with 2 sources and 26 source times, and the third with 34 sources and 2 source times; and no coupled wells would have a source parameter set that appeared as

```
PARAMETER(LSR=3, LSTM=26, LNW=1, LNWT=1, LNWS=1)
```

5.3.7 Rock Type and Solute Transport Parameters

The parameters *LRC* and *LSOLU* equal the number of rock types and number of transported solutes. The number of rock types should equal the number of rock/soil types listed on the *Rock/Soil Zonation Card*. Because one rock/soil type can define the rock/soil type for more than one rock/soil zonation domain the number of rock/soil types does not equal the *Number of Rock/Soil Zonation Domains* declared on the *Rock/Soil Zonation Card*. Each rock/soil type should have a unique name. For *IJK Indexing*, the parameter *LRC*, should equal 1. The number of solutes should equal the number of solutes defined on the *Solute/Fluid Interactions Card*. Each solute should have a unique name. A simulation with 14 rock/soil zonation domains, 3 rock/soil types, 7 solutes for the *Water-Air w/ Transport* operational mode would have a rock types and solutes parameter set that appeared as

```
PARAMETER(LRC=3, LSOLU=7)
```

5.3.8 Output Parameters

The parameters *LREF*, *LPTM*, and *LSF* equal the number of reference nodes, number of plot file output times, and the number of flux surfaces, respectively. The number of reference nodes should equal the *Number of Reference Nodes* input on the *Output Control Card*. The number of plot file output times should equal the *Number of Plot File Times* input on the *Output Control Card*. The *plot.n* file which is generated, by default, at the conclusion of an execution should not be counted as a plot file time. The number of flux surfaces should equal the *Number of Surface Flux Inputs* input on the *Surface Flux Card*. A simulation with 4 reference nodes, 6 plot file output times (other than the plot file generated at the conclusion of the execution), and 23 flux surfaces would have an output parameter set that appeared as

```
PARAMETER(LREF=4, LPTM=6, LSF=23)
```

5.3.9 Observed Data Parameters

The parameters *LOBDT*, and *LOBDS* equal the number of observed data types and observed data samples, respectively. The number of observed data types should equal the *Number of Observed Data Types* input on the *Observed Data Card*. The number of observed data samples should equal the maximum *Number of Observed Data Samples* input on the *Observed Data Card*. A simulation with 3 observed data types: the first with 21 observed data samples, the second with 16 observed data samples and the third with 9 observed data samples; would have an output parameter set that appeared as

```
PARAMETER(OBDT=3, LOBDS=21)
```

5.3.10 Relative Permeability and Chemical Reaction Parameters

The parameters *LTBL*, and *LCHEM* equal the total number of table entries and the number of chemical reactions, respectively. The number of table entries should equal the sum of *Number of Table Entries* on the *Saturation Function*, *Aqueous Relative Permeability Function*, and *Gas Relative Permeability Function Cards*. Tabular NAPL relative permeability functions are not recognized by the STOMP simulator. A *Water* operational mode simulation with 1 chemical reaction, 2 rock/soil types, 34 tabular entries for the saturation function for the first rock/soil type, a nonhysteretic saturation function for the second rock/soil type, 27 tabular entries for the aqueous relative permeability for the first rock/soil type, and 31 tabular entries for the aqueous relative permeability for the second rock/soil type, would have a relative permeability and chemical reaction parameter set that appeared as

```
PARAMETER(LTBL=92, LCHEM=1)
```

5.3.11 Local Uniform Grid Refinement Parameters

The parameters *LUGR* and *LGRL* are used to size dimensioned variables associated with the local uniform grid refinement capabilities of the simulator. The parameter *LUGR* is an on/off switch for local uniform grid refinement; where a value of 1 indicates that local uniform grid refinement is active and a value of 0 indicates no local uniform grid refinement. The parameter *LGRL* controls the maximum number of grid levels used in the grid refinement scheme. Local uniform grid refinement currently is available only for the *Water-Salt* operational mode. For other operational modes these parameters should be set as

```
PARAMETER(LUGR=0, LGRL=1)
```

5.3.12 CO_2 Property Table, Plant, and Atmospheric Condition Parameters

The parameters *LPTA*, *LPLANT*, and *LATM* are used to size dimensioned variables associated with CO_2 property tables, the number of plant varietals, and the number of atmospheric condition times, respectively. The parameter *LPTA* is an on/off switch for CO_2 property tables; where a value of 1 indicates that CO_2 property tables will be used and a value of 0 indicates no CO_2 property tables. Parameter *LPTA* should be set to 1 for simulations involving the *Water-CO2-NaCl* and *Water-CO2-NaCl-Energy* operational modes. The parameter *LPLANT* should equal the *Number of Plant Varietals* on the *Plant Card*. The parameter *LATM* should equal the *Number of Atmospheric Condition Times* on the *Atmospheric Conditions Card*. For

simulations not using the *Water-CO2-NaCl* or *Water-CO2-NaCl-Energy* operational modes and not using the evaporation-transpiration capabilities of the simulator, these parameters should be set as

```
PARAMETER(LPTA=0, LPLANT=1, LATM=1)
```

5.3.13 Computed Parameters

Computed parameters are arithmetic combinations of the declared parameters and must follow the declared parameters in the *parameters* file. Computed parameters related to the conjugate gradient linear system solver are declared within the source code. Definitions of the computed parameters in the parameters file are given in Table 5.. The computed parameter set for all simulations appears as

```
PARAMETER(LANW=LAN+LWELL*LFZ*LNW)
PARAMETER(LUK=LT+LL+LG+LN+LS+LD+LALC+LDCO2)
PARAMETER(LPH=LL+LG+LN
PARAMETER(LCMP=LL+LS+LD)
PARAMETER(LSALC=LS+LALC)
PARAMETER(LFXY=LFX*LFY)
PARAMETER(LFYZ=LFY*LFZ)
PARAMETER(LFZX=LFZ*LFX)
PARAMETER(LFD=LFX*LFY*LFZ)
PARAMETER(LHBW=LUK*LBD*LMNP+LUK-1)
PARAMETER(LJA=LBD+LSP*LANW*LUK)
PARAMETER(LJB=LSP*(2*LAD+1)*LANW*LUK*LUK+LBD)
PARAMETER(LJC=LSP*(LANW*LUK+1)+LBD)
PARAMETER(LJD=LBD*(3*LHBW+1)+LSP)
PARAMETER(LJE=LBD*LANW*LUK+LSP)
PARAMETER(LJF=LANW*LUK)
PARAMETER(LJG=LBD*(3*LHBW+1)+LSP*LANW*LUK)
PARAMETER(LJH=LBD*LANW*LUK+LSP*7*LUK+LSP*3*LUK*LWELL)
PARAMETER(LJI=LBD*LANW*LUK+LSP)
PARAMETER(LJJ=LBD*LANW*LUK+LSP)
PARAMETER(LJK=LBD+LSP*LANW)
PARAMETER(LJL=7)
PARAMETER(LJM=LBD+LSP*(2*LAD+1)*LANW)
PARAMETER(LJN=LBD+LSP*(LANW+1))
PARAMETER(LSU=2+(2*(LGRL-1)))
PARAMETER(LSV=(LUK+2)+(2*(LGRL-1)))
PARAMETER(LSFV=(2*LUK+1)+(LGRL-1))
PARAMETER(LFDT=LFD**LT)
PARAMETER(LFDL=LFD**LL)
PARAMETER(LFDG=LFD**LG)
PARAMETER(LFDN=LFD**LN)
PARAMETER(LFDC=LFD**LC)
PARAMETER(LFDI=LFD**LFW)
PARAMETER(LFDS=LFD**LSALC)
PARAMETER(LFDD=LFD**((LD+LDCO2)-(LD*LDCO2)))
```

```
PARAMETER(LFDA=LFD**LALC)
PARAMETER(LSX=(LFX+1)*LFY*LFZ)
PARAMETER(LSY=LFX*(LFY+1)*LFZ)
PARAMETER(LSZ=LFX*LFY*(LFZ+1))
PARAMETER(LSXT=LSX**LT)
PARAMETER(LSXL=LSX**LL)
PARAMETER(LSXG=LSX**LG)
PARAMETER(LSXN=LSX**LN)
PARAMETER(LSXC=LSX**LC)
PARAMETER(LSXS=LSX**LSALC)
PARAMETER(LSXD=LSX**LD)
PARAMETER(LSYT=LSY**LT)
PARAMETER(LSYL=LSY**LL)
PARAMETER(LSYG=LSY**LG)
PARAMETER(LSYN=LSY**LN)
PARAMETER(LSYC=LSY**LC)
PARAMETER(LSYS=LSY**LSALC)
PARAMETER(LSYD=LSY**LD)
PARAMETER(LSZT=LSZ**LT)
PARAMETER(LSZL=LSZ**LL)
PARAMETER(LSZG=LSZ**LG)
PARAMETER(LSZN=LSZ**LN)
PARAMETER(LSZC=LSZ**LC)
PARAMETER(LSZS=LSZ**LSALC)
PARAMETER(LSZD=LSZ**LD)
PARAMETER(LRCT=LRC**LT)
PARAMETER(LRCL=LRC**LL )
PARAMETER(LRCG=LRC**LG)
PARAMETER(LRCN=LRC**LN)
PARAMETER(LRCC=LRC**LC)
PARAMETER(LRCI=LRC**LFW)
PARAMETER(LRCS=LRC**LSALC)
PARAMETER(LRCD=LRC**LD)
PARAMETER(LBCT=LBC**LT)
PARAMETER(LBCL=LBC**LL)
PARAMETER(LBCG=LBC**LG)
PARAMETER(LBCN=LBC**LN)
PARAMETER(LBCC=LBC**LC)
PARAMETER(LBCI=LBC**LFW)
PARAMETER(LBCS=LBC**LSALC)
PARAMETER(LBCD=LBC**LD)
PARAMETER(LBCA=LBC**LALC)
PARAMETER(LBCU=LUK+LPH+LT+2)
PARAMETER(LBCV=LBCU+(LPH**LPC)*LSOLU)
PARAMETER(LOUPV=400+23*(LSOLU))
PARAMETER(LSCHR=18)
PARAMETER(LRPLC=12)
PARAMETER(LRPGC=4)
PARAMETER(LRPNC=1)
PARAMETER(LRPL=4)
PARAMETER(LNWN=LNW*LFZ)
PARAMETER(LSZW=LNW*(LFZ+1))
PARAMETER(LNWV=6)
PARAMETER(LUKW=LUK*(1+LWELL))
```

```
PARAMETER(LFDGR=LFD**LUGR)
PARAMETER(LFXGR=LFX**LUGR)
PARAMETER(LFYGR=LFY**LUGR)
PARAMETER(LFZGR=LFZ**LUGR)
PARAMETER(LBCGR=LBC**LUGR)
PARAMETER(LP_TA=30**LPTA)
PARAMETER(LT_TA=54**LPTA)
```

Table 5.2. Computed Parameter Definitions

Parameter	Definition
LANW	Number of active field and well nodes
LBCA	Number of alcohol boundary condition variable elements
LBCC	Number of solute boundary condition variable elements
LBCD	Number of dissolved boundary condition variable elements
LBCG	Number of air boundary condition variable elements
LBCGR	Number of boundary condition variable elements with grid refinement
LBCI	Number of ice boundary condition variable elements
LBCL	Number of water boundary condition variable elements
LBCN	Number of oil boundary condition variable elements
LBCS	Number of salt/surfactant boundary condition variable elements
LBCT	Number of energy boundary condition variable elements
LBCU	Number of coupled boundary condition variable elements
LBCV	Number of total boundary condition variable elements
LCMP	Number of components
LFD	Number of nodes
LFDA	Number of alcohol field variable elements
LFDC	Number of solute field variable elements
LFDD	Number of dissolved oil field variable elements
LFDG	Number of air field variable elements
LFDGR	Number of nodes with grid refinement
LFDI	Number of ice field variable elements
LFDL	Number of water field variable elements
LFDN	Number of oil field variable elements
LFDS	Number of salt/surfactant field variable elements
LFDT	Number of energy field variable elements
LFXGR	Number of nodes x plane with grid refinement
LFXY	Number of nodes in the xy plane
LFYGR	Number of nodes y plane with grid refinement
LFYZ	Number of nodes in the yz plane
LFZGR	Number of nodes z plane with grid refinement
LFZX	Number of nodes in the zx plane
LHBW	Jacobian matrix half-band width
LJA	Number of conjugate gradient solution vector elements
LJB	Number of conjugate gradient non-zero matrix elements
LJC	Number of conjugate gradient row pointer vector elements
LJD	Number of banded matrix rows
LJE	Number of banded matrix columns
LJF	Number of solution vector elements
LJG	Number of conjugate gradient matrix pointer rows

Table 5.2. Cont'd.

Parameter	Definition
LJH	Number of conjugate gradient matrix pointer columns
LJI	Number of banded solver pivot array elements
LJJ	Number of banded solver zero-diagonal array elements
LJK	Number of conjugate gradient matrix pointer rows
LJL	Number of conjugate gradient matrix pointer columns
LJM	Number of conjugate gradient non-zero matrix elements
LJN	Number of conjugate gradient row pointer vector elements
LNWN	Number of well nodes
LNWV	Number of well node variable indices
LOUPV	Number of output variable options
LP_TA	Number of CO2 property table pressure-dependent elements
LPH	Number of phases
LRCC	Number of solute rock/soil variable elements
LRCD	Number of dissolved rock/soil variable elements
LRCG	Number of air rock/soil variable elements
LRCI	Number of ice rock/soil variable elements
LRCL	Number of water rock/soil variable elements
LRCN	Number of oil rock/soil variable elements
LRCS	Number of salt/surfactant rock/soil variable elements
LRCT	Number of energy rock/soil variable elements
LRPGC	Number of gas relative permeability variable elements
LRPL	Number of relative permeability tensor elements
LRPLC	Number of aqueous relative permeability variable elements
LRPNC	Number of NAPL relative permeability variable elements
LSALC	Combined salt and alcohol mass equation switch
LSCHR	Number of saturation function variable elements
LSFV	Number of flux variable indices
LSU	Number of field variable indices
LSV	Number of field variable indices
LSX	Number of nodes in the x direction
LSXC	Number of solute flux variable elements in the x direction
LSXD	Number of dissolved oil flux variable elements in the x direction
LSXG	Number of air flux variable elements in the x direction
LSXL	Number of water flux variable elements in the x direction
LSXN	Number of oil flux variable elements in the x direction
LSXS	Number of salt/surfactant flux variable elements in the x direction
LSXT	Number of energy flux variable elements in the x direction
LSY	Number of nodes in the y direction
LSYC	Number of solute flux variable elements in the y direction
LSYD	Number of dissolved oil flux variable elements in the y direction
LSYG	Number of air flux variable elements in the y direction
LSYL	Number of water flux variable elements in the y direction
LSYN	Number of oil flux variable elements in the y direction

Table 5.2. Cont'd.

Parameter	Definition
LSYS	Number of salt/surfactant flux variable elements in the y direction
LSYT	Number of energy flux variable elements in the y direction
LSZ	Number of nodes in the z direction
LSZC	Number of solute flux variable elements in the z direction
LSZD	Number of dissolved oil flux variable elements in the z direction
LSZG	Number of air flux variable elements in the z direction
LSZL	Number of water flux variable elements in the z direction
LSZN	Number of oil flux variable elements in the z direction
LSZS	Number of salt/surfactant flux variable elements in the z direction
LSZT	Number of energy flux variable elements in the z direction
LSZW	Number of well flux variable elements in the z direction
LP_TA	Number of CO2 property table temperature-dependent elements
LUK	Number of solved coupled equations
LUKW	Number of solved coupled equations with coupled wells

5.4 Execution

Executing the simulator is a straight forward and only requires that the executable version of the code and an input file named *input* reside in the current directory. For restart simulations a restart file named *restart* must also reside in the current directory. Because restart files are created with an extension that corresponds with the generating time step, the user must rename the appropriate restart file to *restart*. For a UNIX operating system, execution is started by typing in the name of the executable file. Execution will be indicated by the printing of a STOMP title banner and program disclaimer to the standard input/output device (e.g., screen). Two types of error messages may be generated, during an STOMP execution. The first type is a system generated message that typically indicates a memory, FORTRAN, or other system error identified by the system. The second type of error messages refer to those generated by the STOMP code, which typically refer to input, parameter, or convergence failure type messages. STOMP generated messages are divided into three categories according to severity. The most severe are *ERROR* messages, which abort the program execution. Undersized parameters are typical of errors which yield *ERROR* messages, because execution of the simulator with undersized parameters may yield gross errors, or even worse subtle errors which may pass undetected in the results. Next on the severity level are the *WARNING* messages, which generally warrant notice by the user that a problem with the input file probably exists. The least severe are *NOTE* messages, which are used to record events like the absence of an optional input card.

When an execution is terminated with a STOMP generated *ERROR* message two types of information appear on the standard input/output device. The first piece of information is an error message with or without related data that indicates the nature of the error. Input formatting errors are trapped to a limited degree by the simulator. Unfortunately, the input error message that results may not always agree with the actual location of the identified input error. This is because the original error may have skipped detection. For example, a missing input item on an

input line would not be identified until STOMP identified an incorrect data type or the end of the line was reached. The user should expect to find input errors slightly before the location indicated by the generating error message. The second type of information, which is generated with an *ERROR* message, is a calling sequence. The calling sequence is a slash delineated list of routine names which were called prior to the identified error. The calling sequence for an input error on the *Aqueous Relative Permeability Card* for the *Water* operational mode would appear as

```
/STOMP1/RDINPT1/RDRPL/WRMSG
```

where the called routine names are delineated with a forward slash. For this example, the main routine *STOMP1* called the subroutine *RDINPT1* to read the input file for the *Water* operational mode, which called *RDRPL* the global routine to read the *Aqueous Relative Permeability Card*, which identified an error and called *WRMSG* to write an error message to the standard input/output device (e.g., screen).

6. Output Files

6.1 Introduction

The STOMP simulator can generate, depending on the requested output and saturation function type, two data files for restarting a simulation and three simulation result files. Every execution produces an output file named *output*. If the simulation concludes without a fatal error, then a *restart.n* and a *plot.n* file are also generated. Restart and plot file names include extensions (i.e., *.n*) which correspond to the time step for which the file was written. For example, a restart file named *restart.39* would have been written at the conclusion of the 39th time step. A surface file named *surface* is generated whenever a Surface Flux Card is included in the *input* file. All output files generated by the simulator (*output, restart.n, plot.n,* and *surface* files) are ASCII text files.

6.2 Output File

An output file named *output* is generated with every execution of the simulator. If a file named *output* already exists in the current directory then that file will be erased and overwritten. The user is responsible for renaming previous *output* files prior to executing the simulator. An *output* file begins with a welcome statement, disclaimer, and banner, which should appear similar to

```
Welcome to ...


                          STOMP
       Subsurface Transport Over Multiple Phases


This file was produced by STOMP, a numerical simulator
developed by the Pacific Northwest National Laboratory, with
support from the VOC-Arid Integrated Demonstration Project,
Office of Technology Development, U.S. Department of Energy.
Results from this version of STOMP should not be used for
license related applications.

For inquiries or assistance:  Call (509) 372-6070

                     ---  OUTPUT  ---
```

Following this banner is an input record, which documents the simulation input. An input record is included on the *output* file to document the simulation in the event that the *input*

file is destroyed or becomes separated from the *output* file. The input record is formatted differently that an *input* file, but contains all of the information listed on an *input* file. Optional cards which are not included in the *input* file are noted in the input record. If the simulator identifies an input error, an error message will appear in the input record at the point the error was noted. Input cards are read by the simulator in a specific order and will appear in the input record in that order. Because input cards can be organized randomly on the *input* file the card order on the input record may not match that on the *input* file. The reference node output record follows the input record.

The reference node output record is a table of simulation data and selected reference node variables, which are printed according the frequency requested on the Output Control Card with the input item *Reference Node Output File Frequency*. A table column header is printed to delineate the columns every 10 print records. Definitions of the header abbreviations for the reference node output are given in Table 6.1. Each print record will show simulation data and reference node variable data for each reference node requested. Simulation data comprises the time step, the reference node number, the simulation time, the simulation time step, and the number of Newton-Raphson iterations required to reach convergence. Reference node variable data comprise values of the variables specified on the Output Control Card through the Reference Node Variable inputs. Reference nodes are indexed by node number. The corresponding *x-, y-,* and *z*-direction indices are shown for each reference node on the line preceding the column headers. For example, a simulation with reference node 272 at *x-, y-,* and *z*-direction indices of 2, 3, and 6, respectively would have a reference node header line that appeared as

```
Reference Node(s) (  2, 3,  6:     272)
```

Column headers for reference node variables are delineated with an abbreviated title and associated units. Units are enclosed in brackets below the variable abbreviation. A portion of reference node output record for a horizontal heat pipe problem involving 50 nodes for time steps 70 through 73 appeared as

```
Reference Node(s) (  1,  1,  1:     1) ( 50,  1,  1:    50)
 Step  Node    Time      Timestep  Itr     T           SL          PL
             [day   ]   [day   ]       [c      ]                [pa     ]
   70     1  2.837E+01  1.793E+00  8  7.1893E+01  9.9893E-01  1.0133E+05
   70    50  2.837E+01  1.793E+00  8  1.3564E+02  2.0992E-02 -1.9896E+08
   71     1  3.061E+01  2.242E+00  9  7.1895E+01  9.9903E-01  1.0133E+05
   71    50  3.061E+01  2.242E+00  9  1.4039E+02  1.9265E-02 -2.2722E+08
   72     1  3.341E+01  2.802E+00  9  7.1878E+01  9.9933E-01  1.0133E+05
   72    50  3.341E+01  2.802E+00  9  1.4598E+02  1.7489E-02 -2.6050E+08
   73     1  3.692E+01  3.503E+00  8  7.1869E+01  9.9951E-01  1.0133E+05
   73    50  3.692E+01  3.503E+00  8  1.5225E+02  1.5759E-02 -2.9758E+08
```

where, the reference nodes are nodes 1 and 50, and the reference node variable data includes the temperature, reported in degrees Celsius, the aqueous saturation, and the aqueous pressure, reported in Pascal (absolute).

Convergence failures and subsequent time step reductions are also noted within the reference node output record. Refer to Chapter 5 for a discussion on convergence errors. Three

types of convergence error messages may appear within the output. The first type indicates that the update to a primary variable exceeded the maximum allowable change. An example of this type of convergence error for a horizontal heat pipe problem appeared as

```
      ---  Excessive Primary Variable Change  ---
Temperature =  3.80318E+02 Node =     50
Water Pressure = -1.33730E+09 Node =     50
Gas Pressure =  3.07342E+05 Node =     50
Time Step Reduced From  2.9383E+00 day to  5.8766E-01 day
```

where, an excessive change to at least one primary variable was noted at node 50 and the time step was reduced from 2.938 to 0.5877 days and repeated. The second type indicates that a converged solution was not obtained within the maximum number of Newton-Raphson iterations. An example of this type of convergence error for a three-phase volatile infiltration problem appeared as

```
      ---  Convergence Failure  ---
Water Equation Maximum Residual =  1.3313E-02 Node =    152
Air Equation Maximum Residual =  8.2280E-07 Node =     76
VOC Equation Maximum Residual =  1.0472E-02 Node =    154
Time Step Reduced From  7.0121E-03 hr to  1.4024E-03 hr
```

where, the maximum normalized residuals for the water and oil mass conservation equations were noted, which exceeded the convergence criterion. After this convergence failure the time step was reduced from 0.00701 to 0.00140 hours and repeated. The third type indicates that the linear system solver has failed to reach a solution. An example of this type of convergence error for a three-phase volatile infiltration problem appeared as

```
      ---  Singular Matrix  ---
Jacobian Matrix Index =    339
Node Number           =    113
Time Step Reduced From  1.4849E-02 hr to  2.9697E-03 hr
```

where, a zero or nearly zero diagonal was noted for the oil equation at node 113. After this convergence failure the time step was reduced from 0.0148 to 0.00297 hours and repeated. All three types of error messages include additional information that pertains to the specific convergence problem. Convergence errors result in a reduction in the time step with a repeated attempt to solve the system of governing equations. Four successive convergence failures without an intermediate successfully converged time steps results in termination of the execution. Output files are concluded with the following closing message

```
      ---  End of STOMP Simulation  ---
```

6.3 Plot File

A plot file is generated by default at the conclusion of every execution and otherwise when requested on the Output Control Card with the Plot File Output Times input items. If a file named *plot.n* (where *n* refers to the time step extension) already exists in the current directory then the file will be erased and overwritten. The user is responsible for renaming previous *plot.n*

files prior to executing the simulator. A plot file begins with a welcome statement, disclaimer, and banner, which should appear similar to

```
 Welcome to ...


                            STOMP
        Subsurface Transport Over Multiple Phases


 This file was produced by STOMP, a numerical simulator
 developed by the Pacific Northwest National Laboratory, with
 support from the VOC-Arid Integrated Demonstration Project,
 Office of Technology Development, U.S. Department of Energy.
 Results from this version of STOMP should not be used for
 license related applications.

 For inquiries or assistance:  Call (509) 372-6070

                       ---  PLOT  ---
```

Following this header are data on the current time step, simulation time, and number of nodes in three coordinate directions. The remaining portion of a *plot.n* file comprises arrays of geometric and variable data. These data are arranged in groups, each with a title line that indicates the type of data. Data groups are arranged in rows with 10 data items per row. Each group of data contains a field or surface variable listed sequentially for every node or surface in the computational domain. The first four groups of data appear in every *plot.n* file and contain the x- or radial-direction node positions, y- or azimuthal-direction node positions, z-direction node positions, and node volumes for every node in the computational domain. The remaining groups of data correspond to the requested *Plot File Variables.* Field variable data groups list the value of the field variable at the node centroid. Surface variables and/or fluxes are written to the plot file twice. The first flux variable data group lists the value of the flux variable at the node centroid, whereas the second group lists the value at the node surfaces. Node centroid values for flux variables are simple arithmetic averages of the two node surface values.

6.4 Surface File

A surface file named *surface* is generated during an execution if surfaces were defined on the *Surface Flux Card*. If a file named *surface* already exists in the current directory then that file will be erased and overwritten. The user is responsible for renaming previous *surface* files prior to executing the simulator. A *surface* file begins with a welcome statement, disclaimer, and banner, which should appear similar to

```
 Welcome to ...
```

```
                          STOMP
        Subsurface Transport Over Multiple Phases


 This file was produced by STOMP, a numerical simulator
 developed by the Pacific Northwest National Laboratory, with
 support from the VOC-Arid Integrated Demonstration Project,
 Office of Technology Development, U.S. Department of Energy.
 Results from this version of STOMP should not be used for
 license related applications.

 For inquiries or assistance:  Call (509) 372-6070

                       ---  SURFACE  ---
```

Following the header are columns of surface flux rate and integral data. Surface flux information is written to the *surface* file every time step, at the conclusion of the time step. The first column of data in the *surface* file is the time step in units, which are specified with the *Output Time Units* variable on the Output Control Card. Each surface flux output is written in two columns, the first column being the surface flux rate at the current time step and the second being the integral of the surface flux rate from the beginning of the current simulation. Surface flux integral data are not carried over on restart simulations. Each column in the *surface* file is titled with an abbreviated header and associated units shown in brackets below the column title. Definitions of the header abbreviations for the *surface* file are given in Table 6.2. Each header title is followed with an integer enclosed in parentheses. The integer refers to the surface flux number, as defined on the Surface Flux Card (i.e., the surface flux definition order).

6.5 Restart File

A restart file is generated by default at the conclusion of every execution and otherwise when requested on the Output Control Card with the Plot File Output Times input items. If a file named *restart.n* (where *n* refers to the time step extension) already exists in the current directory then the file will be erased and overwritten. The user is responsible for renaming previous *restart.n* files prior to executing the simulator. The user has the option to specify a "no restart" or "final restart" as a plot variable. A restart file begins with a welcome statement, disclaimer, and banner, which should appear similar to

```
 Welcome to ...


                          STOMP
        Subsurface Transport Over Multiple Phases


 This file was produced by STOMP, a numerical simulator
```

```
developed by the Pacific Northwest National Laboratory, with
support from the VOC-Arid Integrated Demonstration Project,
Office of Technology Development, U.S. Department of Energy.
Results from this version of STOMP should not be used for
license related applications.

For inquiries or assistance:  Call (509) 372-6070

                    ---  RESTART  ---
```

All data values in restart files are in SI units with pressures expressed in gauge. The first data line of a restart file contains the following timing and simulation information: time, time step, maximum time step, time step acceleration factor, convergence criterion, maximum number of Newton-Raphson iterations, number of time steps, and number of nodes. The remaining lines are node data lines, which contain the following field variable data for the current time step for every node (one line per node): temperature, aqueous pressure, gas pressure, NAPL pressure, oil vapor pressure, water vapor pressure, aqueous saturation, gas saturation, and NAPL saturation. If the simulation includes solute transport then the node data lines will also contain solute volumetric concentrations for each solute. All node data lines are terminated with an integer value for the phase condition.

Table 6.1. Reference Node Output Record Abbreviations

Abbreviation	Definition
AST	Apparent total liquid saturation
ASL	Apparent aqueous saturation
ASLM	Minimum effective aqueous saturation
ARH	Atmospheric relative humidity
ARN	Atmospheric solar radiation
AT	Atmospheric temperature
AWS	Atmospheric wind speed
BGL	Gas-aqueous surface tension scaling factor
BGN	Gas-NAPL surface tension scaling factor
BNL	NAPL-aqueous surface tension scaling factor
Cn[(a)]	Solute volumetric concentration
CGn[(a)]	Solute gas volumetric concentration
CGA	Air mass concentration in the gas phase
CGO	Oil mass concentration in the gas phase
CGW	Water mass concentration in the gas phase
CLn[(a)]	Solute aqueous volumetric concentration
CLA	Air mass concentration in the aqueous phase
CLO	Oil mass concentration in the aqueous phase
CLS	Salt or surfactant mass concentration in the aqueous phase
CLW	Water mass concentration in the aqueous phase
CNn[(a)]	Solute NAPL volumetric concentration
CNO	Oil mass concentration in the NAPL phase
CNW	Water mass concentration in the NAPL phase
CRNG	Maximum local gas phase courant number
CRNL	Maximum local aqueous phase courant number
CRNN	Maximum local NAPL phase courant number
CS	Salt volumetric concentration
CSL	Salt aqueous volumetric concentration
DAPS	Dissolved air percent saturation
DSGF	Dual porosity gas saturation (fracture)
DSGM	Dual porosity gas saturation (matrix)
DSLF	Dual porosity aqueous saturation (fracture)
DSLM	Dual porosity aqueous saturation (matrix)
EA_S	Actual evaporation rate
EP_S	Potential evaporation rate

[(a)] n refers to a solute number

Abbreviation	Definition
ESGT	Effective trapped gas saturation
ESML	Minimum effective aqueous saturation
ESNT	Effective trapped NAPL saturation
GPG	Gas pressure (gauge)
GPL	Aqueous pressure (gauge)
GPN	NAPL pressure (gauge)
HHG	Gas hydraulic head (water equivalent)
HHL	Aqueous hydraulic head (water equivalent)
HHN	NAPL hydraulic head (water equivalent)
HKL	Aqueous solute coefficient
HKNF	Free-NAPL solute coefficient
HKNT	Trapped-NAPL solute coefficient
IAM	Integrated water mass
IAMG	Integrated air in the gas phase
IAML	Integrated air in the aqueous phase
IAMN	Integrated air in the NAPL phase
IATG	Integrated trapped air in the gas phase
ICM	Variable grid spacing index
IMA	Integrated air or CO_2 mass
IMGA	Integrated air or CO_2 mass in gas
IMGO	Integrated oil mass in gas
IMGW	Integrated water mass in gas
IMO	Integrated oil mass
IMOG	Integrated oil in the gas phase
IMOL	Integrated oil in the aqueous phase
IMON	Integrated oil in the NAPL phase
IMW	Integrated water mass
IPLW	Integrated well aqueous pumping
IPNW	Integrated mineral CO_2 well NAPL pumping
IWM	Integrated water mass
IWMG	Integrated water in the gas phase
IWML	Integrated water in the aqueous phase
IWMN	Integrated water in the NAPL phase
MCL	Aqueous moisture content
MCN	NAPL moisture content
MCO2	Mineralized CO_2
MCT	Total liquid moisture content
MPH	Matric Potential Head
OEC	Osmotic efficiency coefficient

PATH	Scanning path (-1=drying, 1=wetting)
Abbreviation	**Definition**
PG	Gas pressure (absolute)
PHCN	Phase condition index
PL	Aqueous pressure (absolute)
PLWB	Aqueous well depth
PN	NAPL pressure (absolute)
PORD	Diffusive porosity
POSM	Osmotic pressure
PV_S	Surface vapor pressure
PVA	Air or CO_2 partial pressure
PVW	Water vapor partial pressure
PW	Monitoring well pressure
QLW	Aqueous well flow rate
QLWI	Integrated aqueous well flow volume
QNW	NAPL well flow rate
QNWI	Integrated NAPL well flow volume
QTW	Total well flow rate
QTWI	Integrated total liquid well flow volume
RHOG	Gas density
RHOL	Aqueous density
RHON	NAPL density
RPG	Gas relative permeability
RPL	Aqueous relative permeability
RPN	NAPL relative permeability
RSZN	Rock/Solute zone number
SG	Gas saturation
SGT	Trapped gas saturation
SGTL	Aqueous trapped gas saturation
SGTN	NAPL trapped gas saturation
SL	Aqueous saturation
SLW	Well water saturation
SN	NAPL saturation
SNF	Free NAPL saturation
SNM	Mobile NAPL saturation
SNR	Residual NAPL saturation
SNT	Trapped NAPL saturation
SPCM	Space monitor
ST	Total-liquid saturation

SRn[(a)]	Solute source integral
Abbreviation	**Definition**
SRCA	Air or CO_2 mass source rate
SRCO	Oil mass source rate
SRCT	Thermal energy source rate
SRCS	Salt or surfactant mass source rate
SRCW	Water mass source rate
SRIA	Air or CO_2 mass source integral
SRIO	Oil mass source integral
SRIS	Salt or surfactant mass source integral
SRIT	Thermal energy mass source integral
SRIW	Water mass source integral
ST	Total liquid saturation
T	Temperature
T_S	Surface temperature
TA_S	Actual transpiration rate
THKX	Equivalent thermal conductivity (x-direction)
THKY	Equivalent thermal conductivity (y-direction)
THKZ	Equivalent thermal conductivity (z-direction)
TMA	Total air mass
TMO	Total oil mass
TMW	Total water mass
TMS	Total salt mass
TP_S	Potential transpiration rate
TPNL	NAPL-aqueous trapping number
UCn[(a)]	Solute flux (x-direction)
UG	Gas Darcy velocity (x-direction)
UGNC	Gas Darcy velocity (x-direction, node-centered)
UG_W	Gas well cross-screen flux
UL	Aqueous Darcy velocity (x-direction)
ULNC	Aqueous Darcy velocity (x-direction, node-centered)
UL_W	Water well cross-screen flux
UN	NAPL Darcy velocity (x-direction)
UNNC	NAPL Darcy velocity (x-direction, node-centered)
US	Salt flux (x-direction)
USNC	Salt flux (x-direction, node-centered)
UQ	Energy flux (x-direction)
UQNC	Energy flux (x-direction, node-centered)

[(a)] n refers to a solute number

VCn[(a)]	Solute flux (y-direction)
VG	Gas Darcy velocity (y-direction)
VGNC	Gas Darcy velocity (y-direction, node-centered)
Abbreviation	**Definition**
VISG	Gas phase viscosity
VISL	Aqueous phase viscosity
VISN	NAPL viscosity
VL	Aqueous Darcy velocity (y-direction)
VLNC	Aqueous Darcy velocity (y-direction, node-centered)
VN	NAPL Darcy velocity (y-direction)
VNNC	NAPL Darcy velocity (y-direction, node-centered)
VS	Salt flux (y-direction)
VSNC	Salt flux (y-direction, node-centered)
VQ	Energy flux (y-direction)
VQNC	Energy flux (y-direction, node-centered)
WCn[(a)]	Solute flux (z-direction)
WDL	Aqueous well depth
WDN	NAPL well depth
WDT	Total liquid well depth
WG	Gas Darcy velocity (z-direction)
WGNC	Gas Darcy velocity (z-direction, node-centered)
WG_W	Gas vertical well flux
WL	Aqueous Darcy velocity (z-direction)
WLNC	Aqueous Darcy velocity (z-direction, node-centered)
WL_W	Aqueous vertical well flux
WN	NAPL Darcy velocity (z-direction)
WNNC	NAPL Darcy velocity (z-direction, node-centered)
WS	Salt flux (z-direction)
WSNC	Salt flux (z-direction, node-centered)
WQ	Energy flux (z-direction)
WQNC	Energy flux (z-direction, node-centered)
XGA	Air or CO_2 mass fraction in the gas phase
XGO	Oil mass fraction in the gas phase
XGW	Water mass fraction in the gas phase
XGWW	Monitoring well water mass fraction in the gas phase
XLA	Air or CO_2 mass fraction in the aqueous phase
XLAW	Monitoring well dissolved air or CO_2 mass fraction
XLO	Oil mass fraction in the aqueous phase

[(a)] n refers to a solute number

XLS	Salt or surfactant mass fraction in the aqueous phase
XLW	Water mass fraction in the aqueous phase
XMGA	Air or CO_2 mole fraction in the gas phase
XMGO	Oil mole fraction in the gas phase
Abbreviation	**Definition**
XMGW	Water mole fraction in the gas phase
XMLA	Aqueous CO_2 mole fraction
XMLS	Aqueous salt mole fraction
XMNO	Oil mole fraction in the NAPL phase
XMNW	Water mole fraction in the gas phase
XNW	Water mass fraction in the NAPL phase
XNO	Oil mass fraction in the NAPL phase
YGn[(a)]	Solute fraction in the gas phase
YLn[(a)]	Solute fraction in the aqueous phase
YCNn[(a)]	Solute fraction in NAPL

Table 6.2. Surface Flux Column Title Abbreviations

Abbreviation	Definition
UCMR	Solute flux (x-direction)
UCMI	Solute flux integral (x-direction)
UCO2R	CO_2 mass flux rate (x-direction)
UCO2I	CO_2 mass flux rate integral (x-direction)
UDOR	Dissolved oil flux (x-direction)
UDOI	Dissolved oil flux integral (x-direction)
UGAAR	Gas advective air mass flux (x-direction)
UGAAI	Gas advective air mass flux integral (x-direction)
UGAOR	Gas advective oil mass flux (x-direction)
UGAOI	Gas advective oil mass flux integral (x-direction)
UGAQR	Gas advective heat flux (x-direction)
UGAQI	Gas advective heat flux integral (x-direction)
UGAWR	Gas advective water mass flux (x-direction)
UGAWI	Gas advective water mass flux integral (x-direction)
UGCO2R	Gas CO_2 mass flux rate (x-direction)
UGCO2I	Gas CO_2 mass flux rate integral (x-direction)
UGDAR	Gas diffusive air mass flux (x-direction)
UGDAI	Gas diffusive air mass flux integral (x-direction)
UGDOR	Gas diffusive oil mass flux (x-direction)
UGDOI	Gas diffusive oil mass flux integral (x-direction)
UGDQR	Gas diffusive heat flux (x-direction)
UGDQI	Gas diffusive heat flux integral (x-direction)
UGDWR	Gas diffusive water mass flux (x-direction)
UGDWI	Gas diffusive water mass flux integral (x-direction)
UGMR	Gas mass flux rate (x-direction)
UGMI	Gas mass flux rate integral (x-direction)
UGTOR	Gas total oil mass flux (x-direction)
UGTOI	Gas total oil mass flux integral (x-direction)
UGVR	Gas volumetric flux rate (x-direction)
UGVI	Gas volumetric flux rate integral (x-direction)
ULCO2R	Aqueous CO_2 mass flux rate (x-direction)
ULCO2I	Aqueous CO_2 mass flux rate integral (x-direction)
ULMR	Aqueous mass flux rate (x-direction)
ULMI	Aqueous mass flux rate integral (x-direction)
ULVR	Aqueous volumetric flux rate (x-direction)
ULVI	Aqueous volumetric flux rate integral (x-direction)

UNMR	NAPL mass flux rate (x-direction)
Abbreviation	**Definition**
UNMI	NAPL mass flux integral (x-direction)
UNVR	NAPL volumetric flux rate (x-direction)
UNVI	NAPL volumetric flux integral (x-direction)
UQVR	Energy flux rate (x-direction)
UQVI	Energy flux integral (x-direction)
USMR	Salt mass flux rate (x-direction)
USMI	Salt mass flux integral (x-direction)
UWMR	Water mass flux (x-direction)
UWMI	Water mass flux integral (x-direction)
VCMR	Solute flux (y-direction)
VCMI	Solute flux integral (y-direction)
VCO2R	CO_2 mass flux rate (y-direction)
VCO2I	CO_2 mass flux rate integral (y-direction)
VDOR	Dissolved oil flux (y-direction)
VDOI	Dissolved oil flux integral (y-direction)
VGAAR	Gas advective air mass flux (y-direction)
VGAAI	Gas advective air mass flux integral (y-direction)
VGAOR	Gas advective oil mass flux (y-direction)
VGAOI	Gas advective oil mass flux integral (y-direction)
VGAQR	Gas advective heat flux (y-direction)
VGAQI	Gas advective heat flux integral (y-direction)
VGAWR	Gas advective water mass flux (y-direction)
VGAWI	Gas advective water mass flux integral (y-direction)
VGCO2R	Gas CO_2 mass flux rate (y-direction)
VGCO2I	Gas CO_2 mass flux rate integral (y-direction)
VGDAR	Gas diffusive air mass flux (y-direction)
VGDAI	Gas diffusive air mass flux integral (y-direction)
VGDOR	Gas diffusive oil mass flux (y-direction)
VGDOI	Gas diffusive oil mass flux integral (y-direction)
VGDQR	Gas diffusive heat flux (y-direction)
VGDQI	Gas diffusive heat flux integral (y-direction)
VGDWR	Gas diffusive water mass flux (y-direction)
VGDWI	Gas diffusive water mass flux integral (y-direction)
VGMR	Gas mass flux rate (y-direction)
VGMI	Gas mass flux rate integral (y-direction)
VGTOR	Gas total oil mass flux (y-direction)
VGTOI	Gas total oil mass flux integral (y-direction)
VGVR	Gas volumetric flux rate (y-direction)

VGVI	Gas volumetric flux rate integral (y-direction)
VLCO2R	Aqueous CO_2 mass flux rate (y-direction)
Abbreviation	**Definition**
VLCO2I	Aqueous CO_2 mass flux rate integral (y-direction)
VLMR	Aqueous mass flux rate (y-direction)
VLMI	Aqueous mass flux rate integral (y-direction)
VLVR	Aqueous volumetric flux rate (y-direction)
VLVI	Aqueous volumetric flux rate integral (y-direction)
VNMR	NAPL mass flux rate (y-direction)
VNMI	NAPL mass flux integral (y-direction)
VNVR	NAPL volumetric flux rate (y-direction)
VNVI	NAPL volumetric flux integral (y-direction)
VQVR	Energy flux rate (y-direction)
VQVI	Energy flux integral (y-direction)
VSMR	Salt mass flux rate y-direction)
VSMI	Salt mass flux integral (y-direction)
VWMR	Water mass flux (y-direction)
VWMI	Water mass flux integral (y-direction)
WCMR	Solute flux (z-direction)
WCMI	Solute flux integral (z-direction)
WCO2R	CO_2 mass flux rate (z-direction)
WCO2I	CO_2 mass flux rate integral (z-direction)
WDOR	Dissolved oil flux (z-direction)
WDOI	Dissolved oil flux integral (z-direction)
WGAAR	Gas advective air mass flux (z-direction)
WGAAI	Gas advective air mass flux integral (z-direction)
WGAOR	Gas advective oil mass flux (z-direction)
WGAOI	Gas advective oil mass flux integral (z-direction)
WGAQR	Gas advective heat flux (z-direction)
WGAQI	Gas advective heat flux integral (z-direction)
WGAWR	Gas advective water mass flux (z-direction)
WGAWI	Gas advective water mass flux integral (z-direction)
WGCO2R	Gas CO_2 mass flux rate (z-direction)
WGCO2I	Gas CO_2 mass flux rate integral (z-direction)
WGDAR	Gas diffusive air mass flux (z-direction)
WGDAI	Gas diffusive air mass flux integral (z-direction)
WGDOR	Gas diffusive oil mass flux (z-direction)
WGDOI	Gas diffusive oil mass flux integral (z-direction)
WGDQR	Gas diffusive heat flux (z-direction)
WGDQI	Gas diffusive heat flux integral (z-direction)

WGDWR	Gas diffusive water mass flux (z-direction)
WGDWI	Gas diffusive water mass flux integral (z-direction)
WGMR	Gas mass flux rate (z-direction)
Abbreviation	**Definition**
WGMI	Gas mass flux rate integral (z-direction)
WGTOR	Gas total oil mass flux (z-direction)
WGTOI	Gas total oil mass flux integral (z-direction)
WGVR	Gas volumetric flux rate (z-direction)
WGVI	Gas volumetric flux rate integral (z-direction)
WLCO2R	Aqueous CO_2 mass flux rate (z-direction)
WLCO2I	Aqueous CO_2 mass flux rate integral (z-direction)
WLMR	Aqueous mass flux rate (z-direction)
WLMI	Aqueous mass flux rate integral (z-direction)
WLMR	Aqueous mass flux rate (z-direction)
WLMI	Aqueous mass flux rate integral (z-direction)
WLVR	Aqueous volumetric flux rate (z-direction)
WLVI	Aqueous volumetric flux rate integral (z-direction)
WNMR	NAPL mass flux rate (z-direction)
WNMI	NAPL mass flux integral (z-direction)
WNVR	NAPL volumetric flux rate (z-direction)
WNVI	NAPL volumetric flux integral (z-direction)
WQVR	Energy flux rate (z-direction)
WQVI	Energy flux integral (z-direction)
WSMR	Salt mass flux rate (z-direction)
WSMI	Salt mass flux integral (z-direction)
WWMR	Water mass flux (z-direction)
WWMI	Water mass flux integral (z-direction)

7. References

ANSI. 1978. *American National Standard Programming Language FORTRAN,* X3.9-1978, American National Standards Institute, Broadway, New York.

Poeter, E.P. and M.C. Hill. 1998. *Documentation of UCODE, a computer code for universal inverse modeling.* U.S. Geological Survey Water-Resources Investigations Report 98-4080, 116p.

Reid R.C., J.M. Prausnitz, and B.E. Poling. 1987. *The Properties of Gases and Liquids*, McGraw-Hill, New York.

Talbott S. 1988. *Managing Projects with make*, O'Reilly and Associates, Inc., Newton, Massachusetts.

White M.D., and M. Oostrom. 2000. *STOMP Subsurface Transport Over Multiple Phases Theory Guide Version 2.0,* PNNL-12030, Pacific Northwest National Laboratory, Richland, Washington.

8. STOMP Related Publications

This section includes a partial list of peer-reviewed and conference proceedings papers in which the STOMP simulator has been used for numerical flow and transport applications. Published reports and conference abstracts have not been listed.

Ataie-Ashtiani, B., S.M. Hassanizadeh, M. Oostrom, M.A. Celia, and M.D. White. 2001. Numerical simulation and homogenization of two-phase flow in heterogeneous porous media. J. of Contaminant Hydrology 49: 87-110.

Ataie-Ashtiani, B., S.M. Hassanizadeh, M.A. Celia, M. Oostrom, and M.D. White. 2000. Effective two-phase flow parameters for heterogeneous porous media. In: Proceedings of Computational Methods in Water Resources XIII, ed. L.R. Bentley et al. pp. 755-762, Kluwer Ac. Publ. London.

Ataie-Ashtiani, B., S.M. Hassanizadeh, M. Oostrom and M.D. White. 2000. Numerical simulation and homogenization of two-phase flow in heterogeneous porous media. In: Groundwater Updates, Proceedings of the IAHR International Symposium on Groundwater, K. Sato and Y. Iwasa (eds.), pp. 333-338, Springer-Verlag, Tokyo.

Gee, G.W., A.L. Ward, and Z.F. Zhang, G.S. Campbell, and J. Mathison. 2002. The influence of hydraulic nonequilibrium on pressure plate data. Vadose Zone Journal, 1:172-178.

Hofstee, C. and M. Oostrom. 2002. Formation of residual NAPL in three-phase systems. Experiments and numerial simulations. Proceedings of Computational Methods in Water Resources XIII, ed. S.M. Hassanizadeh et al., pp. 193-202.

Hofstee, C., M. Oostrom, J.H. Dane, and R.W. Walker. 1998. Infiltration and redistribution of perchloroethylene in partially saturated, stratified porous media. J. of Contaminant Hydrol. 34: 293-313.

Lenhard, R.J., M. Oostrom, C.S. Simmons, and M.D. White. 1995. Investigation of density-dependent gas advection of trichloroethylene: Experiment and a model validation exercise. J. of Contaminant Hydrol. 19:47-67.

Lenhard, R.J., M. Oostrom, and M.D. White. 1995. Modeling fluid flow and transport in variably saturated porous media with the STOMP simulator. 2. Verification and validation exercices. Adv. in Water Resour. 18:365-374.

Oostrom, M., C. Hofstee, and J.H. Dane. 1997. Light nonaqueous-phase liquid movement in a variably saturated sand. Soil Science Society of America J. 61: 1547-1554.

Oostrom, M., C. Hofstee, J.H. Dane, and M.D. White. 1996. Simulation of a quantitative two-dimensional multi-fluid flow experiment. In: Proceedings of the XI International Conference on Computational Methods in Water Resources, A.A. Aldema et al. (eds). pp. 449-556. Kluwer Ac. Publ. London.

Oostrom, M., C. Hofstee, R.J. Lenhard, and T.W. Wietsma. 2003. Flow behavior and residual saturation formation of injected carbon tetrachloride in unsaturated heterogeneous porous media. J. of Contaminant Hydrology. 64: 93-112.

Oostrom, M., C. Hofstee, R.C. Walker, and J.H. Dane. 1999. Movement and remediation of trichloroethylene in a saturated, heterogeneous porous medium. 1. Spill behavior and initial dissolution. J. of Contaminant Hydrol. 37: 159-178.

Oostrom, M., C. Hofstee, R.C. Walker, and J.H. Dane. 1999. Movement and remediation of trichloroethylene in a saturated, heterogeneous porous medium. 2. Pump-and-treat and surfactant flushing. J. of Contaminant Hydrol. 37: 179-197.

Oostrom, M. and R.J. Lenhard. 1996. Radon transport into dwellings: Considering groundwater as a potential source. Geoph. Res. Lett. 23:1577-1580.

Oostrom, M. and R.J. Lenhard. 1998. Comparison of relative permeability-saturation-pressure parametric models for infiltration and redistribution of a light nonaqueous phase liquid in sandy porous media. Adv. in Water Resour. 21:145-157.

Oostrom, M. and R.J. Lenhard. 2003. Carbon tetrachloride flow behavior in unsaturated Hanford caliche material: An investigation of residual NAPL. Vadose Zone J. 2: 25-33.

Oostrom, M., M.D. White, and M.L. Brusseau. 2001. Theoretical estimation of interfacial areas of entrapped and free nonwetting fluids in porous media. Advances in Water Resources 24: 887-898.

Oostrom, M., R.J. Lenhard, and M.D. White. 1995. Infiltration of dense and light nonaqueous phase liquids in partly saturated sand columns. In: Proceedings of the 15th American Geophysical Union Hydrology Days, H. Morel-Seytoux (ed) pp. 215-226, Hydrology Days Publications, Atherton, CA.

Oostrom, M., R.J. Lenhard, and M.D. White. 1996. Numerical evaluation of groundwater as a supply for radon in dwellings. In: Proceedings of the XI International Conference on Computational Methods in Water Resources, A.A. Aldema et al. (eds). pp. 193-200. Kluwer Ac. Publ. London.

Rockhold, M. L., C. J. Murray, and M. J. Fayer. 1999. Conditional simulation and upscaling of soil hydraulic properties. In Proceedings of the International Workshop on Characterization and Measurement of the Hydraulic Properties of Unsaturated Porous Media, M. Th. Van Genuchten, F. J. Leij, and L. Wu (eds.), University of California, Riverside, pp.1391-1401.

Rockhold, M. L., R.E. Rossi, and R. G. Hills. 1996. Application of similar media scaling and conditional simulation for modeling water flow and tritium transport at the Las Cruces Trench Site. Water Resour. Res. 32(3):595-609.

Rockhold, M. L. 1999. Parameterizing flow and transport models for field-scale applications in heterogeneous, unsaturated soils. In Assessment on Non-Point Source Pollution in the Vadose Zone, D. L. Corwin, K. Loague, and T. R. Ellsworth (eds.), Geophysical Monograph 108, American Geophysical Union, Washington, D.C., pp.243-260.

Schroth, M.H., J.D. Istok, J.S. Selker, M. Oostrom, and M.D. White. 1998. Multifluid flow in bedded porous media: Laboratory experiments and numerical simulations. Adv. in Water Resour. 22:169-183.

White, M. D. and R. J. Lenhard. 1993. Numerical analysis of a three-phase system with a fluctuating water table. In: Proceedings of Thirteenth Annual American Geophysical Union Hydrology Days, Fort Collins, Colorado. Hydrology Days Publication, Atherton, California.

White, M. D., and B. P. McGrail. 2002. Numerical investigations of multifluid hydrodynamics during injections of Supercritical CO_2 into Porous Media. In: Proceedings of GHGT-6, Sixth International Conference on Greenhouse Gas Control Technologies, Kyoto, Japan. Elsevier Science, Amsterdam, The Netherlands.

White, M.D., and M. Oostrom. 1997. Mathematical modeling of surfactant-enhanced mobilization and kinetic dissolution of DNAPLs in soil columns. In: Proceedings of the 17th American Geophysical Union Hydrology Days, H. Morel-Seytoux (ed), pp. 349-363. Hydrology Days Publications, Atherton, CA.

White, M.D. and M. Oostrom. 1998. Modeling surfactant-enhanced nonaqueous phase liquid remediation of porous media. Soil Science 163:931-940.

White, M.D. and M.Oostrom. 2001. NAPL migration in response to hydraulic controls at the Brooklawn Site near Baton Rouge. In: Proceedings of the 21th American Geophysical Union Hydrology Days, H.J. Morel-Seytoux, ed., pp. 241-254. Colorado State Univ., Fort Collins, CO.

White, M.D., M. Oostrom, and R.J. Lenhard. 1995. Modeling fluid flow and transport in variably saturated porous media with the STOMP simulator. 1. Nonvolatile three-phase model description. Adv. in Water Resour. 18:353-364.

White, M.D., M.D. Williams, and M.Oostrom. 1999. Simulating air entrapment, mobilization, and dissolution for fluctuating water table conditions. In: Proceedings of the 19th American Geophysical Union Hydrology Days, H.J. Morel-Seytoux, ed., pp. 467-482. Hydrology Days Pub., Atherton, CA.

Williams, M.D., and M. Oostrom. 2000. Re-oxygenation of anoxic water in a laboratory fluctuating water table system. J. of Hydrology 230: 70-85.

Zhang, Z.F., A.L. Ward, and G.W. Gee. 2003. Inversely estimating soil hydraulic parameters through field drainage experiment. Vadose Zone Journal, 2:201-211.

Appendix

A. Input Format Notation Guide

Notation	Description
{ Option }	Character string options are indicated by enclosing braces. Options are chosen by entering word(s) within the braces, exactly as shown. Only one option should be chosen for each data entry.
[Optional]	Enclosing brackets indicate optional characters or words. These characters can be entered to include the input file to improve its readability or to specify optional features.
{{ Contains }}	Indicates the option contains a particular word. For example "Fractured Tuff" contains the word "Fractured" thus indicating a dual-porosity type rock/soil.
< *Data Types* >	Indicates repeated formatting.
Char[a]	Character string data type, referenced by superscript "a".
Integer[a]	Integer data type (no character data or decimal points) reference by superscript "a".
Real[a]	Real data type (decimal points and exponential notation are acceptable), reference by superscript "a".
#	A pound symbol in the first column indicates a comment line and will be ignored during execution. Comment lines may be placed inside or outside card structures. All lines outside of the card structures are ignored during execution.
~ Card Name	A tilde symbol in the first column indicates the start of a new card.

,	Data entries are comma delimited. Commas shown in the line format structures must be entered as shown, including a closing comma at the end of each line. Characters following the last comma of a data line are ignored during execution.
Units[a] (m)	Indicates the SI unit for the input data item referenced by superscript "a".
|	Indicates a choice between more than one options
Format:	Indicates line formatting instructions and the beginning of a new input line. Each format statement requires a new input line.
Endcard:	Indicates end of a card.
For: Integer Instructions **Endfor:** Integer	Indicates instruction looping.
If: Name: Card = { Opt_1 } Instructions1 **Elseif:** Name: Card = { Opt_2 } Instructions2 **Elseif:** Instructions3 **Endif:**	Indicates decision logic.
IfDef: Opt_1 Instructions1 **ElseifDef:** Opt_2 Instructions2 **ElseDef:** Instructions3 **EndifDef:**	Indicates C preprocessor options and logic.
Note:	Indicates formatting information.

Appendix
B. STOMP Input Control Card Formats

B.1 Atmospheric Conditions Card

Card Title[a] { ~Atmospheric [Conditions Card] }
Format: *Char*[a]

If: Operational Mode Option Card = { **Water-Air-Energy** }

Atmospheric Start Time: Month[a], Atmospheric Start Time: Day[b],
Atmospheric Start Time: Year[c], Atmospheric Start Time: Time (military format)[d],
Wind Speed Measurement Height[e], Units[f] (m),
Air Temperature/Relative Humidity Measurement Height[g], Units[h] (m),
Local Longitude[i], Units[j] (deg),
Local Meridian[k], Units[l] (deg),
Format: *Char*[a], *Integer*[b], *Integer*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Char*[j], *Real*[k], *Char*[l],

If: Atmospheric Conditions Read From External File

External File with Atmospheric Conditions[a],
Format: *Char*[a],

Elseif: Atmospheric Conditions Read From Input File

Number of Atmospheric Condition Times[a],
Format: *Integer*[a],

For: Number of Atmospheric Condition Times
Atmospheric Condition Time[a], Units[b] (s),
Atmospheric Condition Temperature[c], Units[d] (K),
Atmospheric Condition Pressure[e], Units[f] (kg/(m s^2)),
Atmospheric Condition Water-vapor Relative Humidity[g],
Atmospheric Condition Net Solar Radiation[h], Units[i] (kg/m^3),
Atmospheric Condition Wind Speed[j], Units[k] (m/s),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Real*[h], *Char*[i], *Real*[j], *Char*[k],
Endfor:

Endif:

Endif:

Endcard: Atmospheric Conditions Card

B.1.1 Atmospheric Conditions Card Examples

Extracted from a STOMP3 (Water-Air-Energy) input file:
~Atmospheric Conditions Card
June,29,2000,00:00:00,2.0,m,2.0,m,119.627,deg,120.0,deg,
file,hms2july,

B.2 Aqueous Relative Permeability Card

Card Title[a] { ~Aqueous Rel [ative Permeability Card] }
Format: *Char*[a]

For: Number of Rock/Soil Types or Scaling Groups

If: Rock/Soil or Scaling Group Name = { IJK | JKI | KIJ } Indexing

Note: A parameter value input can be replaced with an external file using the following formatting for ASCII files:

file: *filename*

or the following formattings for binary files:

binary file: *filename*

where; the external file will contain unique parameter values for each node (active or inactive) arranged according to the indexing scheme (i.e., IJK, JKI, or KIJ). Applicable units will be applied to all parameter values in the external file. An example input card is included in section B.2.1

Endif:

If: Anisotropy Option is specified in combination with the Mualem permeability function option, the following two real parameters have to be added to the end of the input line (see example in B2.1):
Horizontal Pore-Scale Parameter (0.5), Horizontal Pore-Scale Parameter (0.5),
Endif:

If: Polmann Anisotropy Option is specified, the following eight real parameters have to be added to the end of the input line (see example in B2.1):
<LnKs> Mean of lnKs with Ks in cm/s,
σ^2_{LnKs} Variance of lnKs with Ks in cm/s,
p Slope of the σ^2 versus lnKs regression line with Ks in cm/s,
β Parameter with Ks in cm/s,
λ Vertical correlation lengths for lnKs with Ks in cm/s,
Mean slope, γ, for lnKs versus λ with Ks in cm/s,
Upper Anisotropy Ratio Limit,
Lower Anisotropy Ration Limit,
Endif:

Aqueous Relative Permeability Card (cont'd)

If: Operational Mode Option = { **Water** } { **Fluid** }
Rock/Soil or Scaling Group Name[a],
Permeability Function Option[b], (Polmann Option and Anisotropy Option (in combination with Mualem only) may be specified)
{ Constant | Mualem [Modified] [Irreducible] |
Burdine | Fatt and Klikoff | Corey |
Haverkamp | Tauma and Vauclin |
Tabular [Linear | Spline] [Water Content | Saturation] }
If: Permeability Function Option = { Constant }
If: Rock/Soil or Scaling Group Name = {{ Fractured }} {{ DP }}
Matrix Aqueous Relative Permeability[c],
Fracture Aqueous Relative Permeability[d],
Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Else:
Aqueous Relative Permeability[c],
Format: *Char*[a], *Char*[b], *Real*[c],
Endif:
Elseif: Permeability Function Option = { Mualem } { Burdine }
If: Saturation Function Option = {{ van Genuchten }}
and Rock/Soil or Scaling Group Name = {{ Fractured }} {{ DP }}
Matrix van Genuchten m parameter[c],
Fracture van Genuchten m parameter[d],
Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Elseif: Saturation Function Option = {{ Brooks and Corey }}
and Rock/Soil or Scaling Group Name = {{ Fractured }} {{ DP }}
Matrix Brooks and Corey λ parameter[c],
Fracture Brooks and Corey λ parameter[d],
Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Elseif: Saturation Function Option = {{ van Genuchten }}
van Genuchten m parameter[c],
Format: *Char*[a], *Char*[b], *Real*[c],
Elseif: Saturation Function Option = {{ Brooks and Corey }}
Brooks and Corey λ parameter[c],
Format: *Char*[a], *Char*[b], *Real*[c],
Endif:
Elseif: Permeability Function Option = { Modified Mualem }
If: Saturation Function Option = {{ van Genuchten }}
van Genuchten m parameter[c], Pore-Scale Parameter[d] (0.5),
Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Elseif: Saturation Function Option = {{ Brooks and Corey }}
Brooks and Corey λ parameter[c], Pore-Scale Parameter[d] (0.5),
Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Endif:
Elseif: Permeability Function Option = { Irreducible Mualem }
If: Saturation Function Option = {{ van Genuchten }}

Aqueous Relative Permeability Card (cont'd)

van Genuchten m parameter[c], Irreducible Saturation[d],
Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Elseif: Saturation Function Option = {{ Brooks and Corey }}
Brooks and Corey λ parameter[c], Irreducible Saturation[d],

Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Endif:
Elseif: Permeability Function Option = { Fatt and Klikoff } { Corey }
Format: *Char*[a], *Char*[b],
Elseif: Permeability Function Option = { Haverkamp }
A Parameter[c], □ Parameter[d],
Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Elseif: Permeability Function Option = { Tauma and Vauclin }
□ Parameter[c], □ Parameter[d],
Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Elseif: Permeability Function Option = { Tabular Water Content | [Linear | Spline] }
Number of Table Entries[c],
Format: *Char*[a], *Char*[b], *Integer*[c],
For: Number of Table Entries
Aqueous Moisture Content[a], Aqueous Relative Permeability[b],
Format: *Real*[a], *Real*[b],
Endfor: Number of Table Entries
Elseif: Permeability Function Option = { Tabular [Saturation] [Linear | Spline] }
Number of Table Entries[c],
Format: *Char*[a], *Char*[b], *Integer*[c],
For: Number of Table Entries
Aqueous Saturation[a], Aqueous Relative Permeability[b],
Format: *Real*[a], *Real*[b],
Endfor: Number of Table Entries
Endif:

Aqueous Relative Permeability Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air** } { **Water-Air-Energy** } { **Water-Salt** } { **Water-Air-Salt** } { **Water-Air-Salt-Energy** }
Rock/Soil Name[a],
Permeability Function Option[b], (Polmann Anisotropy Option may be specified)
{ Constant | Mualem [Irreducible] | Burdine | Fatt and Klikoff | Corey | Haverkamp | Tauma and Vauclin |
Tabular [Linear | Spline] [Water Content | Saturation] }
If: Permeability Function Option = { Constant }
If: Rock/Soil Name = {{ Fractured }} {{ DP }}
Matrix Aqueous Relative Permeability[c],
Fracture Aqueous Relative Permeability[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Else:
Aqueous Relative Permeability[c],
Format: *Char[a], Char[b], Real[c],*
Endif:
Elseif: Permeability Function Option = { Mualem } { Burdine }
If: Saturation Function Option = {{ van Genuchten }}
and Rock/Soil Name = {{ Fractured }} {{ DP }}
Matrix van Genuchten m parameter[c],
Fracture van Genuchten m parameter[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Saturation Function Option = {{ Brooks and Corey }}
and Rock/Soil Name = {{ Fractured }} {{ DP }}
Matrix Brooks and Corey □ parameter[c],
Fracture Brooks and Corey □ parameter[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Saturation Function Option = {{ van Genuchten }}
van Genuchten m parameter[c],
Format: *Char[a], Char[b], Real[c],*
Elseif: Saturation Function Option = {{ Brooks and Corey }}
Brooks and Corey □ parameter[c],
Format: *Char[a], Char[b], Real[c],*
Endif:
Elseif: Permeability Function Option = { Irreducible Mualem }
If: Saturation Function Option = {{ van Genuchten }}
van Genuchten m parameter[c], Irreducible Saturation[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Saturation Function Option = {{ Brooks and Corey }}
Brooks and Corey □ parameter[c], Irreducible Saturation[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Endif:
Elseif: Permeability Function Option = { Fatt and Klikoff } { Corey }
Format: *Char[a], Char[b],*
Elseif: Permeability Function Option = { Haverkamp }

Aqueous Relative Permeability Card (cont'd)

A Parameter[c], □ Parameter[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Permeability Function Option = { Tauma and Vauclin }
□ Parameter[c], □ Parameter[d],

Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Elseif: Permeability Function Option = { Tabular Water Content | [Linear | Spline] }
Number of Table Entries[c],
Format: *Char*[a], *Char*[b], *Integer*[c],
For: Number of Table Entries
Aqueous Moisture Content[a], Aqueous Relative Permeability[b],
Format: *Real*[a], *Real*[b],
Endfor: Number of Table Entries
Elseif: Permeability Function Option = { Tabular [Saturation] [Linear | Spline] }
Number of Table Entries[c],
Format: *Char*[a], *Char*[b], *Integer*[c],
For: Number of Table Entries
Aqueous Saturation[a], Aqueous Relative Permeability[b],
Format: *Real*[a], *Real*[b],
Endfor: Number of Table Entries
Endif:

Aqueous Relative Permeability Card (cont'd)

Elseif: Operational Mode Option = { **Water-Oil** } { **Water-Air-Oil** } { **Water-Air-Oil-Energy** } { **Water-Oil-Dissolved Oil** }
Rock/Soil Name[a],
Permeability Function Option[b]
{ Mualem } { Burdine }
If: Saturation Function Option = {{ van Genuchten }}
van Genuchten m parameter[c],
Format: *Char*[a], *Char*[b], *Real*[c],
Elseif: Saturation Function Option = {{ Brooks and Corey }}
Brooks and Corey parameter[c],
Format: *Char*[a], *Char*[b], *Real*[c],
Endif:
Endif:

Aqueous Relative Permeability Card (cont'd)

Elseif: Operational Mode Option = { **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy** }
Rock/Soil Name,
Permeability Function Option[b], (Polmann Anisotropy Option may be specified)
{ Constant | Mualem [Irreducible] | Burdine | Fatt and Klikoff | Corey |
Free Corey | Haverkamp | Tauma and Vauclin |
Tabular [Linear | Spline] [Water Content | Saturation] }
If: Permeability Function Option = { Constant }
If: Rock/Soil Name= {{ Fractured }} {{ DP }}
Matrix Aqueous Relative Permeability[c],
Fracture Aqueous Relative Permeability[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Else:
Aqueous Relative Permeability[c],
Format: *Char[a], Char[b], Real[c],*
Endif:
Elseif: Permeability Function Option = { Mualem } { Burdine }
If: Saturation Function Option = {{ van Genuchten }}
and Rock/SoilName = {{ Fractured }} {{ DP }}
Matrix van Genuchten m parameter[c],
Fracture van Genuchten m parameter[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Saturation Function Option = {{ Brooks and Corey }}
and Rock/Soil Name = {{ Fractured }} {{ DP }}
Matrix Brooks and Corey □ parameter[c],
Fracture Brooks and Corey □ parameter[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Saturation Function Option = {{ van Genuchten }}
van Genuchten m parameter[c],
Format: *Char[a], Char[b], Real[c],*
Elseif: Saturation Function Option = {{ Brooks and Corey }}
Brooks and Corey □ parameter[c],
Format: *Char[a], Char[b], Real[c],*
Endif:
Elseif: Permeability Function Option = { Irreducible Mualem }
If: Saturation Function Option = {{ van Genuchten }}
van Genuchten m parameter[c], Irreducible Saturation[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Saturation Function Option = {{ Brooks and Corey }}
Brooks and Corey □ parameter[c], Irreducible Saturation[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Endif:
Elseif: Permeability Function Option = { Fatt and Klikoff } { Corey }
Format: *Char[a], Char[b],*
Elseif: Permeability Function Option = { Haverkamp }
A Parameter[c], □ Parameter[d],

Aqueous Relative Permeability Card (cont'd)

Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Permeability Function Option = { Tauma and Vauclin }
□ Parameter[c], □ Parameter[d],
Format: *Char[a], Char[b], Real[c], Real[d],*

Elseif: Permeability Function Option = { Free Corey }
Endpoint Gas Permeability[c], Exponent Gas Relative Permeability[d],
Residual Aqueous Saturation[e], Residual Gas Saturation[f],
Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d], *Real*[e], *Real*[f],
Elseif: Permeability Function Option = { Tabular Water Content | [Linear | Spline] }
Number of Table Entries[c],
Format: *Char*[a], *Char*[b], *Integer*[c],
For: Number of Table Entries
Aqueous Moisture Content[a], Aqueous Relative Permeability[b],
Format: *Real*[a], *Real*[b],
Endfor: Number of Table Entries
Elseif: Permeability Function Option = { Tabular [Saturation] [Linear | Spline] }
Number of Table Entries[c],
Format: *Char*[a], *Char*[b], *Integer*[c],
For: Number of Table Entries
Aqueous Saturation[a], Aqueous Relative Permeability[b],
Format: *Real*[a], *Real*[b],
Endfor: Number of Table Entries
Endif:

Endfor: Number of Rock/Soil Types

Endcard: Aqueous Relative Permeability Card

B.2.1 Aqueous Relative Permeability Card Examples

Extracted from a STOMP1 (Water) input file:
~Aqueous Relative Permeability Card
Sand,Mualem,,

Extracted from a STOMP1 (Water) input file:
~Aqueous Relative Permeability Card
20/30 Ottawa Sand,Mualem,0.56,

Extracted from a STOMP1 (Water) input file:
~Aqueous Relative Permeability Card
IJK Indexing,Mualem,file:data_m.dat,

Extracted from a STOMP1 (Water) input file:
~Aqueous Relative Permeability Card
Sand,Fatt And Klikoff,

Extracted from a STOMP1 (Water) input file:
#R1 is a scaling group
~Aqueous Relative Permeability Card
R1,Mualem,,

Extracted from a STOMP1 (Water) input file:
~Aqueous Relative Permeability Card
Backfill,Mualem w/Polmann,,15.76,3.56,1.1e4,1.84e4,30.0,0.00371,57.710,30.819,
H2 Sand,Mualem w/Polmann,,-14.59,1.50,-7.2e4,6.55e4,50.0,0.00620,48.363,5.682,
H1 Gravelly Sand,Mualem w/Polmann,,14.85,1.94,2.6e4,2.504,30.0,0.00368,17.866,7.92,

Extracted from a STOMP1 (Water) input file:
~Aqueous Relative Permeability Card
Backfill,Anisotropy Mualem,,0.5,0.5,

Extracted from a STOMP1 (Water) input file:
~Aqueous Rel
SM-ML1,Burdine,,
SW1,Burdine,,
SP3,Burdine,,
SM-SP1,Burdine,,
SP2,Burdine,,
SP1,Burdine,,
US,Touma and Vauclin,1.0,2.0,

Extracted from a STOMP1 (Water) input file:
~Aqueous Relative Permeability Card
backfill,Mualem,,
hanford sand,Mualem,,
plio-pleistocene,Mualem,,
upper ringold,Mualem,,
middle ringold,Mualem,,

B.3 Boundary Conditions Card

Card Title[a] {~Boundary [Conditions Card] }
Format: *Char*[a]

Number of Boundary Condition Domains[a],
Format: *Integer*[a],

For: Number of Boundary Condition Domains

Boundary Surface Direction Option[a],
{ Bottom } { South } { West } { East } { North } { Top } { File }

If: Operational Mode Option = { **Water** } { **Fluid** }
Aqueous-Phase Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux | Saturated |Unit Gradient
Free Gradient | Outflow |Hydraulic Gradient | Initial Condition |
Seepage Face | Falling Head | Falling Pond }
For: Number of Solutes
Solute Transport Boundary Type Option[c],
{ Volumetric Conc. | Aqueous Conc. | Zero Flux |
| Outflow | Initial Condition | Inflow Volumetric | Inflow Aqueous }
Endfor: Number of Solutes
Format: *Char*[a], *Char*[b], <*Char*[c],>

Elseif: Operational Mode Option = { **Water-Air** }
Aqueous-Phase Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux | Saturated | Unit Gradient |
Hydraulic Gradient | Initial Condition | Seepage Face |
Gas-Phase Boundary Type Option[c],
{ Dirichlet | Neumann | Zero Flux | Unit Gradient |
|Hydraulic Gradient | Initial Condition }
For: Number of Solutes
Solute Transport Boundary Type Option[d],
{ Volumetric Conc. | Aqueous Conc. | Gas Conc. |
|Zero Flux | Outflow | Initial Condition }
Endfor: Number of Solutes
Format: *Char*[a], *Char*[b], *Char*[c], <*Char*[d],>

Elseif: Operational Mode Option = { **Water-Air-Energy** }
Energy Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux |
| Outflow | Initial Condition | Ground | Convective | Convective-Radiative |
Shuttleworth-Wallace }
Aqueous-Phase Boundary Type Option[c],
{ Dirichlet | Neumann | Zero Flux | Saturated | Outflow |

Boundary Conditions Card (cont'd)

Unit Gradient | Hydraulic Gradient | Initial Condition | Seepage Face }
Gas-Phase Boundary Type Option[d],
{ Dirichlet | Neumann | Zero Flux |
Hydraulic Gradient | Initial Condition }

For: Number of Solutes
Solute Transport Boundary Type Option[e],
{ Volumetric Conc. | Aqueous Conc. | Gas Conc. |
Zero Flux | Outflow | Initial Condition }
Endfor: Number of Solutes
Format: *Char[a], Char[b], Char[c], Char[d], <Char[e],>*
Endif:

Elseif: Operational Mode Option = { **Water-Oil** }
Aqueous-Phase Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux | Outflow |
Unit Gradient | Hydraulic Gradient | Initial Condition }
NAPL Boundary Type Option[c],
{ Dirichlet | Neumann | Zero Flux | Outflow |
Unit Gradient | Hydraulic Gradient | Initial Condition }
For: Number of Solutes
Solute Transport Boundary Type Option[d],
{ Volumetric Conc. | Aqueous Conc. | NAPL Conc. |
Zero Flux | Outflow | Initial Condition |
Inflow Volumetric | Inflow Aqueous | Inflow NAPL }
Endfor: Number of Solutes
Format: *Char[a], Char[b], Char[c], <Char[d],>*

Elseif: Operational Mode Option Card = { **Water-Air-Oil** }
Aqueous-Phase Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux |
Unit Gradient | Hydraulic Gradient | Initial Condition | Fluctuating Water Table }
Gas-Phase Boundary Type Option[c],
{ Dirichlet Outflow | Dirichlet | Neumann | Zero Flux |
Unit Gradient |Hydraulic Gradient | Initial Condition }
NAPL Boundary Type Option[d],
{ Dirichlet | Neumann | Zero Flux | Outflow |
Unit Gradient | Hydraulic Gradient | Initial Condition }
For: Number of Solutes
Solute Transport Boundary Type Option[e],
{ Volumetric Conc. | Aqueous Conc. | Gas Conc. | NAPL Conc. |
Zero Flux | Outflow | Initial Condition | Inflow Volumetric |
Inflow Aqueous | Inflow Gas | Inflow NAPL }
Endfor: Number of Solutes
Format: *Char[a], Char[b], Char[c], Char[d], <Char[e],>*

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Oil-Energy** }
Energy Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux |
Outflow | Initial Condition }
Aqueous-Phase Boundary Type Option[c],
{ Dirichlet | Neumann | Zero Flux |
Unit Gradient | Hydraulic Gradient | Initial Condition }
Gas-Phase Boundary Type Option[d],
{ Dirichlet | Neumann | Zero Flux |

Unit Gradient | Hydraulic Gradient | Initial Condition }
NAPL Boundary Type Option[e],
{ Dirichlet | Neumann | Zero Flux |
Unit Gradient | Hydraulic Gradient | Initial Condition }
For: Number of Solutes
Solute Transport Boundary Type Option[f],
{ Volumetric Conc. | Aqueous Conc. | Gas Conc. | NAPL Conc. |
Zero Flux | Outflow | Initial Condition }
Endfor: Number of Solutes
Format: *Char[a], Char[b], Char[c], Char[d], Char[e], <Char[f],>*

Elseif: Operational Mode Option = { **Water-Oil-Dissolved Oil** }
Aqueous-Phase Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux | Outflow |
Unit Gradient | Hydraulic Gradient | Initial Condition }
NAPL Boundary Type Option[c],
{ Dirichlet | Neumann | Zero Flux | Outflow |
Unit Gradient | Hydraulic Gradient | Initial Condition }
Dissolved Oil Boundary Type Option[d],
{ Volumetric Conc | Aqueous Conc | Zero Flux |
Outflow | Initial Condition | Inflow Volumetric | Inflow Aqueous }
For: Number of Solutes
Solute Transport Boundary Type Option[e],
{ Volumetric Conc. | Aqueous Conc. | NAPL Conc. |
Zero Flux | Outflow | Initial Condition |
Inflow Volumetric | Inflow Aqueous }
Endfor: Number of Solutes
Format: *Char[a], Char[b], Char[c], Char[d], <Char[e],>*

Elseif: Operational Mode Option = { **Water-Salt** }
Aqueous-Phase Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux | Saturated | Unit Gradient |
Free Gradient | Hydraulic Gradient | Initial Condition }
Salt Boundary Type Option[c],
{ Volumetric Conc. | Aqueous Conc. | Zero Flux |
Outflow | Initial Condition | Inflow Volumetric | Inflow Aqueous |

Boundary Conditions Card (cont'd)

[Inflow] Relative Saturation }
For: Number of Solutes
Solute Transport Boundary Type Option[d],
{ Volumetric Conc. | Aqueous Conc. | Zero Flux |
Outflow | Initial Condition | Inflow Volumetric | Inflow Aqueous }
Endfor: Number of Solutes
Format: *Char[a], Char[b], Char[c], <Char[d],>*

Elseif: Operational Mode Option = { **Water-Air-Salt** }
Aqueous-Phase Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux | Unit Gradient |
Saturated | Hydraulic Gradient | Initial Condition }
Gas-Phase Boundary Type Option[c],
{ Dirichlet | Neumann | Zero Flux |
Hydraulic Gradient | Initial Condition }

Salt Boundary Type Option[d],
{ Volumetric Conc. | Aqueous Conc. | Zero Flux | Gas Conc. |
Outflow | Initial Condition | Inflow Volumetric | Inflow Aqueous |
[Inflow] Relative Saturation }
For: Number of Solutes
Solute Transport Boundary Type Option[e],
{ Volumetric Conc. | Aqueous Conc. | Gas Conc. |
Zero Flux | Outflow | Initial Condition }
Endfor: Number of Solutes
Format: *Char[a], Char[b], Char[c], Char[d], <Char[e],>*

Elseif: Operational Mode Option = { **Water-Air-Salt-Energy** }
Energy Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux |
Outflow | Initial Condition | Ground | Convective | Convective Radiative }
Aqueous-Phase Boundary Type Option[c],
{ Dirichlet | Neumann | Zero Flux | Saturated |
Unit Gradient | Hydraulic Gradient | Initial Condition }
Gas-Phase Boundary Type Option[d],
{ Dirichlet | Neumann | Zero Flux |
Hydraulic Gradient | Initial Condition }
Salt Boundary Type Option[e],
{ Volumetric Conc. | Aqueous Conc. | Zero Flux | Gas Conc. |
Outflow | Initial Condition | Inflow Volumetric | Inflow Aqueous |
[Inflow] Relative Saturation }
For: Number of Solutes
Solute Transport Boundary Type Option[f],
{ Volumetric Conc. | Aqueous Conc. | Gas Conc. |
Zero Flux | Outflow | Initial Condition }
Endfor: Number of Solutes

Boundary Conditions Card (cont'd)

Format: *Char[a], Char[b], Char[c], Char[d], Char[e], <Char[f],>*

Elseif: Operational Mode Option = { **Water-CO2-NaCl** }
Aqueous-Phase Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux |
Saturated | Hydraulic Gradient | Initial Condition }
Gas-Phase Boundary Type Option[c],
{ Dirichlet | Neumann | Zero Flux |
Hydraulic Gradient | Initial Condition }
Salt Boundary Type Option[d],
{ Volumetric Conc. | Aqueous Conc. | Zero Flux |
Outflow | Initial Condition |
Inflow Volumetric Conc. | Inflow Aqueous Conc. |
Inflow Mass Fraction | Inflow Relative Saturation |
Aqueous Relative Saturation | Aqueous Mass Fraction }
For: Number of Solutes
Solute Transport Boundary Type Option[e],
{ Volumetric Conc. | Aqueous Conc. | Gas Conc. |
Zero Flux | Outflow | Initial Condition }
Endfor: Number of Solutes
Format: *Char[a], Char[b], Char[c], Char[d], <Char[e],>*

Elseif: Operational Mode Option = { **Water-CO2-NaCl-Energy** }
Energy Boundary Type Option[b],
{ Dirichlet | Neumann | Zero Flux |
Outflow | Initial Condition | Ground | Convective | Convective Radiative }
Aqueous-Phase Boundary Type Option[c],
{ Dirichlet | Neumann | Zero Flux |
Saturated | Hydraulic Gradient | Initial Condition }
Gas-Phase Boundary Type Option[d],
{ Dirichlet | Neumann | Zero Flux |
Hydraulic Gradient | Initial Condition }
Salt Boundary Type Option[e],
{ Volumetric Conc. | Aqueous Conc. | Zero Flux |
Outflow | Initial Condition |
Inflow Volumetric Conc. | Inflow Aqueous Conc. |
Inflow Mass Fraction | Inflow Relative Saturation |
Aqueous Relative Saturation | Aqueous Mass Fraction }
For: Number of Solutes
Solute Transport Boundary Type Option[f],
{ Volumetric Conc. | Aqueous Conc. | Gas Conc. |
Zero Flux | Outflow | Initial Condition }
Endfor: Number of Solutes
Format: *Char[a], Char[b], Char[c], Char[d], Char[e], <Char[f],>*

Endif:

Boundary Conditions Card (cont'd)

I-Start Index[a], I-End Index[b], J-Start Index[c], J-End Index[d],
K-Start Index[e], K-End Index[f], Number of Boundary Times[g],
Format: *Integer[a], Integer[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g],*

If: Operational Mode Option = { **Water** } { **Fluid** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),
If: Aqueous Boundary Type Option =
{ Dirichlet } { Zero Flux } { Outflow }
Aqueous Pressure[c], Units[d] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[c], Units[d] (m/s),
Elseif: Aqueous-Phase Boundary Type Option =
{ Hydraulic Gradient } { Seepage Face }
Base Aqueous Pressure[c], Units[d] (Pa),
Elseif: Aqueous-Phase Boundary Type Option =
{ Falling Head } { Falling Pond }
Ponding Height[c], Units[d] (Pa),
Else:
Null[c], Null[d],
Endif:
For: Number of Solutes
If: Solute Transport Boundary Type Option =
{ [Inflow] Volumetric Conc. }
Solute Volumetric Conc[e], Units[f] (1/m^3),
Elseif: Solute Transport Boundary Type Option =
{ [Inflow] Aqueous Conc. }

Solute Aqueous-Phase Volumetric Conc[e], Units[f] (1/m^3),
Else:
Null[e], Null[f],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], < Real[e], Char[f], >*
Endfor: Number of Boundary Times

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[c], Units[d] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[c], Units[d] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient } { Seepage Face }
Base Aqueous Pressure[c], Units[d] (Pa),
Else:
Null[c], Null[d],
Endif:
Aqueous Dissolved-Air Relative Saturation[e],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[f], Units[g] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[f], Units[g] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Gas Aqueous Pressure[f], Units[g] (Pa),
Else:
Null[f], Null[g],
Endif:
Water-Vapor Relative Humidity[h],
For: Number of Solutes
If: Solute Transport Boundary Type Option = { Volumetric Conc. }
Solute Volumetric Conc[i], Units[j] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[i], Units[j] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Gas Conc. }
Solute Gas-Phase Volumetric Conc[i], Units[j] (1/m^3),
Else:
Null[i], Null[j],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Real[f], Char[g], Real[h], < Real[i], Char[j], >*
Endfor: Number of Boundary Times

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Energy** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),

If: Energy Boundary Type Option = { Shuttleworth-Wallace }
Volumetric Aqueous Flux[c], Units[d] (C),
For: Number of Plant Varietals
Leaf Area Index[e],
Endfor:
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[f], Units[g] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[f], Units[g] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
{ Seepage Face }
Base Aqueous Pressure[f], Units[g] (Pa),
Else:
Null[f], Null[g],
Endif:
Aqueous Dissolved-Air Relative Saturation[h],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[i], Units[j] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[i], Units[j] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Base Gas Pressure[i], Units[j] (Pa),
Else:
Null[i], Null[j],
Endif:
Water-Vapor Relative Humidity[k],
For: Number of Solutes
If: Solute Transport Boundary Type Option = { Volumetric Conc. }
Solute Volumetric Conc[l], Units[m] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[l], Units[m] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Gas Conc. }
Solute Gas Volumetric Conc[l], Units[m] (1/m^3),
Else:
Null[g], Null[h],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d],< Real[e],> Real[f], Char[g], Real[h], Real[i], Char[j], Real[k], < Real[l], Char[m], >*

Elseif: Energy Boundary Type Option = { Convective } { Convective-Radiative }

Boundary Conditions Card (cont'd)

If: Energy Boundary Type Option = { Convective }
Conv. Temperature[c], Units[d] (C), Conv. Heat Transfer Coeff.[e], Units[r] (kg/C s^3),
Elseif: Energy Boundary Type Option = { Convective-Radiative }
Conv. Temperature[c], Units[d] (C), Rad. Temperature[e], Units[r] (C),
Endif:

If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[g], Units[h] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[g], Units[h] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
{ Seepage Face }
Base Aqueous Pressure[g], Units[h] (Pa),
Else:
Null[g], Null[h],
Endif:
Aqueous Dissolved-Air Relative Saturation[i],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[j], Units[k] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[j], Units[k] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Base Gas Pressure[j], Units[k] (Pa),
Else:
Null[j], Null[k],
Endif:
Water-Vapor Relative Humidity[l],
For: Number of Solutes
If: Solute Transport Boundary Type Option = { Volumetric Conc. }
Solute Volumetric Conc[m], Units[n] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[m], Units[n] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Gas Conc. }
Solute Gas Volumetric Conc[m], Units[n] (1/m^3),
Else:
Null[m], Null[n],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j], Char[k], Real[l], < Real[m], Char[n], >*

Else:
If: Energy Boundary Type Option = { Dirichlet }
Temperature[c], Units[d] (C),
Elseif: Energy Boundary Type Option = { Neumann }
Energy Flux[c], Units[d] (W/m^2),

Boundary Conditions Card (cont'd)

Elseif: Energy Boundary Type Option = { Ground }
Air Temperature[c], Units[d] (C),
Else:
Null[c], Null[d],
Endif:
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[e], Units[f] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[e], Units[f] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
{ Seepage Face }
Base Aqueous Pressure[e], Units[f] (Pa),

Else:
Null[e], Null[f],
Endif:
Aqueous Dissolved-Air Relative Saturation[g],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[h], Units[i] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[h], Units[i] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Base Gas Pressure[i], Units[i] (Pa),
Else:
Null[h], Null[i],
Endif:
Water-Vapor Relative Humidity[j],
For: Number of Solutes
If: Solute Transport Boundary Type Option = { Volumetric Conc. }
Solute Volumetric Conc[k], Units[l] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[k], Units[l] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Gas Conc. }
Solute Gas Volumetric Conc[k], Units[l] (1/m^3),
Else:
Null[k], Null[l],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h], Char[i], Real[j], < Real[k], Char[l], >*
Endif:
Endfor: Number of Boundary Times

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Oil** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[c], Units[d] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[c], Units[d] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
Base Aqueous Pressure[c], Units[d] (Pa),
Else:
Null[c], Null[d],
Endif:
Aqueous Dissolved Oil Mass Fraction[e],
If: NAPL Boundary Type Option = { Dirichlet } { Zero Flux }
NAPL Pressure[f], Units[g] (Pa),
Elseif: NAPL Boundary Type Option = { Neumann }
NAPL Volumetric Flux[f], Units[g] (m/s),
Elseif: NAPL Boundary Type Option = { Hydraulic Gradient }
Base NAPL Pressure[f], Units[g] (Pa),
Else:
Null[f], Null[g],
Endif:
For: Number of Solutes
If: Solute Transport Boundary Type Option = { [Inflow] Volumetric Conc. }
Solute Volumetric Conc[h], Units[i] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { [Inflow] Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[h], Units[i] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { [Inflow] NAPL Conc. }
Solute NAPL Volumetric Conc[h], Units[i] (1/m^3),
Else:
Null[h], Null[i],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Real[f], Char[g], < Real[h], Char[i], >*
Endfor: Number of Boundary Times

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Oil** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[c], Units[d] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[c], Units[d] (m/s),
Elseif: Aqueous-Phase Boundary Type Option =
{ Hydraulic Gradient } { Fluctuating Water Table }
Base Aqueous Pressure[c], Units[d] (Pa),
Else:
Null[c], Null[d],
Endif:
Aqueous Dissolved Air Mass Fraction[e],
Aqueous Dissolved Oil Mass Fraction[f],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[g], Units[h] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[g], Units[h] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Base Gas Pressure[g], Units[h] (Pa),
Else:
Null[g], Null[h],
Endif:
Water Vapor Relative Humidity[i],
Oil Vapor Relative Humidity[j],
If: NAPL Boundary Type Option = { Dirichlet } { Zero Flux }
NAPL Pressure[k], Units[l] (Pa),
Elseif: NAPL Boundary Type Option = { Neumann }
NAPL Volumetric Flux[k], Units[l] (m/s),
Elseif: NAPL Boundary Type Option = { Hydraulic Gradient }
Base NAPL Pressure[k], Units[l] (Pa),
Else:
Null[k], Null[l],
Endif:
For: Number of Solutes
If: Solute Transport Boundary Type Option = { [Inflow] Volumetric Conc. }
Solute Volumetric Conc[m], Units[n] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { [Inflow] Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[m], Units[n] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { [Inflow] Gas Conc. }
Solute Gas-Phase Volumetric Conc[m], Units[n] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { [Inflow] NAPL Conc. }
Solute NAPL Volumetric Conc[m], Units[n] (1/m^3),
Else:

Boundary Conditions Card (cont'd)

Null[m], Null[n],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h],*
Real[i], Real[j], Real[k], Char[l], < Real[m], Char[n], >

Endfor: Number of Boundary Times

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Oil-Energy** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),
If: Energy Boundary Type Option = { Dirichlet }
Temperature[c], Units[d] (C)
Elseif: Energy Boundary Type Option = { Neumann }
Energy Flux[c], Units[d] (W/m^2)
Else:
Null[c], Null[d],
Endif:
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[e], Units[f] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[e], Units[f] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
Base Aqueous Pressure[e], Units[f] (Pa),
Else:
Null[e], Null[f],
Endif:
Aqueous Dissolved Air Mass Fraction[g],
Aqueous Dissolved Oil Mass Fraction[h],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[i], Units[j] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[i], Units[j] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Base Gas Pressure[i], Units[j] (Pa),
Else:
Null[i], Null[j],
Endif:
Water Vapor Relative Humidity[k],
Oil Vapor Relative Humidity[l],
If: NAPL Boundary Type Option = { Dirichlet } { Zero Flux }
NAPL Pressure[m], Units[n] (Pa),
Elseif: NAPL Boundary Type Option = { Neumann }
NAPL Volumetric Flux[m], Units[n] (m/s),
Elseif: NAPL Boundary Type Option = { Hydraulic Gradient }
Base NAPL Pressure[m], Units[n] (Pa),
Else:
Null[m], Null[n],
Endif:
For: Number of Solutes
If: Solute Transport Boundary Type Option = { Volumetric Conc. }
Solute Volumetric Conc[o], Units[p] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Aqueous Conc. }

Boundary Conditions Card (cont'd)

Solute Gas-Phase Volumetric Conc[o], Units[p] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { NAPL Conc. }
Solute NAPL Volumetric Conc[o], Units[p] (1/m^3),
Else:
Null[o], Null[p],

Endif:
Endfor: Number of Solutes
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Real*[h], *Real*[i], *Char*[j], *Real*[k], *Real*[l], *Real*[m], *Char*[n], < *Real*[o], *Char*[p], >
Endfor: Number of Boundary Times

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Oil-Dissolved Oil** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),
If: Aqueous-Phase Boundary Type Option = { Dirichlet }
Aqueous Pressure[c], Units[d] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[c], Units[d] (m/s),
Elseif: Aqueous-Phase Boundary Type = { Zero Flux }
Aqueous Pressure[c], Units[d] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
Base Aqueous Pressure[c], Units[d] (Pa),
Else: Null[c], Null[d],
Endif:
If: NAPL Boundary Type Option = { Dirichlet } { Zero Flux }
NAPL Pressure[e], Units[f] (Pa),
Elseif: NAPL Boundary Type Option = { Neumann }
NAPL Volumetric Flux[e], Units[f] (m/s),
Elseif: NAPL Boundary Type Option = { Hydraulic Gradient }
Base NAPL Pressure[e], Units[f] (Pa),
Else: Null[e], Null[f],
Endif:
If: Dissolved Oil Boundary Type Option = { [Inflow] Volumetric Conc. }
Dissolved Oil Volumetric Conc[g], Units[h] (kg/m^3),
Elseif: Dissolved Oil Boundary Type Option =
{ Aqueous Conc } { Inflow Aqueous }
Dissolved Oil Aqueous-Phase Volumetric Conc[g], Units[h] (kg/m^3),
Else: Null[g], Null[h],
Endif:
For: Number of Solutes:
If: Solute Transport Boundary Type Option = { [Inflow] Volumetric Conc. }
Solute Volumetric Conc[i], Units[j] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { [Inflow] Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[i], Units[j] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { NAPL Conc. }
Solute NAPL Volumetric Conc[i], Units[j] (1/m^3),
Else: Null[i], Null[j],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], <Real[i], Char[j],>*
Endfor: Number of Boundary Times

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Salt** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[c], Units[d] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[c], Units[d] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
Base Aqueous Pressure[c], Units[d] (Pa),
Else:
Null[c], Null[d],
Endif:
If: Salt Boundary Type Option = { Inflow Volumetric Conc. } { Volumetric Conc. }
Salt Volumetric Conc[e], Units[f] (kg/m^3),
Elseif: Salt Boundary Type Option =
{ Aqueous Conc. } { Inflow Aqueous }
Salt Aqueous-Phase Volumetric Conc[e], Units[f] (kg/m^3),
Elseif: Salt Boundary Type Option = { [Inflow] Relative Saturation }
Salt Relative Saturation[e], Null[f],
Else:
Null[e], Null[f],
Endif:
For: Number of Solutes
If: Solute Transport Boundary Type Option = { [Inflow] Volumetric Conc. }
Solute Volumetric Conc[g], Units[h] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { [Inflow] Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[g], Units[h] (1/m^3),
Else:
Null[g], Null[h],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], <Real[g], Char[h],>*
Endfor: Number of Boundary Times

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Salt** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[c], Units[d] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[c], Units[d] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
Base Aqueous Pressure[c], Units[d] (Pa),
Else:
Null[c], Null[d],
Endif:
Aqueous Dissolved-Air Relative Saturation[e],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[f], Units[g] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[f], Units[g] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Base Gas Pressure[f], Units[g] (Pa),
Else:
Null[f], Null[g],
Endif:
Water-Vapor Relative Humidity[h],
If: Salt Boundary Type Option = { Volumetric Conc. } { Inflow Volumetric }
Salt Volumetric Conc[i], Units[j] (kg/m^3),
Elseif: Salt Boundary Type Option = { Aqueous Conc. } { Inflow Aqueous }
Salt Aqueous-Phase Volumetric Conc[i], Units[j] (kg/m^3),
Elseif: Salt Boundary Type Option = { [Inflow] Relative Saturation }
Salt Relative Saturation[i], Null[j],
Else:
Null[i], Null[j],
Endif:
For: Number of Solutes
If: Solute Transport Boundary Type Option = { Volumetric Conc. }
Solute Volumetric Conc[k], Units[l] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[k], Units[l] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Gas Conc. }
Solute Gas Volumetric Conc[k], Units[l] (1/m^3),
Else:
Null[k], Null[l],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Real[f], Char[g], Real[h], Real[i], Char[j],*
< Real[k], Char[l], >

Boundary Conditions Card (cont'd)

Endfor: Number of Boundary Times

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Salt-Energy** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),
If: Energy Boundary Type Option = { Convective } { Convective-Radiative }
If: Energy Boundary Type Option = { Convective }
Conv. Temperature[c], Units[d] (C), Conv. Heat Transfer Coeff.[e], Units[r] (kg/C s^3),
Elseif: Energy Boundary Type Option = { Convective-Radiative }
Conv. Temperature[c], Units[d] (C), Rad. Temperature[e], Units[r] (C),
Endif:
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[g], Units[h] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[g], Units[h] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
Base Aqueous Pressure[g], Units[h] (Pa),
Else:
Null[g], Null[h],
Endif:
Aqueous Dissolved-Air Relative Saturation[i],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[j], Units[k] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[j], Units[k] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Base Gas Pressure[j], Units[k] (Pa),
Else:
Null[j], Null[k],
Endif:
Water-Vapor Relative Humidity[l],
If: Salt Boundary Type Option = { Volumetric Conc. }
Salt Volumetric Conc[m], Units[n] (kg/m^3),
Elseif: Salt Boundary Type Option = { Aqueous Conc. }
Salt Aqueous-Phase Volumetric Conc[m], Units[n] (kg/m^3),
Elseif: Salt Boundary Type Option = { Inflow Volumetric }
Salt Volumetric Conc[m], Units[n] (kg/m^3),
Elseif: Salt Boundary Type Option = { Inflow Aqueous }
Salt Aqueous-Phase Volumetric Conc[m], Units[n] (kg/m^3),
Elseif: Salt Boundary Type Option = { [Inflow] Relative Saturation }
Salt Relative Saturation[m], Null[n],
Else:
Null[m], Null[n],
Endif:
For: Number of Solutes

Boundary Conditions Card (cont'd)

If: Solute Transport Boundary Type Option = { Volumetric Conc. }
Solute Volumetric Conc[o], Units[p] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[o], Units[p] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Gas Conc. }

Solute Gas Volumetric Conc[o], Units[p] (1/m^3),
Else:
Null[o], Null[p],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j], Char[k], Real[l], Real[m], Char[n], < Real[o], Char[p], >*
Endif:

Else:
If: Energy Boundary Type Option = { Dirichlet }
Temperature[c], Units[d] (C),
Elseif: Energy Boundary Type Option = { Neumann }
Energy Flux[c], Units[d] (W/m^2),
Elseif: Energy Boundary Type Option = { Ground }
Air Temperature[c], Units[d] (C),
Else:
Null[c], Null[d],
Endif:
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[e], Units[f] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[e], Units[f] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
Base Aqueous Pressure[e], Units[f] (Pa),
Else:
Null[e], Null[f],
Endif:
Aqueous Dissolved-Air Relative Saturation[g],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[h], Units[i] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[h], Units[i] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Base Gas Pressure[h], Units[i] (Pa),
Else:
Null[h], Null[i],
Endif:
Water-Vapor Relative Humidity[j]
If: Salt Boundary Type Option = { Volumetric Conc. }

Boundary Conditions Card (cont'd)

Salt Volumetric Conc[k], Units[l] (kg/m^3),
Elseif: Salt Boundary Type Option = { Aqueous Conc. }
Salt Aqueous-Phase Volumetric Conc[k], Units[l] (kg/m^3),
Elseif: Salt Boundary Type Option = { Inflow Volumetric }
Salt Volumetric Conc[k], Units[l] (kg/m^3),
Elseif: Salt Boundary Type Option = { Inflow Aqueous }
Salt Aqueous-Phase Volumetric Conc[k], Units[l] (kg/m^3),
Elseif: Salt Boundary Type Option = { [Inflow] Relative Saturation }
Salt Relative Saturation[k], Null[l],
Else:
Null[k], Null[l],
Endif:

For: Number of Solutes
If: Solute Transport Boundary Type Option = { Volumetric Conc. }
Solute Volumetric Conc[m], Units[n] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[m], Units[n] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Gas Conc. }
Solute Gas Volumetric Conc[m], Units[n] (1/m^3),
Else:
Null[m], Null[n],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h], Char[i], Real[j], Real[k], Char[l], < Real[m], Char[n], >*
Endif:
Endfor: Number of Boundary Times

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-CO2-NaCl** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[c], Units[d] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[c], Units[d] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
Base Aqueous Pressure[c], Units[d] (Pa),
Else:
Null[c], Null[d],
Endif:
Aqueous Dissolved- CO_2-Relative Saturation[e],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[f], Units[g] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[f], Units[g] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Base Gas Pressure[f], Units[g] (Pa),
Else:
Null[f], Null[g],
Endif:
Water-Vapor Relative Humidity[h],
If: Salt Boundary Type Option = { [Inflow] Volumetric Conc. }
Salt Volumetric Conc[i], Units[j] (kg/m^3),
Elseif: Salt Boundary Type Option = { [Inflow] Aqueous Conc. }
Salt Aqueous-Phase Volumetric Conc[i], Units[j] (kg/m^3),
Elseif: Salt Boundary Type Option =
{ Inflow Aqueous Saturation } { Inflow Relative Saturation }
Salt Relative Saturation[i], Null[j],
Elseif: Salt Boundary Type Option =
{ Inflow Mass Fraction } { Aqueous Mass Fraction }
Salt Mass Fraction[i], Null[j],
Else:
Null[i], Null[j],
Endif:
For: Number of Solutes
If: Solute Transport Boundary Type Option = { Volumetric Conc. }
Solute Volumetric Conc[k], Units[l] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[k], Units[l] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Gas Conc. }
Solute Gas Volumetric Conc[k], Units[l] (1/m^3),
Else:
Null[k], Null[l],

Boundary Conditions Card (cont'd)

Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Real[f], Char[g], Real[h], Real[i], Char[j],*
< Real[k], Char[l], >
Endfor: Number of Boundary Times

Boundary Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-CO2-NaCl-Energy** }
For: Number of Boundary Times
Boundary Time[a], Units[b] (s),
If: Energy Boundary Type Option = { Convective } { Convective-Radiative }
If: Energy Boundary Type Option = { Convective }
Conv. Temperature[c], Units[d] (C), Conv. Heat Transfer Coeff.[e], Units[r] (kg/C s^3),
Elseif: Energy Boundary Type Option = { Convective-Radiative }
Conv. Temperature[c], Units[d] (C), Rad. Temperature[e], Units[r] (C),
Endif:
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[g], Units[h] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[g], Units[h] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
Base Aqueous Pressure[g], Units[h] (Pa),
Else:
Null[g], Null[h],
Endif:
Aqueous Dissolved-CO_2 Relative Saturation[i],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[j], Units[k] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[j], Units[k] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Base Gas Pressure[j], Units[k] (Pa),
Else:
Null[j], Null[k],
Endif:
Water-Vapor Relative Humidity[l],
If: Salt Boundary Type Option = { [Inflow] Volumetric Conc. }
Salt Volumetric Conc[m], Units[n] (kg/m^3),
Elseif: Salt Boundary Type Option = { [Inflow] Aqueous Conc. }
Salt Aqueous-Phase Volumetric Conc[m], Units[n] (kg/m^3),
Elseif: Salt Boundary Type Option =
{ Inflow Aqueous Saturation } { Inflow Relative Saturation }
Salt Relative Saturation[m], Null[n],
Elseif: Salt Boundary Type Option =
{ Inflow Mass Fraction } { Aqueous Mass Fraction }
Salt Mass Fraction[m], Null[n],
Else:
Null[m], Null[n],
Endif:
For: Number of Solutes
If: Solute Transport Boundary Type Option = { Volumetric Conc. }
Solute Volumetric Conc[o], Units[p] (1/m^3),

Boundary Conditions Card (cont'd)

Elseif: Solute Transport Boundary Type Option = { Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[o], Units[p] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Gas Conc. }
Solute Gas Volumetric Conc[o], Units[p] (1/m^3),
Else:

Null[o], Null[p],
Endif:
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j], Char[k], Real[l], Real[m], Char[n], < Real[o], Char[p], >*
Endif:

Else:
If: Energy Boundary Type Option = { Dirichlet }
Temperature[c], Units[d] (C),
Elseif: Energy Boundary Type Option = { Neumann }
Energy Flux[c], Units[d] (W/m^2),
Elseif: Energy Boundary Type Option = { Ground }
Air Temperature[c], Units[d] (C),
Else:
Null[c], Null[d],
Endif:
If: Aqueous-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Aqueous Pressure[e], Units[f] (Pa),
Elseif: Aqueous-Phase Boundary Type Option = { Neumann }
Aqueous Volumetric Flux[e], Units[f] (m/s),
Elseif: Aqueous-Phase Boundary Type Option = { Hydraulic Gradient }
Base Aqueous Pressure[e], Units[f] (Pa),
Else:
Null[e], Null[f],
Endif:
Aqueous Dissolved-Air Relative Saturation[g],
If: Gas-Phase Boundary Type Option = { Dirichlet } { Zero Flux }
Gas Pressure[h], Units[i] (Pa),
Elseif: Gas-Phase Boundary Type Option = { Neumann }
Gas Volumetric Flux[h], Units[i] (m/s),
Elseif: Gas-Phase Boundary Type Option = { Hydraulic Gradient }
Base Gas Pressure[h], Units[i] (Pa),
Else:
Null[h], Null[i],
Endif:
Water-Vapor Relative Humidity[j]
If: Salt Boundary Type Option = { [Inflow] Volumetric Conc. }
Salt Volumetric Conc[k], Units[l] (kg/m^3),
Elseif: Salt Boundary Type Option = { [Inflow] Aqueous Conc. }

Boundary Conditions Card (cont'd)

Salt Aqueous-Phase Volumetric Conc[k], Units[l] (kg/m^3),
Elseif: Salt Boundary Type Option =
{ Inflow Aqueous Saturation } { Inflow Relative Saturation }
Salt Relative Saturation[k], Null[l],
Elseif: Salt Boundary Type Option =
{ Inflow Mass Fraction } { Aqueous Mass Fraction }
Salt Mass Fraction[k], Null[l],
Else:
Null[k], Null[l],
Endif:
For: Number of Solutes
If: Solute Transport Boundary Type Option = { Volumetric Conc. }

Solute Volumetric Conc[m], Units[n] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Aqueous Conc. }
Solute Aqueous-Phase Volumetric Conc[m], Units[n] (1/m^3),
Elseif: Solute Transport Boundary Type Option = { Gas Conc. }
Solute Gas Volumetric Conc[m], Units[n] (1/m^3),
Else:
Null[m], Null[n],
Endif:
Endfor: Number of Solutes
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Real*[h], *Char*[i], *Real*[j], *Real*[k], *Char*[l], < *Real*[m], *Char*[n], >
Endif:
Endfor: Number of Boundary Times

Endif:

Endfor: Number of Boundary Condition Domains

Endcard: Boundary Conditions Card

B.3.1 Boundary Conditions Card Examples

Extracted from a STOMP1 (Water) input file:

```
#--------------------------------------------------
~Boundary Conditions Card
#--------------------------------------------------
2,
East,Hydraulic Gradient,Aqueous Conc,
50,50,1,1,1,113,1,
0,day,183254,Pa,1.,1/ft^3,
West,Neumann,Outflow,
3,3,1,1,14,33,4,
0,min,-0.00021209,ft/sec,,,
37.4675,min,-0.00021209,ft/sec,,,
74.9232,min,-0.000200628,ft/sec,,,
112.379,min,-0.000201526,ft/sec,,,
```

Extracted from a STOMP2 (Water-Air) input file:

```
~Boundary Conditions Card
3,
Top,Zero Flux,Dirichlet,Zero Flux,
1,60,1,1,20,20,1,
0,s,,,,101325,Pa,1.0,,,
West,Hydraulic Gradient,Zero Flux,Outflow,
1,1,1,1,1,20,1,
0,s,106173.84,Pa,0.0,,,,,,
East,Hydraulic Gradient,Zero Flux,Outflow,
60,60,1,1,1,20,1,
0,s,105880.39,Pa,0.0,,,,,,
```

Extracted from a STOMP3 (Water-Air-Energy) input file:

```
~Boundary Conditions Card
2,
West,Dirichlet Energy,Dirichlet Aqueous,Dirichlet Gas,
1,1,1,1,1,1,1,
0,day,70,C,101330,Pa,0,101330,Pa,1,
East,Neumann Energy,Zero Flux Aqueous,Zero Flux Gas,
50,50,1,1,1,1,1,
0,Day,-100,W/m^2,,,,,,
```

Extracted from a STOMP3 (Water-Air-Energy) input file:

```
~Boundary Conditions Card
2,
Top,Zero Flux,Dirichlet,
1,1,1,1,91,91,2,
0,hr,-1.e9,Pa,1.,101758.43,Pa,
25,d,-1.e9,Pa,1.,101758.43,Pa,
Bottom,Dirichlet,Zero Flux,
1,1,1,1,1,1,2,
0,s,110629,Pa,1.0,,,
10,min,109595.22,Pa,1.0,,,
```

Boundary Conditions Card Examples (cont'd)

Extracted from a STOMP5 (Water-Air-Oil) input file:

```
1,
Top,Neumann,Dirichlet,Neumann,
13,15,13,17,79,79,18,
1955.5,yr,-3.04,m/yr,,,101325.0,Pa,,,-0.062,m/yr,
1956,yr,-3.04,m/yr,,,101325.0,Pa,,,-0.062,m/yr,
1956,yr,-2.47,m/yr,,,101325.0,Pa,,,-0.277,m/yr,
1957,yr,-2.47,m/yr,,,101325.0,Pa,,,-0.277,m/yr,
1957,yr,-2.95,m/yr,,,101325.0,Pa,,,-0.277,m/yr,
1958,yr,-2.95,m/yr,,,101325.0,Pa,,,-0.277,m/yr,
1958,yr,-3.92,m/yr,,,101325.0,Pa,,,-0.264,m/yr,
1959,yr,-3.92,m/yr,,,101325.0,Pa,,,-0.264,m/yr,
1959,yr,-3.07,m/yr,,,101325.0,Pa,,,-0.283,m/yr,
1960,yr,-3.07,m/yr,,,101325.0,Pa,,,-0.283,m/yr,
1960,yr,-3.42,m/yr,,,101325.0,Pa,,,-0.283,m/yr,
1961,yr,-3.42,m/yr,,,101325.0,Pa,,,-0.286,m/yr,
1961,yr,-4.23,m/yr,,,101325.0,Pa,,,-0.379,m/yr,
1962,yr,-4.23,m/yr,,,101325.0,Pa,,,-0.379,m/yr,
1962,yr,-1.98,m/yr,,,101325.0,Pa,,,-0.178,m/yr,
1962.5,yr,-1.98,m/yr,,,101325.0,Pa,,,-0.178,m/yr,
1962.5,yr,-0.0,m/yr,,,101325.0,Pa,,,-0.0,m/yr,
2003,yr,-0.0,m/yr,,,101325.0,Pa,,,-0.0,m/yr,
```

Extracted from a STOMP11 (Water-Salt) input file:

```
~Boundary Conditions Card
1,
Bottom,Dirichlet,Outflow,
1,90,1,1,1,1,1,
0.0,min,101325.,Pa,,,
```

Extracted from a STOMP33 (Water-CO2-NaCl-Energy) input file:

```
~Boundary Conditions Card
1,
South,Energy Dirichlet,Aqu. Dirichlet,Gas Dirichlet,Aqu. Mass Frac.,
98,98,1,1,3,3,1,
0,s,45.0,C,138.0,bar,0.0,138.0,bar,1.0,0.0,,
```

B.4 Directional Aqueous Relative Permeability Card

Card Title[a] { ~X-Aqueous Rel [ative Permeability Card] } and/or
Card Title[a] { ~Y-Aqueous Rel [ative Permeability Card] } and/or
Card Title[a] { ~Z-Aqueous Rel [ative Permeability Card] }

For each directional card:

Format: *Char*[a]

For each directional aqueous relative permeability card, IJK indexing can be used:

For: Number of Rock/Soil Types or Scaling Groups

If: Rock/Soil or Scaling Group Name = { IJK | JKI | KIJ } Indexing

Note: A parameter value input can be replaced with an external file using the following formatting for ASCII files:

file: *filename*

or the following formattings for binary files:

binary file: *filename*

where; the external file will contain unique parameter values for each node (active or inactive) arranged according to the indexing scheme (i.e., IJK, JKI, or KIJ). Applicable units will be applied to all parameter values in the external file. Example input cards are included in section B.4.1

Endif:

The format of the line input for each mode is identical to the Aqueous Relative Permeability Card.

B.4.1 Directional Aqueous Relative Permeability Card Examples

Extracted from STOMP1 (Water) input file:
~X-Direction Aqueous Relative Permeabiliy Card
IJK Indexing, Tabular, 12, file: tabh.dat, file: tabkrx.dat

Extracted from STOMP1 (Water) input file:
~Y-Direction Aqueous Relative Permeabiliy Card
IJK Indexing, Tabular, 12, file: tabh.dat, file: tabkry.dat

Extracted from STOMP1 (Water) input file:
~Z-Direction Aqueous Relative Permeabiliy Card
IJK Indexing, Tabular, 12, file: tabh.dat, file: tabkrz.dat

B.5 Dissolved-Oil Transport Card

Card Title[a] { ~Dissolved Oil Transport [Card] } {~Dissolved VOC Transport [Card] }
{ ~Dissolved Organic Transport [Card] }
Format: *Char*[a]

If: Operational Mode ={ **Water-Oil-Dissolved Oil** }
Interphase Mass Transfer Function Option[a],

{ Welty } **Note:** $Sh = \dfrac{k_{n\ell}^{o} d_p}{D_\ell^{o}} = a + bRe^m Sc^n$

{ Parker } **Note:** $Sh = \dfrac{k_{n\ell}^{o} d_p}{D_\ell^{o}} = a + bRe^m \left(s_n n_D\right)^n$

Constant a[b], Constant b[c], Constant m[d], Constant n[e],
Format: *Char*[a], *Real*[b], *Real*[c], *Real*[d], *Real*[e],
Endif:

For: Number of Rock/Soil Types

If: Operational Mode = { **Water-Oil** } { **Water-Air-Oil** }
Rock/Soil Name[a],
Longitudinal Dispersivity[b], Units[c] (m),
Transverse Dispersivity[d], Units[e] (m),
Dissolved-Oil Adsorption Function Option[f],

{ Linear } **Note:** $R_D = \left[1 + \dfrac{k}{s_\ell n_D}\right]$

{ Linear Kd } **Note:** $R_D = \left[1 + \dfrac{K_d \rho_b}{s_\ell n_D}\right]$

{ Freundlich } **Note:** $R_D = \left[1 + \dfrac{nkC_\ell^{n-1}}{s_\ell n_D}\right]$

{ Langmuir } **Note:** $R_D = \left[1 + \dfrac{a}{s_\ell n_D (1 + bC_\ell)^2}\right]$

If: Dissolved-Oil Adsorption Function Option = { Linear }
Constant k[g],
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Char*[e], *Char*[f], *Real*[g],

Dissolved-Oil Transport Card (cont'd)

Elseif: Dissolved-Oil Adsorption Function Option = { Linear Kd }
Constant Kd[g], Units[h] (m^3/kg),

Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Char[h],*
Elseif: Dissolved-Oil Adsorption Function Option = { Freundlich }
Constant k[g],
Constant n[h],
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Real[h],*
Elseif: Dissolved-Oil Adsorption Function Option = { Langmuir }
Constant a[g], Constant b[h], Units[i] (m^3/kg),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Real[h], Char[i],*
Endif:

Endif:

Elseif: Operational Mode = { **Water-Oil-Dissolved Oil** }

Rock/Soil Name[a],
Longitudinal Dispersivity[b], Units[c] (m),
Transverse Dispersivity[d], Units[e] (m),
Dissolved-Oil Adsorption Function Option[f],

{ Linear } **Note:** $R_D = \left[1+\frac{k}{s_\ell n_D}\right]$

{ Linear Kd } **Note:** $R_D = \left[1+\frac{K_d \rho_b}{s_\ell n_D}\right]$

{ Freundlich } **Note:** $R_D = \left[1+\frac{nkC_\ell^{n-1}}{s_\ell n_D}\right]$

{ Langmuir } **Note:** $R_D = \left[1+\frac{a}{s_\ell n_D (1+bC_\ell)^2}\right]$

If: Dissolved-Oil Adsorption Function Option = { Linear }
Constant k[g], Nominal Particle Diameter[h], Units[i] (m),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Real[h], Char[i],*
Elseif: Dissolved-Oil Adsorption Function Option = { Linear Kd }
Constant Kd[g], Units[h] (m^3/kg), Nominal Particle Diameter[i], Units[j] (m),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Char[h], Real[i], Char[j],*
Elseif: Dissolved-Oil Adsorption Function Option = { Freundlich }
Constant k[g], Constant n[h], Nominal Particle Diameter[i], Units[j] (m),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Real[h], Real[h], Char[i],*
Elseif: Dissolved-Oil Adsorption Function Option = { Langmuir }

Dissolved-Oil Transport Card (cont'd)

Constant a[g], Constant b[h], Units[i] (m^3/kg), Nominal Particle Diameter[j], Units[k] (m),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Real[h], Char[i], Real[j], Char[k],*
Endif:
Endif:

Endfor: Number of Rock/Soil Types

Endcard: Dissolved-Oil Transport Card

B.5.1 Dissolved-Oil Transport Card Examples

Extracted from a STOM4 (Water Oil) input file:
~Dissolved Oil Transport Card
Sand,0.17,cm,,cm,linear kd,0.0,m^3/kg,

Extracted from a STOMP8 (Water-Oil-Dissolved Oil) input file:
~Dissolved Oil Transport Card
Welty,0.55,0.25,1.5,1.5,
20/30 Ottawa Sand,0.154,cm,0.0154,cm,Linear Kd,0.,m^3/kg,0.71,mm,

B.6 Gas Relative Permeability Card

Card Title[a] { ~Gas Rel [ative Permeability Card] }
Format: *Char*[a]

For: Number of Rock/Soil Types

If: Rock/Soil Name = { IJK | JKI | KIJ } Indexing

Note: A parameter value input can be replaced with an external file using the following formatting for ASCII files:

file: *filename*

or the following formattings for binary files:

binary file: *filename*

where; the external file will contain unique parameter values for each node (active or inactive) arranged according to the indexing scheme (i.e., IJK, JKI, or KIJ). Applicable units will be applied to all parameter values in the external file. An example input card is included in section B.6.1

Endif:

For: Number of Rock/Soil Types

If: Operational Mode ={ **Water-Air** } { **Water-Air-Energy** } { **Water-Air-Salt** } { **Water-Air-Salt-Energy** }
Rock/Soil Name[a],
Permeability Function Option[b],
{ Constant | Mualem | Burdine | Fatt and Klikoff | Corey | Tabular [Linear | Spline] [Water Content | Saturation] }
If: Permeability Function Option = { Constant }
If: Rock/Soil Name = {{ Fractured }} {{ DP }}
Matrix Gas Relative Permeability[c],
Fracture Gas Relative Permeability[d],
Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Else:
Gas Relative Permeability[c]
Format: *Char*[a], *Char*[b], *Real*[c],
Endif:
Elseif: Permeability Function Option = { Mualem } { Burdine }
If: Saturation Function Option = {{ van Genuchten }} and Rock/Soil Name = {{ Fractured }} {{ DP }}
Matrix van Genuchten m parameter[c],

Gas Permeability Card (cont'd)

Fracture van Genuchten m parameter[d],
Format: *Char*[a], *Char*[b], *Real*[c], *Real*[d],
Elseif: Saturation Function Option = {{ Brooks and Corey }} and Rock/Soil Type Name = {{ Fractured }} {{ DP }}

Matrix Brooks and Corey parameter[c],
Fracture Brooks and Corey parameter[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Saturation Function Option = {{ van Genuchten }}
van Genuchten m parameter[c],
Format: *Char[a], Char[b], Real[c],*
Elseif: Saturation Function Option = {{ Brooks and Corey }}
Brooks and Corey parameter[c],
Format: *Char[a], Char[b], Real[c],*
Endif:
Elseif: Permeability Function Option = { Corey }
Irreducible Gas Saturation[c], Irreducible Aqueous Saturation[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Permeability Function Option = { Fatt and Klikoff }
Format: *Char[a], Char[b],*
Elseif: Permeability Function Option = { Tabular Water Content [Linear | Spline] }
Number of Table Entries[c],
Format: *Char[a],Char[b],Integer[c],*
For: Number of Table Entries
Water Content[a], Gas Relative Permeability[b],
Format: *Real[a], Real[b],*
Endfor:
Elseif: Permeability Function Option = { Tabular [Saturation] [Linear | Spline] }
Number of Table Entries[c],
Format: *Char[a],Char[b],Integer[c],*
For: Number of Table Entries
Saturation[a], Gas Relative Permeability[b],
Format: *Real[a], Real[b],*
Endfor:
Endif:

Elseif: Operational Mode = { **Water-Air-Oil** } { **Water-Air-Oil-Energy** }
Rock/Soil Name[a],
Permeability Function Option[b],
{ Mualem | Burdine }
If: Saturation Function Option = { van Genuchten }
van Genuchten m parameter[c],
Format: *Char[a], Char[b], Real[c],*
Elseif: Saturation Function Option = { Brooks and Corey }
Brooks and Corey parameter[c],
Format: *Char[a], Char[b], Real[c],*

Gas Permeability Card (cont'd)

Endif:
Endif:

Elseif: Operational Mode = { **Water-Air-Oil** } { **Water-Air-Oil-Energy** }
Rock/Soil Name[a],
Permeability Function Option[b],
{ Mualem | Burdine }
If: Saturation Function Option = {{ van Genuchten }}
van Genuchten m parameter[c],
Format: *Char[a], Char[b], Real[c],*

Elseif: Saturation Function Option = {{ Brooks and Corey }}
Brooks and Corey □ parameter[c],
Format: *Char[a], Char[b], Real[c],*
Endif:
Endif:

Elseif: Operational Mode ={ **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy** }
Rock/Soil Name[a],
Permeability Function Option[b],
{ Constant | Mualem | Burdine | Fatt and Klikoff | Corey | Free Corey |
| Tabular [Linear | Spline] [Water Content | Saturation] }
If: Permeability Function Option = { Constant }
If: Rock/Soil Name = {{ Fractured }} {{ DP }}
Matrix Gas Relative Permeability[c],
Fracture Gas Relative Permeability[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Else:
Gas Relative Permeability[c]
Format: *Char[a], Char[b], Real[c],*
Endif:
Elseif: Permeability Function Option = { Mualem } { Burdine }
If: Saturation Function Option = {{ van Genuchten }}
and Rock/Soil Name = {{ Fractured }} {{ DP }}
Matrix van Genuchten m parameter[c],
Fracture van Genuchten m parameter[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Saturation Function Option = {{ Brooks and Corey }}
and Rock/Soil Name = {{ Fractured }} {{ DP }}
Matrix Brooks and Corey □ parameter[c],
Fracture Brooks and Corey □ parameter[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Saturation Function Option = {{ van Genuchten }}
van Genuchten m parameter[c],
Format: *Char[a], Char[b], Real[c],*
Elseif: Saturation Function Option = {{ Brooks and Corey }}

Gas Permeability Card (cont'd)

Brooks and Corey □ parameter[c],
Format: *Char[a], Char[b], Real[c],*
Endif:
Elseif: Permeability Function Option = { Corey }
Irreducible Gas Saturation[c], Irreducible Aqueous Saturation[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Elseif: Permeability Function Option = { Free Corey }
Endpoint Gas Permeability[c], Exponent Gas Relative Permeability[d],
Residual Aqueous Saturation[e], Residual Gas Saturation[f],
Format: *Char[a], Char[b], Real[c], Real[d], Real[e], Real[f],*
Elseif: Permeability Function Option = { Fatt and Klikoff }
Format: *Char[a], Char[b],*
Number of Table Entries[c],
Format: *Char[a],Char[b],Integer[c],*
For: Number of Table Entries
Water Content[a], Gas Relative Permeability[b],

Format: *Real*[a], *Real*[b],
Endfor:
Elseif: Permeability Function Option = { Tabular [Saturation] [Linear | Spline] }
Number of Table Entries[c],
Format: *Char*[a],*Char*[b],*Integer*[c],
For: Number of Table Entries
Saturation[a], Gas Relative Permeability[b],
Format: *Real*[a], *Real*[b],
Endfor:
Endif:

Endfor: Number of Rock/Soil Types

Endcard: Gas Relative Permeability Card

B.6.1 Gas Relative Permeability Examples

Extracted from a STOMP2 (Water-Air) input file:
~Gas Relative Permeability Card
Sand,Mualem,,

Extracted from a STOMP2 (Water-Air) input file:
~Gas Relative Permeability Card
IJK Indexing,Constant,file:rel_g_x.dat,

Extracted from a STOMP2 (Water-Air) input file:
~Gas Rel
20/30 Ottawa Sand,Mualem,0.56,

Extracted from a STOMP2 (Water-Air) input file:
~Gas Relative Permeability Card
Sand,Fatt and Klikoff,

Extracted from a STOMP2 (Water-Air) input file:
~Gas Relative Permeability
SM-ML1,Burdine,,
SW1,Burdine,,
Tabular Input
SP3,Tabular,6,
1.0, 1.0,
0.8, 0.64,
0.6, 0.36,
0.4, 0.16,
0.2, 0.04,
0.0, 0.0,
SM-SP1,Mualem,0.5,
SP2,Burdine,,
SP1,Burdine,,

B.7 Grid Card

Card Title[a] { ~Grid [Card] }
Format: *Char*[a]

Coordinate System Option[a],
{ [Uniform | Tilted] [Reference [Point]] Cartesian |
[Uniform] [Reference [Point]] Cylindrical |
Generalized }

If: Coordinate System Option = { Tilted [Cartesian] }
X-Z Plane Tilt Angle[b], Units[c] (deg),
Y-Z Plane Tilt Angle[d], Units[e] (deg),
If: Coordinate System Option = [Reference]
X Reference Point[f], Units[g] (m), Direction Index[h],
Y Reference Point[i], Units[j] (m), Direction Index[k],
Z Reference Point[l], Units[m] (m), Direction Index[n],
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Char*[e], *Real*[f], *Char*[g], *Integer*[h],
Real[i], *Char*[j], *Integer*[k], *Real*[l], *Char*[m], *Integer*[n],
Elseif:
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Char*[e],
Endif:
Elseif:
If: Coordinate System Option = [Reference]
X Reference Point[b], Units[c] (m), Direction Index[d],
Y Reference Point[e], Units[f] (m), Direction Index[g],
Z Reference Point[h], Units[i] (m), Direction Index[j],
Format: *Char*[a], *Real*[b], *Char*[c], *Integer*[d], *Real*[e], *Char*[f], *Integer*[g], *Real*[h], *Char*[i], *Integer*[j],
Elseif:
Format: *Char*[a],
Endif:
Endif:

Note: A negative direction index implies a descending direction.

If: Coordinate System Option = { Generalized }
External File Name[a], Dimensional Units[b] (m),
Format: *Char*[a], *Char*[b],
Elseif:
Number of X-Dir. Nodes[a],
Number of Y-Dir. Nodes[b],
Number of Z-Dir. Nodes[c],
Format: *Integer*[a], *Integer*[b], *Integer*[c],
Endif:

Grid Card (cont'd)

If: Coordinate System Option = { Cartesian }

For: Number of X-Dir. Nodes
< Surface Position[a], Units[b] (m), > or < Count[a] @ Node Width[b], Units[c] (m), >
Format: < *Real[a], Char[b],* > or < *Integer[a]@Real[a], Char[b],* >
Endfor: Number of X-Dir. Nodes

For: Number of Y-Dir. Nodes
< Surface Position[a], Units[b] (m), > or < Count[a] @ Node Width[b], Units[c] (m), >
Format: < *Real[a], Char[b],* > or < *Integer[a]@Real[a], Char[b],* >
Endfor: Number of Y-Dir. Nodes

For: Number of Z-Dir. Nodes
< Surface Position[a], Units[b] (m), > or < Count[a] @ Node Width[b], Units[c] (m), >
Format: < *Real[a], Char[b],* > or < *Integer[a]@Real[a], Char[b],* >
Endfor: Number of Z-Dir. Nodes

Elseif: Coordinate System Option = { Cylindrical }

For: Number of Radial-Dir. Nodes
< Surface Position[a], Units[b] (m), > or < Count[a] @ Node Width[b], Units[c] (m), >
Format: < *Real[a], Char[b],* > or < *Integer[a]@Real[a], Char[b],* >
Endfor: Number of Radial-Dir. Nodes

For: Number of Azimuthal-Dir. Nodes
< Surface Position[a], Units[b] (deg), > or < Count[a] @ Node Width[b], Units[c] (deg), >
Format: < *Real[a], Char[b],* > or < *Integer[a]@Real[a], Char[b],* >
Endfor: Number of Azimuthal-Dir. Nodes

For: Number of Z-Dir. Nodes
< Surface Position[a], Units[b] (m), > or < Count[a] @ Node Width[b], Units[c] (m), >
Format: < *Real[a], Char[b],* > or < *Integer[a]@Real[a], Char[b],* >
Endfor: Number of Z-Dir. Nodes

Elseif: Coordinate System Option = { Uniform Cartesian }

X-Dir. Node Dimension[a], Units[b] (m)
Format: *Real[a], Char[b],*

Y-Dir. Node Dimension[a], Units[b] (m)
Format: *Real[a], Char[b],*

Z-Dir. Node Dimension[a], Units[b] (m)
Format: *Real[a], Char[b],*

Grid Card (cont'd)

Elseif: Coordinate System Option = { Uniform Cylindrical }

Radial-Dir. Node Dimension[a], Units[b] (m)
Format: *Real[a], Char[b],*

Azimuthal-Dir. Node Dimension[a], Units[b] (deg)
Format: *Real[a], Char[b],*

Z-Dir. Node Dimension[a], Units[b] (m)
Format: *Real[a], Char[b],*

Endif:

Endcard: Grid Card

B.7.1 Grid Card Examples

Extracted from a STOMP1 (Water) input file:

```
#----------------------
~Grid Card
#----------------------
Uniform Cartesian,
20,1,10,
10,cm,
10,cm,
10,cm,
```

Extracted from a STOMP1 (Water) input file:

```
~Grid Card
Cylindrical,
50,1,113,
#Nonuniform grid spacing
0,in,3.125,in,5.125,in,8,in,12,in,18@6,in,10@12,in,10@24,in,8@48,in,
0,deg,45,deg,
0,in,113@6,in,
```

B.8 Hydraulic Properties Card

Card Title[a] { ~Hydraulic [Properties Card] }
Format: *Char*[a]

For: Number of Rock/Soil or Scaling-Group Types

If: Rock/Soil or Scaling Group Name = { IJK | JKI | KIJ } Indexing

Note: The parameter input can be replaced with an external file using the following formatting for ASCII files:

file: *filename*

or the following formattings for binary files:

binary file: *filename*

where; the external file will contain unique parameter values for each node (active or inactive) arranged according to the indexing scheme (i.e., IJK, JKI, or KIJ). Applicable units will be applied to all parameter values in the external file. An example input card is included in section B.8.1

Endif:

Rock/Soil or Scaling-Group Name[a],

If: Rock/Soil or Scaling-Group Name = {{ Fractured }} {{ DP }}

X-Dir. (Radial-Dir.) Matrix Intrinsic Permeability[b], Units[c] (m^2),
or X-Dir. (Radial-Dir.) Matrix Hydraulic Conductivity[b], Units[c] (hc m/s),

Y-Dir. (Azimuthal-Dir.) Matrix Intrinsic Permeability[d], Units[e] (m^2),
or Y-Dir. (Azimuthal-Dir.) Matrix Hydraulic Conductivity[d], Units[e] (hc m/s),

Z-Dir. Matrix Intrinsic Permeability[f], Units[g] (m^2),
or Z-Dir. Matrix Hydraulic Conductivity[f], Units[g] (hc m/s),

X-Dir. (Radial-Dir.) Fracture Intrinsic Permeability[h], Units[i] (m^2),
or X-Dir. (Radial-Dir.) Fracture Hydraulic Conductivity[h], Units[i] (hc m/s),

Y-Dir. (Azimuthal-Dir.) Fracture Intrinsic Permeability[j], Units[l] (m^2),
or Y-Dir. (Azimuthal-Dir.) Fracture Hydraulic Conductivity[j], Units[k] (hc m/s),

Z-Dir. Fracture Intrinsic Permeability[l], Units[m] (m^2),
or Z-Dir. Fracture Hydraulic Conductivity[l], Units[m] (hc m/s),

Hydraulic Properties Card (cont'd)

If: Operational Mode Option: { **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy** }
and NaCl precipitation is considered,
Pore-body Fractional Length[o], Fractional Critical Porosity[p],

Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Char*[e], *Real*[f], *Char*[g], *Real*[h], *Char*[i], *Real*[j], *Char*[k], *Real*[l], *Char*[m], *Real*[h], *Real*[i],

Else:

Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Char*[e], *Real*[f], *Char*[g], *Real*[h], *Char*[i], *Real*[j], *Char*[k], *Real*[l], *Char*[m],

Endif:

Else:

X-Dir. (Radial-Dir.) Intrinsic Permeability[b], Units[c] (m^2),
or X-Dir. (Radial-Dir.) Hydraulic Conductivity[b], Units[c] (hc m/s),

Y-Dir. (Azimuthal-Dir.) Intrinsic Permeability[d], Units[e] (m^2),
or Y-Dir. (Azimuthal-Dir.) Hydraulic Conductivity[d], Units[e] (hc m/s),

Z-Dir. Intrinsic Permeability[f], Units[g] (m^2),
or Z-Dir. Hydraulic Conductivity[f], Units[g] (hc m/s),

If: Operational Mode Option: { **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy** }
and NaCl precipitation is considered,
Pore-body Fractional Length[h], Fractional Critical Porosity[i],
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Char*[e], *Real*[f], *Char*[g], *Real*[h], *Real*[i],
Else:
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Char*[e], *Real*[f], *Char*[g],
Endif:

Endif:

Endfor: Number of Rock/Soil or Scaling-Group Types

Endcard: Hydraulic Properties Card

B.8.1 Hydraulic Properties Card Examples

Extracted from a STOMP1 (Water) input file:
~Hydraulic Properties Card
20/30 Ottawa Sand,1.6e-7,cm^2,1.6e-7,cm^2,1.6e-7,cm^2,

Extracted from a STOMP1 (Water) input file:
~Hydraulic Properties Card
IJK Indexing, file:ksx.dat,hc:cm/s, file:ksy.dat,hc:cm/s, file:ksz.dat,hc:cm/s,

Extracted from a STOMP1 (Water) input file:
~Hydraulic Properties Card
Geologic Media,1.020408e-9,m^2,,,1.020408e-9,m^2,

Extracted from a STOMP1 (Water) input file:
#R1 is a scaling group
~Hydraulic Properties Card
R1,477.09,hc:cm/day,477.09,hc:cm/day,477.09,hc:cm/day,

Extracted from a STOMP1 (Water) input file:
~Hydraulic Properties Card
SP1,4.1987e-12,m^2,,,4.1987e-12,m^2,
SP2,9.3436e-13,m^2,,,9.3436e-13,m^2,
SM-ML1,5.3223e-13,m^2,,,5.3223e-13,m^2,
SM-SP1,7.695e-12,m^2,,,5.13e-12,m^2,
SP3,5.505e-12,m^2,,,3.67e-12,m^2,
SW1,9.195e-12,m^2,,,6.13e-12,m^2,
US,1.0e+04,hc cm/hr,,,1.0e+06,hc cm/hr,

B.9 Inactive Nodes Card

Card Title[a] { ~Inactive [Nodes Card] }
Format: *Char*[a]

Inactive Domain Input Option[a],
{ [Rock | Soil] | Zonation File | File | Integer }

If: Inactive Domain Input Option = { [Rock | Soil] }

Number of Rock/Soil Type Lines[b],
Format: *Char*[a], *Integer*[b],

For: Number of Rock/Soil Type Lines
Inactive Rock/Soil Types[a],
Format: *Char*[a],
Endfor: Number of Rock/Soil Type Lines

Elseif: Inactive Domain Input Option = { Zonation File }

Zonation File Name[b],
Format: *Char*[a], *Char*[b],

Elseif: Inactive Domain Input Option = { File }

Inactive Node File Name[b],
Format: *Char*[a], *Char*[b],

Elseif: Inactive Domain Input Option = { Integer }

Number of Inactive Node Domains[a],
Format: *Integer*[a],

For: Number of Inactive Node Domains
I-Start Index[a], I-End Index[b],
J-Start Index[c], J-End Index[d],
K-Start Index[e], K-End Index[f],
Format: *Integer*[a], *Integer*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f],
Endfor: Number of Inactive Node Domains

Endif:

Endcard: Inactive Nodes Card

B.9.1 Inactive Nodes Card Examples

Extracted from a STOMP1 (Water) input file:

```
~Inactive Nodes Card
4,
1,1,1,1,14,113,
2,2,1,1,14,33,
2,2,1,1,101,113,
2,2,1,1,70,70,
```

Extracted from a STOMP1 (Water) input file:

```
#----------
~Inactive
#----------
1,
2,2,1,1,14,20
```

Extracted from a STOMP1 (Water) input file:

```
#
~Inactive Nodes Card
#
file,inactive.dat,
```

Extracted from a STOMP1 (Water) input file:

```
~Inactive Nodes Card
zonation file, zonation_4,
```

Extracted from a STOMP1 (Water) input file:

```
~Inactive Nodes Card
Rock/Soil Types,2,
fill  material,
engineered structure,
```

B.10 Initial Conditions Card

Card Title[a] { ~Initial [Conditions Card] }
Format: *Char*[a]

If: Operational Mode Option: { **Water** } { **Fluid** }
{ **Water-Air** } { **Water-Air-Energy** } { **Water-Salt** } { **Water-Air-Salt** }
{ **Water-Air-Salt-Energy** } { **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy** }
Initial Saturation Option[a], Initial Saturation Option[b],
{ Gas Pressure, Aqueous Pressure |
Gas Pressure, Aqueous Saturation |
Aqueous Pressure, Aqueous Saturation }
Format: *Char*[a], *Char*[b],
Endif:

Number of Initial Conditions Domains[a]
Format: *Integer*[a],

For: Number of Initial Conditions Domains

Note: The [Overwrite] option is used in conjunction with Restart simulations.

If: Operational Mode Option = { **Water** } { **Fluid** }
Variable Name Option[a],
{ File [Binary] | Rock | Zonation |
Temperature [Overwrite] | Aqueous Pressure [Overwrite] |
Gas Pressure [Overwrite] | Matrix Pressure [Overwrite] |
Aqueous Saturation | Trapped Gas Saturation [Overwrite] |
Solute [Volumetric Conc.] [Overwrite], Solute Name |
Solute Aqueous [Volumetric Conc.] [Overwrite] , Solute Name }

Elseif: Operational Mode Option = { **Water-Air** }
{ Water-Air-Energy }
Variable Name Option[a],
{ File [Binary] | Rock | Zonation |
Temperature [Overwrite] | Aqueous Pressure [Overwrite]
Gas Pressure [Overwrite] |
Aqueous Saturation | Trapped Gas Saturation [Overwrite] |
Aqueous Dissolved Air Mole frac[tion] [Overwrite] |
Aqueous Dissolved Air Mass Fraction [Overwrite] |
Aqueous Dissolved Air Relative Saturation [Overwrite] |
Solute [Volumetric Conc.] [Overwrite], Solute Name |
Solute Aqueous [Volumetric Conc.] [Overwrite], Solute Name |
Solute Gas [Volumetric Conc.] [Overwrite], Solute Name }

Initial Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Oil** }
Variable Name Option[a],
{ File [Binary] | Rock | Zonation |
Temperature [Overwrite] | Aqueous Pressure [Overwrite] |
Gas Pressure [Overwrite] | NAPL Pressure [Overwrite] |
Trapped NAPL Saturation [Overwrite] |
Residual NAPL Saturation [Overwrite] |
Trapped Gas Saturation [Overwrite] |
Aqueous Dissolved Oil Mole frac[tion] [Overwrite] |
Aqueous Dissolved Oil Mass Fraction [Overwrite] |
Aqueous Dissolved Oil Conc [Overwrite] |
Solute [Volumetric Conc.] [Overwrite], Solute Name |
Solute Aqueous [Volumetric Conc.] [Overwrite], Solute Name |
Solute NAPL [Volumetric Conc.] [Overwrite], Solute Name }

Elseif: Operational Mode Option = { **Water-Air-Oil** }
{ Water-Air-Oil-Energy }
Variable Name Option[a],
{ File [Binary] | Rock | Zonation |
Temperature [Overwrite] | Aqueous Pressure [Overwrite] |
Gas Pressure [Overwrite] | NAPL Pressure [Overwrite] |
Trapped NAPL Saturation [Overwrite] | Trapped Gas Saturation [Overwrite] |
Aqueous Dissolved Air Mole frac[tion] [Overwrite] |
Aqueous Dissolved Air Mass Fraction [Overwrite] |
Aqueous Dissolved Air Conc [Overwrite] |
Aqueous Dissolved Oil Mole frac[tion] [Overwrite] |
Aqueous Dissolved Oil Mass Fraction [Overwrite] |
Aqueous Dissolved Oil Conc [Overwrite] |
Solute [Volumetric Conc.] [Overwrite], Solute Name |
Solute Aqueous [Volumetric Conc.], Solute Name |
Solute Gas [Volumetric Conc.] [Overwrite], Solute Name |
Solute NAPL [Volumetric Conc.] [Overwrite], Solute Name }

Elseif: Operational Mode Option = { **Water-Oil-Dissolved Oil** }
Variable Name Option[a],
{ File [Binary] | Rock | Zonation |
Temperature [Overwrite] | Aqueous Pressure [Overwrite] |
Gas Pressure [Overwrite] | NAPL Pressure [Overwrite] |
Trapped NAPL Saturation [Overwrite] | Trapped Gas Saturation [Overwrite] |
Aqueous Dissolved Oil Mass Fraction [Overwrite] |
Aqueous Dissolved Oil Mole frac[tion] [Overwrite] |
Aqueous Dissolved Oil Conc[Overwrite] |
Solute [Volumetric Conc.] [Overwrite], Solute Name |
Solute Aqueous [Volumetric Conc.] [Overwrite], Solute Name |
Solute NAPL [Volumetric Conc.] [Overwrite], Solute Name }

Initial Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-Salt** }
Variable Name Option[a],
{ File [Binary] | Rock | Zonation |

Temperature [Overwrite] | Aqueous Pressure [Overwrite] |
Gas Pressure [Overwrite] |
Aqueous Saturation | Trapped Gas Saturation [Overwrite] |
Salt Mass Fraction [Overwrite] |
Salt Relative Saturation [Overwrite] |
Salt Aqueous [Volumetric Conc.] [Overwrite] |
Solute [Volumetric Conc.] [Overwrite], Solute Name |
Solute Aqueous [Volumetric Conc.] [Overwrite], Solute Name }

Elseif: Operational Mode Option = { **Water-Air-Salt** } { **Water-Air-Salt-Energy** }
Variable Name Option[a],
{ File [Binary] | Rock | Zonation |
Temperature [Overwrite] | Aqueous Pressure [Overwrite] |
Gas Pressure [Overwrite] |
Aqueous Saturation | Trapped Gas Saturation [Overwrite] |
Aqueous Dissolved Air Relative Saturation [Overwrite] |
Aqueous Dissolved Air Mass Fraction [Overwrite] |
Aqueous Dissolved Air Conc [Overwrite] |
Salt Mass Fraction [Overwrite] |
Salt Relative Saturation [Overwrite] |
Salt Aqueous [Volumetric Conc.] [Overwrite] |
Solute [Volumetric Conc.] [Overwrite], Solute Name |
Solute Aqueous [Volumetric Conc.] [Overwrite], Solute Name |
Solute Gas [Volumetric Conc.] [Overwrite], Solute Name }

Initial Conditions Card (cont'd)

Elseif: Operational Mode Option = { **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy** }
Variable Name Option[a],
{ File [Binary] | Rock | Zonation |
Temperature [Overwrite] | Aqueous Pressure [Overwrite] |
Gas Pressure [Overwrite] |
Aqueous Saturation | Trapped Gas Saturation [Overwrite] |
Dissolved CO_2 Partial Pressure [Overwrite] |
Dissolved CO_2 Relative Saturation [Overwrite] |
Dissolved CO_2 Mass Fraction [Overwrite] |
Dissolved CO_2 Conc [Overwrite] |
Salt Mass Fraction [Overwrite] |
Salt Relative Saturation [Overwrite] |
Salt Aqueous [Volumetric Conc.] [Overwrite] |
Salt [Volumetric Conc.] [Overwrite] |
Solute [Volumetric Conc.] [Overwrite], Solute Name |
Solute Aqueous [Volumetric Conc.] [Overwrite], Solute Name |
Solute Gas [Volumetric Conc.] [Overwrite], Solute Name }

Endif:

If: Variable Name Option = { Pressure }
Pressure[b], Units[c] (Pa),
If: Variable Name Option = { File [Binary] }
File Name[d], File Units[e] (Pa),
Note: File contains pressure values for every node.
Format: *Char[a], Real[b], Char[c], Char[d], Char[e],*
Elseif: Variable Name Option = { Zonation | Rock }
Rock/Soil or Scaling Group Name[d],
Format: *Char[a], Real[b], Char[c], Char[d],*
Else:
X-Dir. Gradient[d], Units[e] (1/m),
Y-Dir. Gradient[f], Units[g] (1/m),
Z-Dir. Gradient[h], Units[i] (1/m),
I-Start Index[j], I-End Index[k],
J-Start Index[l], J-End Index[m],
K-Start Index[n], K-End Index[o],
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Real[f], Char[g], Real[h],*
Char[i], Integer[j], Integer[k], Integer[l], Integer[m], Integer[n], Integer[o],
Endif:

Elseif: Variable Name Option = { Temperature }
Temperature[b], Units[c] (C),
If: Variable Name Option = { File [Binary] }
File Name[d], File Units[e] (C),
Note: File contains temperature values for every node.

Initial Conditions Card (cont'd)

Format: *Char[a], Real[b], Char[c], Char[d], Char[e],*
Elseif: Variable Name Option = { Zonation | Rock }
Rock/Soil or Scaling Group Name[d],
Format: *Char[a], Real[b], Char[c], Char[d],*

Else:
X-Dir. Gradient[d], Units[e] (1/m),
Y-Dir. Gradient[f], Units[g] (1/m),
Z-Dir. Gradient[h], Units[i] (1/m),
I-Start Index[j], I-End Index[k],
J-Start Index[l], J-End Index[m],
K-Start Index[n], K-End Index[o],
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Real[f], Char[g], Real[h], Char[i], Integer[j], Integer[k], Integer[l], Integer[m], Integer[n], Integer[o],*
Endif:

Elseif: Variable Name Option = { Saturation }
Saturation[b], Null[c],
If: Variable Name Option = { File [Binary] }
Note: File contains saturation values for every node.
Filename[d],
Format: *Char[a], Real[b], Char[c], Char[d],*
Elseif: Variable Name Option = { Zonation | Rock }
Rock/Soil or Scaling Group Name[d],
Format: *Char[a], Real[b], Char[c], Char[d],*
Else:
X-Dir. Gradient[d], Units[e] (1/m),
Y-Dir. Gradient[f], Units[g] (1/m),
Z-Dir. Gradient[h], Units[i] (1/m),
I-Start Index[j], I-End Index[k],
J-Start Index[l], J-End Index[m],
K-Start Index[n], K-End Index[o],
Format: *Char[a], Real[b],Null[c], Real[d], Char[e], Real[f], Char[g], Real[h], Char[i], Integer[j], Integer[k], Integer[l], Integer[m], Integer[n], Integer[o],*
Endif:

Elseif: Variable Name Option = { Mass Fraction }
Mass Fraction[b], Null[c],
If: Variable Name Option = { File [Binary] }
File Name[d], Null[e] ,
Note: File contains mass fraction values for every node.
Format: *Char[a], Real[b], Char[c], Char[d], Char[e],*
Elseif: Variable Name Option = { Zonation | Rock }
Rock/Soil or Scaling Group Name[d],
Format: *Char[a], Real[b], Char[c], Char[d],*
Else:

Initial Conditions Card (cont'd)

X-Dir. Gradient[d], Units[e] (1/m),
Y-Dir. Gradient[f], Units[g] (1/m),
Z-Dir. Gradient[h], Units[i] (1/m),
I-Start Index[j], I-End Index[k],
J-Start Index[l], J-End Index[m],
K-Start Index[n], K-End Index[o],
Format: *Char[a], Real[b],Null[c], Real[d], Char[e], Real[f], Char[g], Real[h], Char[i], Integer[j], Integer[k], Integer[l], Integer[m], Integer[n], Integer[o],*
Endif:

Elseif: Variable Name Option = { Mole frac[tion] }
Mole frac[tion][b],Null[c],
If: Variable Name Option = { File [Binary] }
File Name[d], Null[e] ,
Note: File contains mole frac[tion] values for every node.
Format: *Char[a], Real[b], Char[c], Char[d], Char[e],*
Elseif: Variable Name Option = { Zonation | Rock }
Rock/Soil or Scaling Group Name[d],
Format: *Char[a], Real[b], Char[c], Char[d],*
Else:
X-Dir. Gradient[d], Units[e] (1/m),
Y-Dir. Gradient[f], Units[g] (1/m),
Z-Dir. Gradient[h], Units[i] (1/m),
I-Start Index[j], I-End Index[k],
J-Start Index[l], J-End Index[m],
K-Start Index[n], K-End Index[o],
Format: *Char[a], Real[b], Null[c], Real[d], Char[e], Real[f], Char[g], Real[h], Char[i], Integer[j], Integer[k], Integer[l], Integer[m], Integer[n], Integer[o],*
Endif:

Elseif: Variable Name Option = { Salt }
Volumetric Conc[b], Units[c] (kg/m^3[3]),
If: Variable Name Option = { File [Binary] }
File Name[d], Units[e] (kg/m^3),
Note: File contains salt Conc values for every node.
Format: *Char[a], Real[b], Char[c], Char[d], Char[e],*
Elseif: Variable Name Option = { Zonation | Rock }
Rock/Soil or Scaling Group Name[d],
Format: *Char[a], Real[b], Char[c], Char[d],*
Else:
X-Dir. Gradient[d], Units[e] (1/m),
Y-Dir. Gradient[f], Units[g] (1/m),
Z-Dir. Gradient[h], Units[i] (1/m),
I-Start Index[j], I-End Index[k],
J-Start Index[l], J-End Index[m],

Initial Conditions Card (cont'd)

K-Start Index[n], K-End Index[o],
Format: *Char[a], Real[b], Null[c], Real[d], Char[e], Real[f], Char[g], Real[h], Char[i], Integer[j], Integer[k], Integer[l], Integer[m], Integer[n], Integer[o],*
Endif:

Elseif: Variable Name Option = { Solute }
Solute Name[b], Volumetric Conc[c], Units[d] (1/m^3),
If: Variable Name Option = { File [Binary] }
File Name[d], Units[e] (1/m^3),
Note: File contains solute Conc values for every node.
Format: *Char[a], Real[b], Char[c], Char[d], Char[e],*
Elseif: Variable Name Option = { Zonation | Rock }
Rock/Soil or Scaling Group Name[d],
Format: *Char[a], Real[b], Char[c], Char[d],*
Else:
X-Dir. Gradient[d], Units[e] (1/m), Y-Dir. Gradient[f], Units[g] (1/m),

Z-Dir. Gradient[h], Units[i] (1/m),
I-Start Index[j], I-End Index[k],
J-Start Index[l], J-End Index[m],
K-Start Index[n], K-End Index[o],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h],*
Real[i], Char[j], Integer[k], Integer[l], Integer[m], Integer[n], Integer[o], Integer[p],
Endif:
Endif:

Endfor: Number of Initial Conditions Domains

Endcard: Initial Conditions Card

B.10.1 Initial Conditions Card Examples

Extracted from a STOMP1 (Water) input file:
~Initial Conditions Card
Gas Pressure,Aaqueous Pressure,
2,
Gas Pressure,101325,Pa,,,,,,1,80,1,1,1,66,
Aqueous Pressure,91534.848,Pa,,,,,-9793.519,1/m,1,80,1,1,1,66,

Extracted from a STOMP1 (Water) input file:
~Initial Conditions Card
Gas Pressure,Aqueous Pressure,
17,
Aqueous Pressure Zonation,9.8614e+4,Pa,bf,
Aqueous Pressure Zonation,9.5260e+4,Pa,ss2,
Aqueous Pressure Zonation,9.7295e+4,Pa,ps2,
Aqueous Pressure Zonation,9.5272e+4,Pa,ss7,
Aqueous Pressure Zonation,9.2080e+4,Pa,ep3,
Aqueous Pressure Zonation,9.8334e+4,Pa,ur,
Aqueous Pressure Zonation,9.7191e+4,Pa,mr,
Aqueous Pressure Zonation,9.5260e+4,Pa,ss1,
Aqueous Pressure Zonation,9.7295e+4,Pa,ps1,
Aqueous Pressure Zonation,9.5260e+4,Pa,ss6,
Aqueous Pressure Zonation,9.2080e+4,Pa,ep1,
Aqueous Pressure Zonation,9.5260e+4,Pa,ss3,
Aqueous Pressure Zonation,9.5260e+4,Pa,ss4,
Aqueous Pressure Zonation,9.2080e+4,Pa,ep2,
Aqueous Pressure Zonation,9.5260e+4,Pa,ss5,
Gas Pressure,102130.86,Pa,,,,,-12.6549,1/m,1,50,1,50,1,50,
Temperature,16.979,C,,,,,-0.065625,1/m,1,50,1,50,1,50,

Extracted from a STOMP2 (Water-Air) input file:
~Initial Conditions Card
Gas Pressure,Aqueous Pressure,
4,
Temperature,20.0,C,,,,,,,1,60,1,1,1,20,
Aqueous Pressure,106173.84,Pa,-1.7554,1/cm,,,-97.9352,1/cm,1,60,1,1,1,20,
Gas Pressure,101331.852,Pa,,,,,-0.11713,1/cm,1,60,1,1,1,20,
Overwrite Solute Aqueous Conc,TCE,200,1/l,,,,,,,28,32,1,1,10,13,

Extracted from a STOMP4 (Water-Oil) input file:

~Initial Conditions Card
3,
Aqueous Pressure,116015.3,Pa,,,,,-9793.5192,1/m,1,1,1,1,1,2,
NAPL Pressure,116015.3,Pa,,,,,-9793.5192,1/m,1,1,1,1,1,2,
Trapped NAPL Saturation,0.15,,,,,,,,1,1,1,1,1,1,

Extracted from a STOMP5 (Water-Air-Oil) input file:

```
#   ------------------------------------------------------------------------
~Initial Conditions Card
#   ------------------------------------------------------------------------
87,
Aqueous Pressure, 1.02855E+06,Pa,,,,,,1, 27,1, 34,  1,  1,
Aqueous Pressure, 9.89357E+05,Pa,,,,,,1, 27,1, 34,  2,  2,
Aqueous Pressure, 9.50162E+05,Pa,,,,,,1, 27,1, 34,  3,  3,
Aqueous Pressure, 9.20767E+05,Pa,,,,,,1, 27,1, 34,  4,  4,
Aqueous Pressure, 9.01169E+05,Pa,,,,,,1, 27,1, 34,  5,  5,
Aqueous Pressure, 8.71772E+05,Pa,,,,,,1, 27,1, 34,  6,  6,
Aqueous Pressure, 8.32578E+05,Pa,,,,,,1, 27,1, 34,  7,  7,
Aqueous Pressure, 7.93383E+05,Pa,,,,,,1, 27,1, 34,  8,  8,
Aqueous Pressure, 7.63986E+05,Pa,,,,,,1, 27,1, 34,  9,  9,
Aqueous Pressure, 7.44389E+05,Pa,,,,,,1, 27,1, 34, 10, 10,
Aqueous Pressure, 7.14994E+05,Pa,,,,,,1, 27,1, 34, 11, 11,
Aqueous Pressure, 6.75799E+05,Pa,,,,,,1, 27,1, 34, 12, 12,
Aqueous Pressure, 6.36604E+05,Pa,,,,,,1, 27,1, 34, 13, 13,
Aqueous Pressure, 5.97410E+05,Pa,,,,,,1, 27,1, 34, 14, 14,
Aqueous Pressure, 5.58215E+05,Pa,,,,,,1, 27,1, 34, 15, 15,
Aqueous Pressure, 5.19020E+05,Pa,,,,,,1, 27,1, 34, 16, 16,
Aqueous Pressure, 4.79825E+05,Pa,,,,,,1, 27,1, 34, 17, 17,
Aqueous Pressure, 4.40631E+05,Pa,,,,,,1, 27,1, 34, 18, 18,
Aqueous Pressure, 4.01436E+05,Pa,,,,,,1, 27,1, 34, 19, 19,
Aqueous Pressure, 3.62241E+05,Pa,,,,,,1, 27,1, 34, 20, 20,
Aqueous Pressure, 3.23046E+05,Pa,,,,,,1, 27,1, 34, 21, 21,
Aqueous Pressure, 2.83852E+05,Pa,,,,,,1, 27,1, 34, 22, 22,
Aqueous Pressure, 2.44657E+05,Pa,,,,,,1, 27,1, 34, 23, 23,
Aqueous Pressure, 2.05462E+05,Pa,,,,,,1, 27,1, 34, 24, 24,
Aqueous Pressure, 1.66268E+05,Pa,,,,,,1, 27,1, 34, 25, 25,
Aqueous Pressure, 1.27075E+05,Pa,,,,,,1, 27,1, 34, 26, 26,
Aqueous Pressure, 8.78947E+04,Pa,,,,,,1, 27,1, 34, 27, 27,
Aqueous Pressure, 4.87094E+04,Pa,,,,,,1, 27,1, 34, 28, 28,
Aqueous Pressure, 8.86231E+03,Pa,,,,,,1, 27,1, 34, 29, 29,
Aqueous Pressure,-2.41632E+04,Pa,,,,,,1, 27,1, 34, 30, 30,
Aqueous Pressure,-5.08930E+04,Pa,,,,,,1, 27,1, 34, 31, 31,
Aqueous Pressure,-8.46251E+04,Pa,,,,,,1, 27,1, 34, 32, 32,
Aqueous Pressure,-1.32961E+05,Pa,,,,,,1, 27,1, 34, 33, 33,
Aqueous Pressure,-2.17807E+05,Pa,,,,,,1, 27,1, 34, 34, 34,
Aqueous Pressure,-3.79198E+05,Pa,,,,,,1, 27,1, 34, 35, 35,
Aqueous Pressure,-5.10390E+05,Pa,,,,,,1, 27,1, 34, 36, 36,
Aqueous Pressure,-6.41645E+05,Pa,,,,,,1, 27,1, 34, 37, 37,
Aqueous Pressure,-7.72952E+05,Pa,,,,,,1, 27,1, 34, 38, 38,
Aqueous Pressure,-10.04302E+05,Pa,,,,,,1, 27,1, 34, 39, 39,
Aqueous Pressure,-1.03569E+06,Pa,,,,,,1, 27,1, 34, 40, 40,
Aqueous Pressure,-1.16711E+06,Pa,,,,,,1, 27,1, 34, 41, 41,
Aqueous Pressure,-1.25673E+06,Pa,,,,,,1, 27,1, 34, 42, 42,
Aqueous Pressure,-1.31351E+06,Pa,,,,,,1, 27,1, 34, 43, 43,
Aqueous Pressure,-2.52169E+06,Pa,,,,,,1, 27,1, 34, 44, 44,
Aqueous Pressure,-4.87873E+06,Pa,,,,,,1, 27,1, 34, 45, 45,
Aqueous Pressure,-7.23587E+06,Pa,,,,,,1, 27,1, 34, 46, 46,
```

Initial Conditions Card Examples (cont'd)

Aqueous Pressure,-9.59311E+06,Pa,,,,,,1, 27,1, 34, 47, 47,
Aqueous Pressure,-1.19504E+07,Pa,,,,,,1, 27,1, 34, 48, 48,
Aqueous Pressure,-1.43078E+07,Pa,,,,,,1, 27,1, 34, 49, 49,
Aqueous Pressure,-1.69601E+07,Pa,,,,,,1, 27,1, 34, 50, 50,
Aqueous Pressure,-1.81358E+07,Pa,,,,,,1, 27,1, 34, 51, 51,
Aqueous Pressure,-1.93114E+07,Pa,,,,,,1, 27,1, 34, 52, 52,
Aqueous Pressure,-2.04870E+07,Pa,,,,,,1, 27,1, 34, 53, 53,
Aqueous Pressure,-2.16626E+07,Pa,,,,,,1, 27,1, 34, 54, 54,
Aqueous Pressure,-2.28382E+07,Pa,,,,,,1, 27,1, 34, 55, 55,
Aqueous Pressure,-2.40138E+07,Pa,,,,,,1, 27,1, 34, 56, 56,
Aqueous Pressure,-2.46660E+07,Pa,,,,,,1, 27,1, 34, 57, 57,
Aqueous Pressure,-2.47949E+07,Pa,,,,,,1, 27,1, 34, 58, 58,
Aqueous Pressure,-2.49238E+07,Pa,,,,,,1, 27,1, 34, 59, 59,
Aqueous Pressure,-2.50526E+07,Pa,,,,,,1, 27,1, 34, 60, 60,
Aqueous Pressure,-2.51814E+07,Pa,,,,,,1, 27,1, 34, 61, 61,
Aqueous Pressure,-2.53102E+07,Pa,,,,,,1, 27,1, 34, 62, 62,
Aqueous Pressure,-2.54390E+07,Pa,,,,,,1, 27,1, 34, 63, 63,
Aqueous Pressure,-2.55678E+07,Pa,,,,,,1, 27,1, 34, 64, 64,
Aqueous Pressure,-2.56966E+07,Pa,,,,,,1, 27,1, 34, 65, 65,
Aqueous Pressure,-2.58254E+07,Pa,,,,,,1, 27,1, 34, 66, 66,
Aqueous Pressure,-2.59541E+07,Pa,,,,,,1, 27,1, 34, 67, 67,
Aqueous Pressure,-2.60828E+07,Pa,,,,,,1, 27,1, 34, 68, 68,
Aqueous Pressure,-2.61590E+07,Pa,,,,,,1, 27,1, 34, 69, 69,
Aqueous Pressure,-2.61826E+07,Pa,,,,,,1, 27,1, 34, 70, 70,
Aqueous Pressure,-2.62062E+07,Pa,,,,,,1, 27,1, 34, 71, 71,
Aqueous Pressure,-2.62297E+07,Pa,,,,,,1, 27,1, 34, 72, 72,
Aqueous Pressure,-2.62533E+07,Pa,,,,,,1, 27,1, 34, 73, 73,
Aqueous Pressure,-2.62769E+07,Pa,,,,,,1, 27,1, 34, 74, 74,
Aqueous Pressure,-2.63005E+07,Pa,,,,,,1, 27,1, 34, 75, 75,
Aqueous Pressure,-2.63195E+07,Pa,,,,,,1, 27,1, 34, 76, 76,
Aqueous Pressure,-2.63338E+07,Pa,,,,,,1, 27,1, 34, 77, 77,
Aqueous Pressure,-2.63482E+07,Pa,,,,,,1, 27,1, 34, 78, 78,
Aqueous Pressure,-2.63626E+07,Pa,,,,,,1, 27,1, 34, 79, 79,
Aqueous Pressure,-2.63770E+07,Pa,,,,,,1, 27,1, 34, 80, 80,
Aqueous Pressure,-2.63914E+07,Pa,,,,,,1, 27,1, 34, 81, 81,
Aqueous Pressure,-2.63914E+07,Pa,,,,,,1, 27,1, 34, 82, 82,
Aqueous Pressure,-2.63914E+07,Pa,,,,,,1, 27,1, 34, 83, 83,
Aqueous Pressure,-2.63914E+07,Pa,,,,,,1, 27,1, 34, 84, 84,
Aqueous Pressure,-2.63914E+07,Pa,,,,,,1, 27,1, 34, 85, 85,
Gas Pressure, 1.03234E+05,Pa,,,,,-11.71,1/m,1, 27,1, 34, 1, 85,
NAPL Pressure,-1.e+09,Pa,,,,,,1, 27,1, 34,1, 85,

Extracted from a STOMP33 (Water-CO2-NaCl-Energy) input file:
~Initial Conditions Card
Gas Pressure,Aqueous Pressure,
3,
Gas Pressure,138.0,Bar,,,,,,1,100,1,1,1,50,
Aqueous Pressure,138.0,Bar,,,,,,1,100,1,1,1,50,
Temperature,25.0,C,,,,,,1,100,1,1,1,50,

B.11 Mechanical Properties Card

Card Title[a] { ~Mechanical [Properties Card] }
Format: *Char*[a]

For: Number of Rock/Soil or Scaling-Group Types

If: Rock/Soil or Scaling Group Name = { IJK | JKI | KIJ } Indexing

Note: Parameter input can be replaced with an external file using the following formatting for ASCII files:

file: *filename*

or the following formattings for binary files:

binary file: *filename*

where; the external file will contain unique parameter values for each node (active or inactive) arranged according to the indexing scheme (i.e., IJK, JKI, or KIJ). Applicable units will be applied to all parameter values in the external file. See the example input card in section B11.1.

Endif:

If: Operational Mode Option = { **Water w/o Transport** } { **Fluid w/o Transport** }

Rock/Soil or Scaling-Group Name[a], Particle Density[b] (2650.0), Units[c] (kg/m^3),
If: Rock/Soil or Scaling-Group Name = {{ Fractured }}
Matrix Total Porosity[d], Matrix Diffusive Porosity[e],
Fracture Total Porosity[f], Fracture Diffusive Porosity[g],
Specific Storativity[h], Units[i] (1/m),
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Real*[e], *Real*[f], *Real*[g], *Real*[h], *Char*[i],
Elseif: Rock/Soil or Scaling-Group Name = {{ DP }}
Matrix Total Porosity[d], Matrix Diffusive Porosity[e],
Fracture Total Porosity[f], Fracture Diffusive Porosity[g],
Matrix Specific Storativity[h], Units[i] (1/m),
Fracture Specific Storativity[j], Units[k] (1/m),
Characteristic Length[l], Units[m] (m),
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Real*[e], *Real*[f], *Real*[g], *Real*[h], *Char*[i],
Real[j], *Char*[k], *Real*[l], *Char*[m],
Else:
Total Porosity[d], Diffusive Porosity[e], Specific Storativity[f], Units[g] (1/m),
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Real*[e], *Real*[f], *Char*[g],
Endif:

Mechanical Properties Card (cont'd)

Elseif: Operational Mode Option = { **Water w/ Transport** } { **Fluid w/ Transport** }
{ Water-Salt }

Rock/Soil or Scaling-Group Name[a], Particle Density[b] (2650.0), Units[c] (kg/m^3),
If: Rock/Soil or Scaling-Group Name = {{ Fractured }}
  Matrix Total Porosity[d], Matrix Diffusive Porosity[e],
  Fracture Total Porosity[f], Fracture Diffusive Porosity[g],
  Specific Storativity[h], Units[i] (1/m),
  Tortuosity Function Option[j],
    { Constant | Millington and Quirk }
  If: Tortuosity Function Option = { Constant }
    Aqueous-Phase Tortuosity[k],
    Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Real[g], Real[h], Char[i], Char[j], Real[k],*
  Else:
    Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Real[g], Real[h], Char[i], Char[j],*
  Endif:
Elseif: Rock/Soil or Scaling-Group Name = {{ DP }}
  Matrix Total Porosity[d], Matrix Diffusive Porosity[e],
  Fracture Total Porosity[f], Fracture Diffusive Porosity[g],
  Matrix Specific Storativity[h], Units[i] (1/m),
  Fracture Specific Storativity[j], Units[k] (1/m),
  Characteristic Length[l], Units[m] (m),
  Tortuosity Function Option[n],
    { Constant | Millington and Quirk }
  If: Tortuosity Function Option = { Constant }
    Aqueous-Phase Tortuosity[o],
    Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Real[g], Real[h], Char[i],*
      Real[j], Char[k], Real[l], Char[m], Char[n], Real[o],
  Else:
    Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Real[g], Real[h], Char[i],*
      Real[j], Char[k], Real[l], Char[m], Char[n],
  Endif:
Else:
  Total Porosity[d], Diffusive Porosity[e], Specific Storativity[f], Units[g] (1/m),
  Tortuosity Function Option[h],
    { Constant | Millington and Quirk }
  If: Tortuosity Function Option = { Constant }
    Aqueous-Phase Tortuosity[i],
    Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Char[g], Char[h], Real[i],*
  Else:
    Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Char[g], Char[h],*
  Endif:
Endif:

Mechanical Properties Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air** } { **Water-Air-Energy** } { **Water-Air-Salt** } { **Water-Air-Salt-Energy** } { **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy }**

Rock/Soil or Scaling-Group Name[a], Particle Density[b] (2650.0), Units[c] (kg/m^3),
If: Rock/Soil or Scaling-Group Name = {{ Fractured }}
Matrix Total Porosity[d], Matrix Diffusive Porosity[e],
Fracture Total Porosity[f], Fracture Diffusive Porosity[g],
Specific Storativity[h], Units[i] (1/m),
Tortuosity Function Option[j],
{ Constant | Millington and Quirk }
If: Tortuosity Function Option = { Constant }
Aqueous-Phase Tortuosity[k], Gas-Phase Tortuosity[l],
Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Real[g], Real[h], Char[i], Char[j], Real[k], Real[l],*
Else:
Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Real[g], Real[h], Char[i], Char[j],*
Endif:
Else:
Total Porosity[d], Diffusive Porosity[e], Specific Storativity[f], Units[g] (1/m),
Tortuosity Function Option[h],
{ Constant | Millington and Quirk }
If: Tortuosity Function Option = { Constant }
Aqueous-Phase Tortuosity[i], Gas-Phase Tortuosity[j],
Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Char[g], Char[h], Real[i], Real[j],*
Else:
Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Char[g], Char[h],*
Endif:
Endif:

Elseif: Operational Mode Option = { **Water-Oil** } { **Water-Oil-Dissolved Oil** }

Rock/Soil or Scaling-Group Name[a], Particle Density[b] (2650.0), Units[c] (kg/m^3),
If: Rock/Soil or Scaling-Group Name = {{ Fractured }}
Matrix Total Porosity[d], Matrix Diffusive Porosity[e],
Fracture Total Porosity[f], Fracture Diffusive Porosity[g],
Specific Storativity[h], Units[i] (1/m),
Tortuosity Function Option[j],
{ Constant | Millington and Quirk }
If: Tortuosity Function Option = { Constant }
Aqueous-Phase Tortuosity[k], NAPL-Phase Tortuosity[l],
Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Real[g], Real[h], Char[i], Char[j], Real[k], Real[l],*
Else:
Format: *Char[a], Real[b], Char[c], Real[d], Real[e], Real[f], Real[g], Real[h], Char[i], Char[j],*

Mechanical Properties Card (cont'd)

Endif:
Else:
Total Porosity[d], Diffusive Porosity[e], Specific Storativity[f], Units[g] (1/m),
Tortuosity Function Option[h],

{ Constant | Millington and Quirk }
If: Tortuosity Function Option = { Constant }
Aqueous-Phase Tortuosity[i], NAPL-Phase Tortuosity[j],
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Real*[e], *Real*[f], *Char*[g], *Char*[h], *Real*[i], *Real*[j],
Else:
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Real*[e], *Real*[f], *Char*[g], *Char*[h],
Endif:
Endif:

Elseif: Operational Mode Option = { **Water-Air-Oil** } { **Water-Air-Oil-Energy** }

Rock/Soil or Scaling-Group Name[a], Particle Density[b] (2650.0), Units[c] (kg/m^3),
If: Rock/Soil or Scaling-Group Name = {{ Fractured }}
Matrix Total Porosity[d], Matrix Diffusive Porosity[e],
Fracture Total Porosity[f], Fracture Diffusive Porosity[g],
Specific Storativity[h], Units[i] (1/m),
Tortuosity Function Option[j],
{ Constant | Millington and Quirk }
If: Tortuosity Function Option = { Constant }
Aqueous-Phase Tortuosity[k], Gas-Phase Tortuosity[l], NAPL-Phase Tortuosity[m],
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Real*[e], *Real*[f],
Real[g], *Real*[h], *Char*[i], *Char*[j], *Real*[k], *Real*[l], *Real*[m],
Else:
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Real*[e], *Real*[f], *Real*[g], *Real*[h], *Char*[i], *Char*[j],
Endif:
Else:
Total Porosity[d], Diffusive Porosity[e], Specific Storativity[f], Units[g] (1/m),
Tortuosity Function Option[h],
{ Constant | Millington and Quirk }
If: Tortuosity Function Option = { Constant }
Aqueous-Phase Tortuosity[i], Gas-Phase Tortuosity[j], NAPL-Phase Tortuosity[k],
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Real*[e], *Real*[f],
Char[g], *Char*[h], *Real*[i], *Real*[j], *Real*[k],
Else:
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Real*[e], *Real*[f], *Char*[g], *Char*[h],
Endif:
Endif:
Endif:

Endfor: Number of Rock/Soil or Scaling Group Types

Mechanical Properties Card (cont'd)

Endcard: Mechanical Properties Card

B.11.1 Mechanical Properties Card Examples

Extracted from a STOMP1 (Water) input file:
~Mechanical Properties Card
IJK Indexing,2690,kg/m^3,file:por.dat,file:por.dat,,1/m,Millington and Quirk,

Extracted from a STOMP1 (Water) input file:
~Mechanical Properties Card
SP1,2.63,g/cm^3,0.19,0.19,,,,
SP2,2.63,g/cm^3,0.24,0.24,,,,
SM-ML1,2.63,g/cm^3,0.35,0.35,,,,
SM-SP1,2.63,g/cm^3,0.37,0.37,,,,
SP3,2.63,g/cm^3,0.27,0.27,,,,
SW1,2.63,g/cm^3,0.28,0.28,,,,
US,2.63,g/cm^3,0.96,0.96,,,,

Extracted from a STOMP1 (Water) input file:
~Mechanical Properties Card
R1 is a scaling group
R1,,,0.422,0.422,,,Millington and Quirk,

Extracted from a STOMP3 (Water-Air-Energy) input file:
~Mechanical Properties Card
Silt Loam-Gravel Admix,2720,kg/m^3,0.456,0.456,0,,Millington and Quirk,
#Asphaltic Concrete,2630,kg/m^3,0.04,0.04,0,,Millington and Quirk,
Gravel Drainage,2725,kg/m^3,0.419,0.419,0,,Millington and Quirk,
Gravel Filter,2725,kg/m^3,0.419,0.419,0,,Millington and Quirk,
#Backfill,2800,kg/m^3,0.307,0.307,0,,Millington and Quirk,
Riprap,2950,kg/m^3,0.2711,0.2711,0,,Millington and Quirk,
Sand Filter,2755,kg/m^3,0.445,0.445,0,,Millington and Quirk,
Compacted Silt Loam,2720,kg/m^3,0.411,0.411,0,,Millington and Quirk,
#Top Course,2800,kg/m^3,0.2585,0.2585,0,,Millington and Quirk,

Extracted from a STOMP33 (Water-$C0_2$-Air-Energy) input file:
~Mechanical Properties Card
30/40 Accusand,2650,kg/m^3,0.327,0.327,,,Millington and Quirk,

B.12 NAPL Relative Permeability Card

Card Title[a] { ~NAPL Rel [ative Permeability Card] }
Format: *Char*[a]

For: Number of Rock/Soil Types

If: Operational Mode = { **Water-Oil** } { **Water-Air-Oil** } { **Water-Air-Oil-Energy** } { **Water-Oil-Dissolved Oil** }
Rock/Soil Name[a],
Permeability Function Option[b],
{ Constant | Mualem | Burdine }
If: Permeability Function Option = { Constant }
NAPL Relative Permeability[c],
Format: *Char*[a], *Char*[b], *Real*[c],
Elseif: Permeability Function Option = { Mualem } { Burdine }
If: Saturation Function Option = { van Genuchten }
van Genuchten m parameter[c],
Format: *Char*[a], *Char*[b], *Real*[c],
Elseif: Saturation Function Option = { Brooks and Corey }
Brooks and Corey parameter[c],
Format: *Char*[a], *Char*[b], *Real*[c],
Endif:
Endif:
Endif:

Endfor: Number of Rock/Soil Types

Endcard: NAPL Relative Permeability Card

B.12.1 NAPL Relative Permeability Card Examples

Extracted from a STOMP4 (Water-Oil) input file:
~NAPL Relative Permeability Card
Sand,Mualem,,

Extracted from a STOMP4 (Water-Oil) input file:
~NAPL Relative Permeability
20/30 Ottawa Sand,Constant,0.58,

Extracted from a STOMP4 (Water-Oil) input file:
~NAPL Relative Permeability Input Card
Sand,Mualem,0.56,

Extracted from a STOMP4 (Water-Oil) input file:
~NAPL Rel
SM-ML1,Burdine,,
Constant NAPL relative permeability
SW1,Constant,1.0,
SP3,Constant,1.0,
SM-SP1,Burdine,,
SP2,Burdine,,
SP1,Burdine,,
US,Mualem,,

Extracted from a STOMP4 (Water-Oil) input file:
~NAPL Relative Permeability Card
pcl,Burdine,,
sm,Burdine,,
sp,Burdine,,
sw,Burdine,,
ml,Burdine,,
acl,Burdine,,
fm,Burdine,,

B.13 Observed Data Card

Card Title[a] { ~Observed Data [Card] }
Format: *Char*[a]

If: Operational Mode Option Card = { **Water** } { **Fluid** }

Number of Observed Data Types[a],
Format: *Integer*[a],
If: Execution Mode Option = { Normal w/ Inverse } { Restart w/ Inverse }

Observed Data Type[a], { Field } { Reference } { Surface Flux } { Surface Rate }
{ Surface Integral }
If: Observed Data Type = { Field }
Field Observation Variable[b],
{ Aqueous Pressure | Aqueous Saturation | Aqueous Moisture Content |
Aqueous Hydraulic Head | X Aqueous Volumetric Flux |
Y Aqueous Volumetric Flux | Z Aqueous Volumetric Flux |
Matric Potential | Solute Volumetric Conc | Solute Aqueous Conc |
Solute Aqueous Mole frac[tion] | X Solute Flux | Y Solute Flux | Z Solute Flux }
If: Field Observation Variable = { Solute Volumetric Conc }
{ Solute Aqueous Conc } { Solute Aqueous Mole frac[tion] }
{ X Solute Flux } { Y Solute Flux } { Z Solute Flux }
Solute Name[c], Field Observation Output Units[d],
Field Observation X-Dir Coordinate[e], Field Observation X-Dir Coordinate Units[f],
Field Observation Y-Dir Coordinate[g], Field Observation X-Dir Coordinate Units[h],
Field Observation Z-Dir Coordinate[i], Field Observation X-Dir Coordinate Units[j]
Observed Data Statistical Index[k], Observed Data Statistic[l],
Observed Data Time Weighting Factor[m], Observed Data Space Weighting Factor[n],
Format: *Char*[a], *Char*[b], *Char*[c],*Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Char*[j], *Integer*[k], *Real*[l], *Real*[m], *Real*[n],
Else:
Field Observation Output Units[c],
Field Observation X-Dir Coordinate[d], Field Observation X-Dir Coordinate Units[e],
Field Observation Y-Dir Coordinate[f], Field Observation X-Dir Coordinate Units[g],
Field Observation Z-Dir Coordinate[h], Field Observation X-Dir Coordinate Units[i],
Observed Data Statistical Index[j], Observed Data Statistic[k],
Observed Data Time Weighting Factor[l], Observed Data Space Weighting Factor[m],
Format: *Char*[a], *Char*[b], *Char*[c], *Real*[d], *Char*[e], *Real*[f], *Char*[g], *Real*[h], *Char*[i], *Integer*[j], *Real*[k], *Real*[l], *Real*[m],
Endif:

Elseif: Observed Data Type = { Reference }
Reference Observation Variable[b],
{ Aqueous Pressure | Aqueous Saturation | Aqueous Moisture Content |
Aqueous Hydraulic Head | X Aqueous Volumetric Flux |

Observed Data Card (cont'd)

Y Aqueous Volumetric Flux | Z Aqueous Volumetric Flux |
Matric Potential | Solute Volumetric Conc | Solute Aqueous Conc |
Solute Aqueous Mole frac[tion] | X Solute Flux | Y Solute Flux | Z Solute Flux }

If: Reference Observation Variable = { Solute Volumetric Conc }
{ Solute Aqueous Conc } { Solute Aqueous Mole frac[tion] }
{ X Solute Flux } { Y Solute Flux } { Z Solute Flux }
Solute Name[c], Reference Observation Output Units[d],
IJK Index 1[e], IJK Index 2[f], IJK Index 3[g],
Observed Data Statistical Index[h], Observed Data Statistic[i],
Observed Data Time Weighting Factor[j], Observed Data Space Weighting Factor[i],
Format: *Char[a], Char[b], Char[c], Integer[d], Integer[e], Integer[f], Integer[s], Real[h], Real[i], Real[j],*
Else:
Reference Observation Output Units[c],
IJK Index 1[c], IJK Index 2[d], IJK Index 3[e],
Observed Data Statistical Index[f], Observed Data Statistic[g],
Observed Data Time Weighting Factor[h], Observed Data Space Weighting Factor[i],
Format: *Char[a], Char[b], Integer[c], Integer[d], Integer[e], Integer[f], Real[g], Real[h], Real[i],*
Endif:

Elseif: Observed Data Type = { Surface Flux | Surface Rate }
Surface Rate Observation Variable[b],
{ Aqueous Volumetric Flux | Aqueous Mass Flux | Solute Flux }
If: Surface Rate Observation Variable = { Solute Flux }
Solute Name[c], Surface Rate Observation Output Units[d],
Surface Rate Observation Orientation[d],
I-Start Domain Index[f], I-End Domain Index[g],
J-Start Domain Index[h], J-End Domain Index[i],
K-Start Domain Index[j], K-End Domain Index[k],
Observed Data Statistical Index[l], Observed Data Statistic[m],
Observed Data Time Weighting Factor[n], Observed Data Space Weighting Factor[o],
Format: *Char[a], Char[b], Char[c], Char[d], Char[e], Integer[f], Integer[s], Integer[h], Integer[i], Integer[j], Integer[k], Integer[l], Real[m], Real[n], Real[o],*
Else:
Surface Rate Observation Output Units[c],
Surface Rate Observation Orientation[d],
I-Start Domain Index[e], I-End Domain Index[f],
J-Start Domain Index[g], J-End Domain Index[h],
K-Start Domain Index[i], K-End Domain Index[j],
Observed Data Statistical Index[k], Observed Data Statistic[l],
Observed Data Time Weighting Factor[m], Observed Data Space Weighting Factor[n],
Format: *Char[a], Char[b], Char[c], Char[d], Integer[e], Integer[f], Integer[s], Integer[h], Integer[i], Integer[j], Integer[k], Real[l], Real[m], Real[n],*
Endif:

Else: Observed Data Type = { Surface Integral }
{ Aqueous Volumetric Flux Integral | Aqueous Mass Flux Integral |
Observed Data Card (cont'd)

Solute Flux Integral }
If: Surface Rate Observation Variable = { Solute Integral }
Solute Name[c], Surface Integral Observation Output Units[d],
Surface Integral Observation Orientation[d],
I-Start Domain Index[f], I-End Domain Index[g],
J-Start Domain Index[h], J-End Domain Index[i],
K-Start Domain Index[j], K-End Domain Index[k],
Observed Data Statistical Index[l], Observed Data Statistic[m],

Observed Data Time Weighting Factor[n], Observed Data Space Weighting Factor[o],
Format: *Char[a], Char[b], Char[c], Char[d], Char[e], Integer[f], Integer[g], Integer[h], Integer[i], Integer[j], Integer[k], Integer[l], Real[m], Real[n], Real[o],*

Else:

Surface Integral Observation Output Units[c],
Surface Integral Observation Orientation[d],
I-Start Domain Index[e], I-End Domain Index[f],
J-Start Domain Index[g], J-End Domain Index[h],
K-Start Domain Index[i], K-End Domain Index[j],
Observed Data Statistical Index[k], Observed Data Statistic[l],
Observed Data Time Weighting Factor[m], Observed Data Space Weighting Factor[n],
Format: *Char[a], Char[b], Char[c], Char[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i], Integer[j], Integer[k], Real[l], Real[m], Real[n],*

Endif:

Endif:

If: Observed Data Read in from External File
File[a] (File), Filename[b], External File Time Units[c], External File Variable Units[d],
Format: *Char[a], Char[b], Char[c], Char[d],*

Elseif: Observed Data Read in from Input file
Number of Observed Data Samples[a],
Format: Integer[a],
Time[a], Time Units[b], Value[c], Value Units[d],
Format: *Real[a], Char[b], Real[c], Char[d],*

Endif:

Endif:

Endif:

Endcard: Observed Data Card

B.13.1 Observed Data Card Examples

Extracted from a STOMP3 (Water-Air-Energy) input file:

```
~Observed Data Card
3,
field,aqueous moisture content,,0.5,cm,0.5,cm,165.5,cm,1,0.01,0.8,0.95,
21,
0,s,0.218,,
484,s,0.213,,
1080,s,0.213,,
1680,s,0.204,,
2880,s,0.204,,
4080,s,0.205,,
5280,s,0.207,,
7080,s,0.204,,
8880,s,0.209,,
13100,s,0.201,,
16700,s,0.201,,
19700,s,0.205,,
68900,s,0.2,,
99300,s,0.196,,
187000,s,0.191,,
427000,s,0.174,,
618000,s,0.164,,
767000,s,0.159,,
1030000,s,0.151,,
1380000,s,0.143,,
1980000,s,0.136,,
field,matric potential,cm,0.5,cm,0.5,cm,60.5,cm,1,4.0,0.8,0.8,
21,
0,s,-1,cm,
484,s,-1,cm,
1080,s,-2,cm,
1680,s,-3,cm,
2880,s,-5,cm,
4080,s,-7,cm,
5280,s,-9,cm,
7080,s,-10,cm,
8880,s,-11,cm,
13100,s,-14,cm,
16700,s,-16,cm,
19700,s,-16,cm,
68900,s,-20,cm,
99300,s,-20,cm,
187000,s,-23,cm,
427000,s,-24,cm,
618000,s,-26,cm,
767000,s,-27,cm,
1030000,s,-28,cm,
1380000,s,-31,cm,
1980000,s,-33,cm,
```

B.14 Oil Properties Card

Card Title[a] { ~Volatile Organic Compound Prop [erties Card] }
{ ~Organic Compound Prop [erties Card] }
{ ~VOC Prop [erties Card] } { ~Oil Prop [erties Card] }
Format: *Char[a]*

Oil Name[a],
Format: *Char[a]*,

Molecular Weight[a], Units[b] (kgmol/kg),
Freezing Point Temperature[c], Units[d] (K),
Normal BOiling Point[e], Units[f] (K),
Critical Temperature[g], Units[h] (K),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h],*

Critical Pressure[a], Units[b] (bar),
Critical Molar Volume[c], Units[d] (cm^3/mole),
Critical Compressibility[e],
Pitzner Acentric Factor[f],
Dipole Moment[g], Units[h] (Debyes),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h],*

Note: $\bar{c}^{o}_{pn} = a + bT + cT^2 + d\,T^3,\ \frac{J}{mol\,K}$

Isobaric Molar Specific Heat Constants a[a],
Isobaric Molar Specific Heat Constants b[b],
Isobaric Molar Specific Heat Constants c[c],
Isobaric Molar Specific Heat Constants d[d],
Format: *Real[a], Real[b], Real[c], Real[d],*

Saturated Vapor Pressure Function Option[a]

{ Equation 1 } **Note:** $\ln\left[\frac{P^o_g}{P^o_c}\right] = \frac{\left(ax+bx^{1.5}+cx^3+dx^6\right)}{(1-x)}\ where,\ x = 1-\frac{T}{T^o_c}$

{ Equation 2 } **Note:** $\ln\left(P^o_g\right) = a-\frac{b}{T}+c\ln(T)+d\left[\frac{P^o_g}{T^2}\right]$

{ Equation 3 } **Note:** $\ln\left(P^o_g\right) = a-\frac{b}{(c+T)}$

{ Constant } **Note:** $P^o_g = \bar{P}^o_g$

Oil Properties Card (cont'd)

If: Saturated Vapor Pressure Function Option = { Equation 1 }
Constant a[b], Constant b[c], Constant c[d], Constant d[e],
Format: *Char[a], Real[b], Real[c], Real[d], Real[e],*
Elseif: Saturated Vapor Pressure Function Option = { Equation 2 }
Constant a[b], Constant b[c], Constant c[d], Constant d[e],
Format: *Char[a], Real[b], Real[c], Real[d], Real[e],*
Elseif: Saturated Vapor Pressure Function Option = { Equation 3 }
Constant a[b], Constant b[c], Constant c[d],
Format: *Char[a], Real[b], Real[c], Real[d],*
Elseif: Saturated Vapor Pressure Function Option = { Constant }
Saturated Oil Vapor Pressure[b], Units[c] (Pa),
Format: *Char[a], Real[b], Char[c],*
Endif:

Liquid Density Function Option[a]

{ HBT Technique } **Note:** [Reid et al. 1987, pp. 55-66]

{ Modified Rackett w/Reference } **Note:** [Reid et al. 1987, pp. 67]

{ Modified Rackett } **Note:** [Reid et al. 1987, pp. 67]

{ Constant } **Note:** $\square_n^o = \square_n^o$

If: Liquid Density Function Option = { HBT Technique }
Pure Component Characteristic Volume[b], Units[c] (L/mol),
HBT Acentric Factor[d],
Format: *Char[a], Real[b], Char[c], Real[d],*
Elseif: Liquid Density Function Option = { Modified Rackett w/ Reference }
Rackett Compressibility Factor[b],
HBT Acentric Factor[c],
Reference Liquid Density[d], Units[e] (kg/m^3),
Reference Temperature[f], Units[g] (C),
Format: *Char[a], Real[b], Real[c], Real[d], Char[e], Real[f], Char[s],*
Elseif: Liquid Density Function Option = { Modified Rackett }
Rackett Compressibility Factor[b],
HBT Acentric Factor[c],
Format: *Char[a],Real[b],Real[c],*
Elseif: Liquid Density Function Option = { Constant }
Reference Liquid Density[b], Units[c] (kg/m^3),
Format: *Char[a],Real[b],Char[c],*
Endif:

Oil Properties Card (cont'd)

Liquid Viscosity Function Option[a]

{ Reference } **Note:** $\mu_n^o = \left(\mu_n^o\right)^{0.2661} + \left(\frac{T - \bar{T}_n^o}{233.0}\right)^{3.758}$ *in cP and K*

{ Equation 1 } **Note:** $\mu_n^o = aT^b$ *in cP and K*

{ Equation 2 } **Note:** $\mu_n^o = \exp\left(a + \frac{b}{T}\right)$ *in cP and K*

{ Equation 3 } **Note:** $\mu_n^o = \exp\left(a + \frac{b}{T} + cT + dT^2\right)$ *in cP and K*

{ Constant } **Note:** $\mu_n^o = \bar{\mu}_n^o$

If: Liquid Viscosity Function Option = { Reference }
Reference Liquid Viscosity[b], Units[c] (Pa s),
Reference Temperature[d], Units[e] (C),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e],*
Elseif: Liquid Viscosity Function Option = { Equation 1 }
Constant a[b], Constant b[c],
Format: *Char[a], Real[b], Real[c],*
Elseif: Liquid Viscosity Function Option = { Equation 2 }
Constant a[b], Constant b[c],
Format: *Char[a], Real[b], Real[c],*
Elseif: Liquid Viscosity Function Option = { Equation 3 }
Constant a[b], Constant b[c],
Constant c[d], Constant d[e],
Format: *Char[a], Real[b], Real[c], Real[d], Real[e],*
Elseif: Liquid Viscosity Function Option = { Constant }
Reference Liquid Viscosity[b], Units[c] (Pa s),
Format: *Char[a], Real[b], Char[c],*
Endif:

Henry's Constant for Aqueous Solubility[a] (1.e+20), Units[b] (Pa),
Format: *Real[a],Char[b],*

Endcard: Oil Properties Card

B.14.1 Oil Properties Card Examples

Extracted from a STOMP4 (Water-Oil) input file:
~Oil Properties Card
PCE,
165.834,kg/kgmol,251.,K,394.4,K,620.2,K,
47.6,bar,289.6,cm^3/mol,0.2758,0.2515,0.0,debyes,
,,,,
Equation 1,-7.36067,1.82732,-3.47735,-1.00033,
Modified Rackett,0.2758,0.2515,
Equation 2,-3.334,946.4,
9.463e+07,Pa,

Extracted from a STOMP4 (Water-Oil) input file:
~Volatile Organic Compound Properties Card
Test NAPL,
170.34,g/mol,-9.55,C,225.35,C,385.05,C,
18.2,bar,713,cm^3/mol,0.24,0.0,0,debyes,
-9.328,1.149,-0.0006347,1.359e-07,
Equation 2,77.628,10012.5,-9.236,10030.0,
Constant,840,kg/m^3,
Constant,0.0047,Pa s,
1.0e10,Pa,

Extracted from a STOMP4 (Water-Oil) input file:
~Oil Properties Card
brooklawn napl, (hexachloro-1,3-butadiene)
260.7602,g/mol,164.2,K,268.7,K,425.0,K,
43.3,bar,221.0,cm^3/mol,0.270,0.195,0.0,debyes,
-1.687e+0,3.419e-1,-2.340e-4,6.335e-8,
Equation 5,0.20892e+2,0.30247e+04,-0.64044e+02,(1,1,2-trichloroethane)
Constant,1.39,g/cm^3,
Constant,32.9,cP,
1.582e10,Pa,(gives a solubility of 2 gm/liter)

Extracted from a STOMP5 (Water-Air-Oil) input file:
```
  #   ------------------------------------------------------------------------
~Volatile Organic Compound Properties Card
#   ------------------------------------------------------------------------
```
Hanford Site Z9 Carbon tetrachloride Mixture,
153.82,g/mol,250.,K,349.9,K,556.4,K,
45.6,bar,275.9,cm^3/mol,0.272,0.193,0.0,debyes,
4.072e+1,2.0496e-1,-2.27e-4,8.843e-8,
Constant,10830,Pa,
Constant,1426,kg/m^3,
Constant,1.11e-3,Pa s,
1.3062e8,Pa,

Oil Properties Card Examples (cont'd)

Extracted from a STOMP5 (Water-Air-Oil) input file:

```
~Oil Properties Card
PCE,
165.834,g/mol,251.,K,394.4,K,620.2,K,
47.6,bar,289.6,cm^3/mol,0.2758,0.2515,0.0,debyes,
-1.431e+1,5.506e-1,-4.513e-4,1.429e-7,
Equation 1,-7.36067,1.82732,-3.47735,-1.00033,
Constant,1.623,g/cm^3,
Constant,0.89e-3,Pa s,
1.13009e+08,Pa,
```

B.15 Output Control Card

Card Title[a] { ~Output [Control Card] }
Format: *Char*[a]

Number of Reference Nodes[a],
Format: *Integer*[a],

For: Number of Reference Nodes
I Index[a], J Index[b], K Index[c],
Endfor: Number of Reference Nodes
Format: *Integer*[a], *Integer*[b], *Integer*[c],

Reference Node Screen Output Frequency[a],
Reference Node Output File Frequency[b],
Output Time Units[c] (s),
Output Length Units[d] (m),
Screen Significant Digits[e],
Output File Significant Digits[f],
Plot File Significant Digits[g]
Format: *Integer*[a], *Integer*[b], *Char*[c], *Char*[d], *Integer*[e], *Integer*[f], *Integer*[g],

Number of Reference Node Variables[a],
Format: *Integer*[a],

For: Number of Reference Node Variables
Reference Node Variable Option[a], Reference Node Variable Units[b],
Format: *Real*[a], *Char*[b],
Endfor: Number of Reference Node Variables

Note: Refer to following pages for Reference Node Variable Options and Units.

Number of Plot File Times[a]
Format: *Integer*[a],

For: Number of Plot File Times
Plot File Output Time[a], Units[b] (s)
Format: *Real*[a], *Char*[b],
Endfor: Number of Plot File Times

Number of Plot File Variables[a]
Format: *Integer*[a],

For: Number of Plot File Variables
Plot File Variable Option[a], Plot File Variable Units[b],
Format: *Char*[a], *Char*[b],
Output Control Card (cont'd)

Endfor: Number of Plot File Variables

Note: Refer to the following pages for Plot File Variable Options and Units.

Endcard: Output Control Card

Reference Node Variable and Plot File Variable Options

If: Operational Mode Option = { **Water** }
{ apparent aqueous saturation | aqueous courant [number] | aqueous density |
aqueous fracture saturation | aqueous gauge pressure | aqueous hydraulic head |
aqueous matrix saturation | aqueous moisture cont[ent] | aqueous pressure |
aqueous relative perm[eability] | aqueous saturation | aqueous viscosity |
aqueous well depth*[p] | diffusive porosity | effective trapped gas saturation | gas gauge pressure |
gas pressure | gas saturation | integrated water mass[*] | matric potential | matrix pressure |
phase condition | rock/soil type | scanning path | solute aqueous conc[entration] |
solute aqueous mole frac[tion] | solute integrated mass | solute integrated aqueous |
solute inventory | solute source int[egral] | solute volumetric conc[entration] |
temperature | total water mass | trapped gas saturation | well flow integral*[p] | well flow rate*[p] |
water aqueous mass fraction | water mass source int[egral] | water mass source rate |
x aqueous relative permeability |
x aqueous vol[umetric flux] | x solute flux | xnc aqueous vol[umetric flux (node centered)] |
y aqueous relative permeability |
y aqueous vol[umetric flux] | y solute flux | ync aqueous vol[umetric flux (node centered)] |
y aqueous relative permeability |
z aqueous vol[umetric flux] | z solute flux | znc aqueous vol[umetric flux (node centered)] }

[*] Reference Node Variable Only
[p] Proprietary

Output Control Card (cont'd)

Reference Node Variable and Plot File Variable Options

Elseif: Operational Mode Option = { **Water-Air** }
{ air aqueous conc[entration] | air aqueous mass frac[tion] | air gas conc[entration] |
air gas mass frac[tion] | air gas mole frac[tion] | air mass source int[egral] |
air mass source rate | air partial pressure |
apparent aqueous saturation | aqueous courant [number] | aqueous density |
aqueous fracture saturation | aqueous gauge pressure | aqueous hydraulic head |
aqueous matrix saturation | aqueous moisture cont[ent] | aqueous pressure |
aqueous relative perm[eability] | aqueous saturation | aqueous viscosity |
aqueous well depth*[p] | axial aquous flux | axial gas flux |
diffusive porosity | dissolved air saturation |
effective trapped gas saturation | gas courant [number] | gas density |
gas fracture saturation | gas gauge pressure | gas hydraulic head |
gas matrix saturation | gas pressure | gas relative perm[eability] |
gas saturation | integrated air mass* | integrated aqueous air [mass]* |
integrated aqueous water [mass]* | { integrated gas air [mass]* |
integrated gas water [mass]* | { integrated trapped gas air [mass]* |
integrated water mass* | phase condition | rock/soil type |
solute aqueous conc[entration] | solute aqueous conc[entration] |
solute aqueous mole frac[tion] |solute gas conc[entration] | solute gas mole frac[tion] |
solute gas conc[entration] | solute source int[egral] | solute volumetric conc[entration] |
temperature | total air mass | total water mass | trapped gas saturation |
vertical aqueous flux | vertical gas flux |
water aqueous conc[entration] | water aqueous mass frac[tion] | water gas conc[entration] |
water gas mass frac[tion] | water gas mole frac[tion] | water source int[egral] |
water mass source int[egral] | water mass source rate \ water vapor partial pres[sure] |
well flow integral*[p] | well flow rate*[p] |
x aqueous vol[umetric flux] | x gas vol[umetric flux] | x solute flux |
xnc aqueous vol[umetric flux (node centered)] | xnc gas vol[umetric flux (node centered)] |
y aqueous vol[umetric flux] | y gas vol[umetric flux] | y solute flux |
ync aqueous vol[umetric flux (node centered)] | ync gas vol[umetric flux (node centered)] |
z aqueous vol[umetric flux] | z gas vol[umetric flux] | z solute flux |
{ znc aqueous vol[umetric flux (node centered)] | znc gas vol[umetric flux (node centered)] }

* Reference Node Variable Only
p Proprietary

Output Control Card (cont'd)

Reference Node Variable and Plot File Variable Options

Elseif: Operational Mode Option = { **Water-Air-Energy** }
{ actual evaporation rate | actual transpiration rate |
air aqueous conc[entration] | air aqueous mass frac[tion] | air gas conc[entration] |
air gas mass frac[tion] | air gas mole frac[tion] | air mass source int[egral] |
air mass source rate | air partial pressure |
apparent aqueous saturation | aqueous courant [number] | aqueous density |
aqueous fracture saturation | aqueous gauge pressure | aqueous hydraulic head |
aqueous matrix saturation | aqueous moisture cont[ent] | aqueous pressure |
aqueous relative perm[eability] | aqueous saturation | aqueous viscosity |
atmospheric relative humidity | atmospheric solar radiation |
atmospheric temperature | atmospheric wind speed |
axial aquous flux | axial gas flux | diffusive porosity |
dissolved air saturation | effective trapped air | energy source int[egral] |
energy source rate | gas courant [number] |
gas density | gas fracture saturation | gas gauge pressure |
gas hydraulic head | gas matrix saturation | gas pressure |
gas relative perm[eability] | gas saturation | integrated air mass* |
integrated aqueous air [mass]* | integrated aqueous water [mass]* |
integrated gas air [mass]* | integrated gas water [mass]* | integrated trapped gas air* |
integrated water mass* | phase condition |
potential evaporation rate | potential transpiration rate |
rock/soil type | solute aqueous conc[entration] | solute aqueous mole fra[ction] |
solute gas conc[entration] | solute gas mole fra[ction] | solute source int[egral] |
solute volumetric conc[entration] | temperature |
surface temperature | surface vapor pressure | total air mass frac[tion] |
total water mass frac[tion] | vertical aqueous flux | vertical gas flux |
water aqueous conc[entration] | water aqueous mass frac[tion] |
water gas conc[entration] | water gas mass frac[tion] | water gas mole frac[tion] |
water mass source int[egral] | water mass source rate | water vapor partial pressure |
x aqueous vol[umetric flux] | x gas vol[umetric flux] | x heat flux | x solute flux |
x thermal cond[uctivity] |xnc aqueous vol[umetric flux (node centered)] |
xnc gas vol[umetric flux (node centered)] | xnc heat flux (node centered) |
y aqueous vol[umetric flux] | y gas vol[umetric flux] | y heat flux |y solute flux |
y thermal cond[uctivity] | ync aqueous vol[umetric flux (node centered)] |
ync gas vol[umetric flux (node centered)] | ync heat flux (node centered) |
z aqueous vol[umetric flux] | z gas vol[umetric flux] | z heat flux |z solute flux |
z thermal cond[uctivity] | znc aqueous vol[umetric flux (node centered)] |
znc gas vol[umetric flux (node centered)] | znc heat flux (node centered) }

* Reference Node Variable Only
p Proprietary

Output Control Card (cont'd)

Reference Node Variable and Plot File Variable Options

Elseif: Operational Mode Option = { **Water-Oil** }
{ apparent aqueous saturation | apparent total saturation |
apparent aqueous saturation | aqueous courant [number] | aqueous density |
aqueous gauge pressure | aqueous hydraulic head | aqueous moisture cont[ent] |
aqueous pressure | aqueous relative perm[eability] | aqueous saturation | asqueous viscosity |
aqueous trapped gas saturation | aqueous well flow integral*p |
aqueous well flow rate*p | diffusive porosity | effective trapped gas saturation |
effective trapped napl saturation | free napl saturation | integrated aqueous water [mass]* |
integrated aqueous oil [mass]* | integrated oil mass* | integrated water mass* |
mobile napl saturation | napl courant [number] | napl density |
napl gauge pressure | napl hydraulic head | napl moisture cont[ent] |
napl pressure | napl relative perm[eability] | napl saturation |
napl trapped gas saturation | napl well flow integral*p | napl well flow rate*p |
oil aqueous conc[entration] | oil aqueous mass frac[tion] |
oil gas conc[entration] | oil gas mass frac[tion] | oil gas mole frac[tion] |
oil mass source int[egral] | oil mass source rate | phase condition |
residual napl saturation | rock/soil type |
solute aqueous conc[entration] | solute aqueous mole frac[tion] | solute napl conc[entration] |
solute napl mole frac[tion] | solute source int[egral] | solute volumetric conc[entration] |
temperature | total moisture cont[ent] | total oil mass |
total saturation | total water mass | total well flow integral*p |
total well flow rate*p | trapped gas saturation |
trapped napl saturation | water aqueous conc[entration] | water aqueous mass frac[tion] |
water gas conc[entration] | water gas mass frac[tion] | water gas mole frac[tion] |
water mass source int[egral] | water mass source rate | well depth*p |
x aqueous vol[umetric flux] | x napl vol[umetric flux] | x solute flux |
xnc aqueous vol[umetric flux (node centered)] | xnc napl vol[umetric flux (node centered)] |
y aqueous vol[umetric flux] | y napl vol[umetric flux] | y solute flux |
ync aqueous vol[umetric flux (node centered)] | ync napl vol[umetric flux (node centered)] |
z aqueous vol[umetric flux] | z napl vol[umetric flux] | z solute flux |
znc aqueous vol[umetric flux (node centered)] | znc napl vol[umetric flux (node centered)] }

Coupled well model variables (proprietary}:
{ aqueous depth | aqueous pumping rate | aqueous pumping rate | aqueous saturation |
oil water mass | napl depth | napl pumping rate | napl pumping rate | napl saturation |
pressure | total-liquid depth | total-liquid pumping rate | total-liquid pumping rate }

* Reference Node Variable Only
p Proprietary

Output Control Card (cont'd)

Reference Node Variable and Plot File Variable Options

Elseif: Operational Mode Option = { **Water-Air-Oil** }
{ air aqueous conc[entration] | air aqueous mass frac[tion] | air gas conc[entration] |
air gas mass frac[tion] | air gas mole frac[tion] | air mass source int[egral] |
air mass source rate | air partial pressure | apparent total saturation |
apparent aqueous saturation | aqueous courant [number] | aqueous density |
aqueous gauge pressure | aqueous hydraulic head | aqueous moisture cont[ent] |
aqueous pressure | aqueous relative perm[eability] | aqueous saturation | aqueous viscosity |
diffusive porosity | effective trapped gas saturation | effective trapped napl saturation |
free napl saturation | gas courant [number] | gas density | gas gauge pressure |
gas hydraulic head | gas pressure | gas relative perm[eability] |
integrated air mass* | integrated aqueous air [mass]* | integrated aqueous oil [mass]* |
integrated aqueous water [mass]* | integrated gas air [mass]* |
integrated gas oil [mass]* | integrated gas water [mass]* | integrated oil mass* |
integrated water mass* | gas saturation | mobile napl saturation |
napl courant [number] | napl density |
napl gauge pressure | napl hydraulic head | napl moisture cont[ent] |
napl pressure | napl relative perm[eability] | napl saturation | napl vapor partial pressure |
oil aqueous conc[entration] | oil aqueous mass frac[tion] | oil gas conc[entration] |
oil gas mass frac[tion] | oil gas mole frac[tion] | oil mass source int[egral] |
oil mass source rate | phase condition | residual napl saturation |
rock/soil type | solute aqueous conc[entration] |
solute aqueous mole frac[tion] | solute gas conc[entration] | solute gas mole frac[tion] |
solute napl conc[entration] | solute napl mole frac[tion] | solute source int[egral] |
solute volumetric conc[entration] | temperature | total air mass |
total moisture cont[ent] | total oil mass | total saturation |
total water mass | trapped gas saturation | trapped napl saturation |
water aqueous conc[entration] | water aqueous mass frac[tion] | water gas conc[entration] |
water gas mass frac[tion] | water gas mole frac[tion] | water mass source int[egral] |
water mass source rate | water vapor partial pressure |
x aqueous vol[umetric flux] | x gas vol[umetric flux] | x napl vol[umetric flux] |
x solute flux | xnc aqueous vol[umetric flux (node centered)] |
xnc gas vol[umetric flux (node centered)] | xnc napl vol[umetric flux (node centered)] |
y aqueous vol[umetric flux] | y gas vol[umetric flux] | y napl vol[umetric flux] |
y solute flux | ync aqueous vol[umetric flux (node centered)] |
ync gas vol[umetric flux (node centered)] | ync napl vol[umetric flux (node centered)] |
z aqueous vol[umetric flux] | z gas vol[umetric flux] | z napl vol[umetric flux] |
z solute flux | znc aqueous vol[umetric flux (node centered)] |
znc gas vol[umetric flux (node centered)] | znc napl vol[umetric flux (node centered)] }

* Reference Node Variable Only

Output Control Card (cont'd)

Reference Node Variable and Plot File Variable Options

Elseif: Operational Mode Option = { **Water-Air-Oil-Energy** }
{ air aqueous conc[entration] | aer aqueous mass frac[tion] | air conc[entration] |
air gas mass frac[tion] | air gas mole frac[tion] |
air mass source int[egral] | air mass source rate | apparent total saturation |
apparent aqueous saturation | aqueous courant [number] | aqueous density |
aqueous gauge pressure | aqueous hydraulic head | aqueous moisture cont[ent] |
aqueous pressure | aqueous relative perm[eability] | aqueous saturation | aqueous viscosity |
diffusive porosity | energy source int[egral] | energy source rate |
effective trapped gas saturation | effective trapped napl saturation |
gas courant [number] | gas density | gas gauge pressure |
gas hydraulic head | gas pressure | gas relative perm[eability] |
gas saturation | integrated air mass* | integrated aqueous air [mass]* |
integrated aqueous oil [mass]* | integrated aqueous water [mass]* | integrated gas air [mass]* |
integrated gas oil [mass]* | integrated gas water [mass]* | integrated oil mass* |
integrated water mass* | napl courant [number] | napl density |
napl gauge pressure | napl hydraulic head | napl moisture cont[ent] |
napl pressure | napl relative perm[eability] | napl saturation |
oil aqueous conc[entration] | oil aqueous mass frac[tion] | oil gas conc[entration] |
oil gas mass frac[tion] | oil gas mole frac[tion] | oil mass source int[egral] |
oil mass source rate | phase condition | rock/soil type | solute aqueous conc[entration] |
solute aqueous mol | solute gas conc[entration] | solute gas mole frac[tion] |
solute napl conc[entration] | solute napl mole frac[tion] | solute source int[egral] |
solute volumetric conc[entration] | temperature | total air mass |
total moisture cont[ent] | total oil mass | total saturation |
total water mass | trapped gas saturation | trapped napl saturation |
water aqueous conc[entration] | water aqueous mass frac[tion] | water gas conc[entration] |
water gas mass frac[tion] | water gas mole frac[tion] |
water mass source int[egral] | water mass source rate |
x aqueous vol[umetric flux] | x gas vol[umetric flux] | x heat flux |
x napl vol[umetric flux] | x solute flux | x thermal cond[uctivity] |
xnc aqueous vol[umetric flux (node centered)] | xnc gas vol[umetric flux (node centered)] |
xnc heat flux [node centered] | xnc napl vol[umetric flux (node centered)] |
y aqueous vol[umetric flux] | y gas vol[umetric flux] | y heat flux |
y napl vol[umetric flux] | y solute flux | y thermal cond[uctivity] |
ync aqueous vol[umetric flux (node centered)] | ync gas vol[umetric flux (node centered)] |
ync heat flux [node centered] } | ync napl vol[umetric flux (node centered)] |
z aqueous vol[umetric flux] | z gas vol[umetric flux] | z heat flux |
z napl vol[umetric flux] | z solute flux | z thermal cond[uctivity] |
znc aqueous vol[umetric flux (node centered)] | znc gas vol[umetric flux (node centered)] |
znc ync heat flux [node centered] | znc napl vol[umetric flux (node centered)] }
Endif:

* Reference Node Variable Only

Output Control Card (cont'd)

Reference Node Variable and Plot File Variable Options

Elseif: Operational Mode Option = { **Water-Oil-Dissolved Oil** }
{ air aqueous conc [entration] | air aqueous mass frac[tion] |
air gas conc [entration] | air gas mass frac[tion] |
air gas mole frac[tion] | apparent total saturation |
apparent aqueous saturation | aqueous courant [number] |
aqueous density | aqueous gauge pressure |
aqueous hydraulic head | aqueous moisture cont |
aqueous pressure | aqueous relative perm[eability] |
aqueous sat[uration] | aqueous viscosity | aqueous trapped gas | aqueous viscosity |
diffusive porosity | effective trapped gas saturation | effective trapped napl saturation |
integrated oil mass* | integrated water mass* | integrated aqueous oil mass* |
integrated aqueous water mass* |
napl courant [number] | napl density | napl gauge pressure |
napl hydraulic head | napl moisture cont[ent] |
napl pressure | napl relative perm [eability] |
napl saturation |oil aqueous conc [entration] |
oil aqueous mass frac[tion] | oil gas conc [entration] |
oil gas mass frac[tion] | oil gas mole frac[tion] |
oil mass source int[egral] | oil mass source rate | phase condition |
rock/soil type | solute aqueous conc [entration] |
solute aqueous mole frac[tion] | solute napl conc [entration] |
solute napl mole frac[tion] | solute source |
solute volumetric conc [entration] | temperature |
total moisture cont[ent] | total oil mass |
total saturation | total water mass |
trapped gas saturation | trapped napl saturation | water aqueous conc [entration] |
water aqueous mass frac[tion] | water gas conc [entration] |
water gas mass frac[tion] | water gas mole frac[tion] |
water mass source int[egral] | water mass source rate |
x aqueous vol [unmetric flux] | x dissolved-oil flux | x napl vol [unmetric flux] |
x solute flux | xnc aqueous vol [unmetric flux] | xnc dissolved-oil flux |
xnc napl vol [unmetric flux] |
y aqueous vol [unmetric flux] | y dissolved-oil flux | y napl vol [unmetric flux] |
y solute flux | ync aqueous vol [unmetric flux] | ync dissolved-oil flux |
ync napl vol [unmetric flux] |
z aqueous vol [unmetric flux] | z dissolved-oil flux | z napl vol [unmetric flux] |
z solute flux | znc aqueous vol [unmetric flux] | znc dissolved-oil flux |
znc napl vol [umetric flux] }

Reference Node Variable Only

Output Control Card (cont'd)

Reference Node Variable and Plot File Variable Options

Elseif: Operational Mode Option = { **Water-Salt** }
{ apparent aqueous saturation | aqueous courant [number] | aqueous density |
aqueous fracture saturation | aqueous gauge pressure | aqueous hydraulic head |
aqueous matrix saturation | aqueous moisture cont[ent] | aqueous pressure |
aqueous relative perm[eability] | aqueous salt mass frac[tion] |
aqueous saturation | aqueous viscosity |
aqueous trapped gas saturation | aqueous well depth*[p] | diffusive porosity |
effective trapped gas saturation | gas-aqueous scaling | gas gauge pressure | gas pressure
gas saturation | integrated water [mass][*] | integrated aqueous water [mass][*] |
phase condition | rock/soil type | osmotic eff | osmotic pressure |
salt aqueous conc[entration] | salt mass source int[egral] | salt mass source rate |
salt volumetric conc[entration] | solute aqueous conc[entration] |
solute gas conc[entration] | solute source | solute volumetric conc[entration] |
temperature | total water [mass] | total salt [mass] | trapped gas saturation |
water mass source int[egral] | water mass source rate |
x aqueous vol[umetric flux] | x salt flux | x solute flux |
xnc aqueous vol[umetric flux (node centered)] | xnc salt flux }
y aqueous vol[umetric flux] | y salt flux | y solute flux |
ync aqueous vol[umetric flux (node centered) | ync salt flux |
z aqueous vol[umetric flux] | z salt flux | z solute flux |
znc aqueous vol[umetric flux (node centered)] | znc salt flux }

[*] Reference Node Variable Only
[p] Proprietary Only

Output Control Card (cont'd)

Reference Node Variable and Plot File Variable Options

Elseif: Operational Mode Option = { **Water-Air-Salt** }
{ air aqueous conc[entration] | air aqueous mass frac[tion] | air gas conc[entration] |
air gas mass frac[tion] | air gas mole frac[tion] | air mass source int[egral] |
air mass source rate | apparent aqueous saturation |
aqueous courant [number] | aqueous density |
aqueous fracture | aqueous gauge pressure | aqueous hydraulic head |
aqueous matrix | aqueous moisture cont | aqueous pressure |
aqueous relative perm[eability] | aqueous salt mass frac[tion] |
aqueous saturation | aqueous viscosity |
diffusive porosity | effective trapped gas saturation | gas courant [number] |
gas density | gas fracture saturation | gas gauge pressure |
gas hydraulic head | gas matrix saturation | gas pressure |
gas relative perm[eability] | gas saturation |
integrated air mass* | integrated aqueous air [mass]*
integrated aqueous water [mass]* | integrated gas air [mass]* |
integrated gas water [mass]* { integrated water mass* |
osmotic eff | osmotic pressure | phase condition | rock/soil type | salt aqueous conc[entration] |
salt conc[entration] | salt mass source int[egral] | salt mass source rate |
solute aqueous conc[entration] | solute aqueous mole frac[tion] |
solute gas conc[entration] | solute gas mole frac[tion] | solute source int[egral] |
solute volumetric conc[entration] | temperature | total air mass | total salt mas |
total water mass | water aqueous conc[entration] | water aqueous mass frac[tion] |
water gas conc[entration] | water gas mass frac[tion] | water gas mole frac[tion] |
water mass source int[egral] | water mass source rate |
x aqueous vol[umetric flux] | x gas vol[umetric flux] | x salt flux | x solute flux |
xnc aqueous vol[umetric flux (node centered)] | xnc gas vol[umetric flux (node centered)] |
xnc heat flux (node centered) | xnc salt flux (node centered) |
y aqueous vol[umetric flux] | y gas vol[umetric flux] | y salt flux | y solute flux |
ync aqueous vol[umetric flux (node centered)] | ync gas vol[umetric flux (node centered)] |
ync heat flux (node centered) |ync salt flux (node centered) |
z aqueous vol[umetric flux] | z gas vol[umetric flux] |z salt flux | z solute flux |
znc aqueous vol[umetric flux (node centered)] | znc gas vol[umetric flux (node centered)] |
znc heat flux (node centered) | znc salt flux (node centered) }

* Reference Node Variable Only

Output Control Card (cont'd)

Reference Node Variable and Plot File Variable Options

Elseif: Operational Mode Option = { **Water-Air-Salt-Energy** }

{ air aqueous conc[entration] | air aqueous mass frac[tion] | air gas conc[entration] |
air gas mass frac[tion] | air gas mole frac[tion] | air mass source int[egral] |
air mass source rate | apparent aqueous saturation |
aqueous courant [number] | aqueous density |
aqueous fracture | aqueous gauge pressure | aqueous hydraulic head |
aqueous matrix | aqueous moisture cont | aqueous pressure |
aqueous relative perm[eability] | aqueous salt mass frac[tion] |
aqueous saturation | aqueous viscosity |
diffusive porosity | effective trapped gas saturation | energy source int[egral]
energy source rate | gas courant [number] |
gas density | gas fracture saturation | gas gauge pressure |
gas hydraulic head | gas matrix saturation | gas pressure |
gas relative perm[eability] | gas saturation |
integrated air mass* | integrated aqueous air [mass]*
integrated aqueous water [mass]* | integrated gas air [mass]* |
integrated gas water [mass]* { integrated water mass* |
osmotic eff | osmotic pressure | phase condition | rock/soil type | salt aqueous conc[entration] |
salt conc[entration] | salt mass source int[egral] | salt mass source rate |
solute aqueous conc[entration] | solute aqueous mole frac[tion] |
solute gas conc[entration] | solute gas mole frac[tion] | solute source int[egral] |
solute volumetric conc[entration] | temperature | total air mass | total salt mas |
total water mass | water aqueous conc[entration] | water aqueous mass frac[tion] |
water gas conc[entration] | water gas mass frac[tion] | water gas mole frac[tion] |
water mass source int[egral] | water mass source rate |
x aqueous vol[umetric flux] | x gas vol[umetric flux] | x heat flux | x salt flux | x solute flux |
x thermal cond[uctivity] | xnc aqueous vol[umetric flux (node centered)] |
xnc gas vol[umetric flux (node centered)] | xnc heat flux (node centered) |
xnc salt flux (node centered) |
y aqueous vol[umetric flux] | y gas vol[umetric flux] | y heat flux | y salt flux | y solute flux |
y thermal cond[uctivity] | ync aqueous vol[umetric flux (node centered)] |
ync gas vol[umetric flux (node centered)] | ync heat flux (node centered) |
ync salt flux (node centered) |
z aqueous vol[umetric flux] | z gas vol[umetric flux] | z heat flux | z salt flux | z solute flux |
z thermal cond[uctivity] | znc aqueous vol[umetric flux (node centered)] |
znc gas vol[umetric flux (node centered)] | znc heat flux (node centered) |
znc salt flux (node centered) }

* Reference Node Variable Only

Output Control Card (cont'd)

Reference Node Variable and Plot File Variable Options

Elseif: Operational Mode Option = { **Water- CO2-NaCl** }
{ apparent aqueous saturation | aqueous courant [number] |
aqueous density | aqueous fracture |
aqueous gauge pressure | aqueous hydraulic head |
aqueous matrix | aqueous moisture cont | aqueous pressure |
aqueous relative perm[eability] | aqueous salt mass frac[tion] |
aqueous saturation | aqueous viscosity |
CO_2 aqueous conc[entration] | CO_2 aqueous mass frac[tion] |
CO_2 aqueous mole frac[tion] | CO_2 gas conc[entration] |
CO_2 gas mass frac[tion] | CO_2 gas mole frac[tion] | CO_2 mass source int[egral] |
CO_2 mass source rate | apparent aqueous saturation | aqueous courant |
diffusive porosity | effective trapped gas saturation |
gas aqueous scaling | gas courant [number] |
gas density | gas fracture saturation | gas gauge pressure |
gas hydraulic head | gas matrix saturation | gas pressure |
gas relative perm[eability] | gas saturation |
integrated CO_2 mass* | integrated aqueous CO_2 [mass]* |
integrated aqueous water [mass]* | integrated gas CO_2 [mass]* |
integrated gas water [mass]* | integrated water mass* |
osmotic eff | osmotic pressure | phase condition |
rock/soil type | salt aqueous conc[entration] | salt aqueous mass fraction |
salt aqueous mole frac[tion] |salt conc[entration] | salt mass source int[egral] |
salt mass source rate | salt saturation |
solute aqueous conc[entration] | solute aqueous mole frac[tion] |
solute gas conc[entration] | solute gas mole frac[tion] | solute source int[egral] |
solute volumetric conc[entration] | temperature | total CO_2 mass | total salt mass |
total water mass | water aqueous conc[entration] | water aqueous mass frac[tion] |
water gas conc[entration] |water gas mass frac[tion] | water gas mole frac[tion] |
water mass source int[egral | water mass source rate |
x aqueous vol[umetric flux] | x gas vol[umetric flux] |
x salt flux | x solute flux | xnc aqueous vol[umetric flux (node centered)] |
xnc gas vol[umetric flux (node centered)] | xnc salt flux (node centered) |
y aqueous vol[umetric flux] | y gas vol[umetric flux] |
y salt flux | y solute flux | ync aqueous vol[umetric flux (node centered)] |
ync gas vol[umetric flux (node centered)] | ync salt flux (node centered) |
z aqueous vol[umetric flux] | z gas vol[umetric flux] |
z salt flux |z solute flux | znc aqueous vol[umetric flux (node centered)] |
znc gas vol[umetric flux (node centered)] | znc salt flux (node centered) }

* Reference Node Variable Only

Reference Node Variable and Plot File Variable Options

Elseif: Operational Mode Option = { **Water- CO2-NaCl-Energy** }
{ apparent aqueous saturation | aqueous courant [number] | aqueous density |
aqueous fracture | aqueous gauge pressure | aqueous hydraulic head |
aqueous matrix | aqueous moisture cont | aqueous pressure |
aqueous relative perm[eability] | aqueous salt mass frac[tion] |
aqueous saturation | aqueous viscosity |
CO_2 aqueous conc[entration] | CO_2 aqueous mass frac[tion] |
CO_2 aqueous mole frac[tion] | CO_2 gas conc[entration] |
CO_2 gas mass frac[tion] | CO_2 gas mole frac[tion] | CO_2 mass source int[egral] |
CO_2 mass source rate | diffusive porosity | effective trapped gas saturation |
energy source int[egral] | energy source rate |
gas courant [number] | gas aqueous scaling |
gas density | gas fracture saturation | gas gauge pressure |
gas hydraulic head | gas matrix saturation | gas pressure |
gas relative perm[eability] | gas saturation |
integrated CO_2 mass* | integrated aqueous CO_2 [mass]* |
integrated aqueous water [mass]* | integrated gas CO_2 [mass]* |
integrated gas water [mass]* | integrated water mass* |
osmotic eff | osmotic pressure | phase condition | saturated CO_2 aqueous mass fraction |
rock/soil type | salt aqueous conc[entration] | salt aqueous mass fraction |
salt aqueous mole frac[tion] |salt conc[entration] | salt mass source int[egral] |
salt mass source rate | salt saturation |
solute aqueous conc[entration] | solute aqueous mole frac[tion] |
solute gas conc[entration] | solute gas mole frac[tion] | solute source int[egral] |
solute volumetric conc[entration] | temperature | total CO_2 mass | total salt mass |
total water mass | water aqueous conc[entration] | water aqueous mass frac[tion] |
Water gas conc[entration] | water gas mass frac[tion] | water gas mole frac[tion] |
water mass source int[egral | water mass source rate |
x aqueous vol[umetric flux] | x gas vol[umetric flux] |
x heat flux | x salt flux | x solute flux |
x thermal cond[uctivity] | xnc aqueous vol[umetric flux (node centered)] |
xnc gas vol[umetric flux (node centered)] | xnc heat flux (node centered) |
xnc salt flux (node centered) |
y aqueous vol[umetric flux] | y gas vol[umetric flux] |
y heat flux | y salt flux | y solute flux |
y thermal cond[uctivity] | ync aqueous vol[umetric flux (node centered)] |
ync gas vol[umetric flux (node centered)] | ync heat flux (node centered) |
ync salt flux (node centered) |
z aqueous vol[umetric flux] | z gas vol[umetric flux] |
z heat flux | z salt flux |z solute flux |
z thermal cond[uctivity] |znc aqueous vol[umetric flux (node centered)] |
znc gas vol[umetric flux (node centered)] | znc heat flux (node centered) |
znc salt flux (node centered) }

* Reference Node Variable Only

Output Control Card (cont'd)

Default Units for the Reference Node and Plot File Variables

Air Conc (kg/m^3)
CO_2 Conc (kg/m^3)
Aqueous Relative Permeability (null)
Aqueous Volumetric Flux (m/s)
Energy Source Integral (J)
Gas Saturation (null)
Gauge Pressure (Pa)
Heat Flux (W/m^2)
Mass Fraction (null)
Mole frac[tion] (null)
NAPL Saturation (null)
Phase Condition (null)
Relative Permeability (null)
Salt Conc. (kg/m^3)
Salt Flux ($kg/m^3 s$)
Solute Mole frac[tion] (null)
Temperature (C)
Oil Conc (kg/m^3)
Water Conc (kg/m^3)
Well Depth (m)
Air Mass Source Integral (kg)
CO_2 Mass Source Integral (kg)
Aqueous Saturation (null)
Density (kg/m^3)
Gas Relative Permeability (null)
Gas Volumetric Flux (m/s)
Head (m),
Mass (kg)
Moisture Content (null)
NAPL Relative Permeability (null)
NAPL Volumetric Flux (m/s)
Pressure (Pa)
Saturation (null)
Salt Aqueous Conc. (kg/m^3)
Solute Conc. ($1/m^3$)
Solute Flux ($1/m^2 s$)
Thermal Conductivity (W/m K)
Oil Mass Source Integral (kg)
Water Mass Source Integral (kg)

B.15.1 Output Control Card Examples

Extracted from STOMP1 (Water) input file:

```
~Output Options Card
5,
57,1,82,
57,1,60,
57,1,40,
57,1,18,
57,1,9,
2,1,yr,m,6,6,6,
27,
aqueous saturation,,
aqueous pressure,Pa,
aqueous relative perm,,
aqueous moisture content,,
x aqueous vol,mm/yr,
z aqueous vol,mm/yr,
solute integrated mass,U:0.01,,
solute aqueous conc,  U:0.01, 1/L,
solute volumetric conc,U:0.01, 1/L,
solute integrated mass,U:0.03,,
solute aqueous conc,  U:0.03, 1/L,
solute volumetric conc,U:0.03, 1/L,
solute integrated mass,U:0.10,,
solute aqueous conc,  U:0.10, 1/L,
solute volumetric conc,U:0.10, 1/L,
solute integrated mass,U:0.30,,
solute aqueous conc,  U:0.30, 1/L,
solute volumetric conc,U:0.30, 1/L,
solute integrated mass,U:0.60,,
solute aqueous conc,  U:0.60, 1/L,
solute volumetric conc,U:0.60, 1/L,
solute integrated mass,U:1.00,,
solute aqueous conc,  U:1.00, 1/L,
solute volumetric conc,U:1.00, 1/L,
solute integrated mass,Tc,,
solute aqueous conc,  Tc,   1/L,
solute volumetric conc,Tc,   1/L,
5,
2000,yr,
2000.0383299,yr,
2001,yr,
2010,yr,
2050,yr,
21,
aqueous saturation,,
aqueous pressure,Pa,
aqueous relative perm,,
aqueous moisture content,,
rock/Soil type,,
```

Output Card Examples (cont'd)

xnc aqueous vol,mm/yr,
znc aqueous vol,mm/yr,
solute aqueous conc, U:0.01, 1/L,
solute volumetric conc,U:0.01, 1/L,
solute aqueous conc, U:0.03, 1/L,
solute volumetric conc,U:0.03, 1/L,
solute aqueous conc, U:0.10, 1/L,
solute volumetric conc,U:0.10, 1/L,
solute aqueous conc, U:0.30, 1/L,
solute volumetric conc,U:0.30, 1/L,
solute aqueous conc, U:0.60, 1/L,
solute volumetric conc,U:0.60, 1/L,
solute aqueous conc, U:1.00, 1/L,
solute volumetric conc,U:1.00, 1/L,
solute aqueous conc, Tc, 1/L,
solute volumetric conc,Tc, 1/L,

Extracted from STOMP3 (Water-Air-Energy) input file:
~Output Options Card
1,
1,1,10,
#1,1,9,
#1,1,8,
#1,1,7,
#1,1,6,
#1,1,1,
1,1,day,m,6,6,6,
12,
Aqueous Saturation,,
Temperature,C,
Surface Temperature,C,
Surface Vapor Pressure,Pa,
Atmospheric Temperature,C,
Atmospheric Solar Radiation,W/m^2,
Atmospheric Wind Speed,m/s,
Atmospheric Relative Humidity,,
Potential Evaporation,gm/s,
Actual Evaporation,gm/s,
Potential Transpiration,gm/s,
Actual Transpiration,gm/s,
0,
6,
Aqueous Saturation,,
Aqueous Pressure,Pa,
Gas Pressure,Pa,
Temperature,C,
Surface Temperature,C,
Surface Vapor Pressure,Pa,

Output Card Examples (cont'd)

Extracted from STOMP5 (Water-Air-Oil) input file:

```
~Output Options Card
3,
1,1,42,
1,1,62,
1,1,92,
10,10,hr,cm,6,6,6,
4,
aqueous saturation,,
apparent Water saturation,,
NAPL saturation,,
Oil aqueous conc,mg/L,
3,
0.025,hr,
0.075,hr,
0.125,hr,
6,
aqueous saturation,,
apparent Water saturation,,
NAPL saturation,,
Oil aqueous conc,mg/L,
trapped Oil saturation,,
total trapping number,,
```

Extracted from STOMP11 (Water-Salt) input file:

```
~Output Options Card
6,
30,1,120,
30,1,119,
30,1,118,
60,1,120,
60,1,119,
60,1,118,
1,1,min,m,6,6,6,
6,
Aqueous Saturation,,
Aqueous Moisture Content,,
Aqueous Pressure,,
Z Aqueous Vol,cm/hr,
Salt Aqueous Concentration,kg/m^3,
Aqueous Density,kg/m^3,
5,
0.,min,
5.,min,
10.,min,
20.,min,
30.,min,
0,
```

Output Card Examples (cont'd)

Extracted from STOMP33 (Water-CO2-NaCl-Energy) input file:

```
~Output Options Card
4,
3,1,3,
3,1,4,
4,1,3,
98,1,3,
1,1,hr,cm,6,6,6,
5,
Gas Saturation,,
Temperature,C,
Salt Aqueous Mass Fraction,,
CO2 Aqueous Mass Fraction,,
Gas Pressure,Bar,
7,
0.5,min,
1.0,min,
2.0,min,
5.0,min,
10.0,min,
20.0,min,
50.0,min,
6,
Gas Saturation,,
Temperature,C,
Salt Aqueous Mass Fraction,,
CO2 Aqueous Mass Fraction,,
Gas Pressure,Bar,
Aqueous Density,kg/m^3,
```

B.16 Plant Card

Card Title[a] { ~Plant [Card] }
Format: *Char*[a]

If: Operational Mode Option Card = { **Water-Air-Energy** }

Number of Plant Varietals[a],
Format: *Integer*[a],

For: Number of Plant Varietals
Plant Name[a], [Plant Stress Option[b]], { Stress }
Format: *Char*[a], *Char*[b],
Max. Root Depth[a], Units[b] (m),
Format: *Real*[a], *Char*[b],
Null Root Depth[a], Units[b] (m),
Format: *Real*[a], *Char*[b],
Root Depth Fit Parameter[a],
Format: *Real*[a],
Plant Canopy Height[a], Units (m)[b],
Format: *Real*[a], *Char*[b],
If: Plant Stress Option = { Stress }
Water Stress Point 1[a], Units[b] (m), Water Stress Point 2[c], Units[d] (m),
Water Stress Point 3[e], Units[f] (m), Water Stress Point 4[g], Units[h] (m),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h],
Else:
Root Uptake Reduced 50%[a], Units[b] (m),
Format: *Real*[a], *Char*[b],
Endif:
Crop Coefficient Start[a], Crop Coefficient Day of Year (Start)[b], Units (s)[c],
Crop Coefficient (Mature Stage 1)[d], Crop Coefficient Day of Year (Mature Stage 1)[e],
Units (s)[f], Crop Coefficient (Mature Stage 2)[g],
Crop Coefficient Day of Year (Mature Stage 2)[h], Units (s)[i],
Crop Coefficient (Die-off)[j], Crop Coefficient Day of Year (Die-off)[k], Units (s)[l],
Format: *Real*[a], *Real*[b], *Char*[c], *Real*[d], *Real*[e], *Char*[f], *Real*[g], *Real*[h], *Char*[i], *Real*[j], *Real*[k], *Char*[l],
Endfor: Number of Plant Varietals

Endif:

Endcard: Plant Card

B.16.1 Plant Card Examples

Extracted from STOMP3 (Water-Air-Energy) input file:

```
~Plant Card
2,
sage brush,stress,
1.5,m,
0.35,m,
2.57,
1.0,m,
0.0,m,0.1,m,10.0,m,70.,m,
0.3,0,day,1.0,90,day,1.0,260,day,0.3,365,day,
cheat grass,stressed,
0.6,m,
0.35,m,
2.57,
0.1,m,
0.0,m,0.1,m,0.5,m,15.0,m,
0.2,0,day,0.9,60,day,0.9,180,day,0.2,210,day,
```

B.17 Rock/Soil Zonation Card

Card Title[a] { ~Rock/Soil [Zonation Card] }
Format: *Char*[a]

Rock/Soil Zonation Input Option[a],
{ [Formatted | Unformatted] Zonation File | [IJK | JKI | KIJ] Indexing | Integer }

If: Rock/Soil Zonation Input Option = { [Formatted | Unformatted] File }
Rock/Soil Zonation File Name[b],
Format: *Char*[a], *Char*[b],
For: Number of Rock/Soil Zonation Domains (defined in external file)
Rock/Soil or Scaling Group Name[a],
Format: *Char*[a],
Endfor: Number of Rock/Soil Zonation Domains

Elseif: Rock/Soil Zonation Input Option = { Indexing }

Note: Each node is assigned a different rock/Soil zonation index according to the indexing scheme chosen (i.e., IJK, JKI, or KIJ). This option is useful for stochastic realizations.

Elseif: Rock/Soil Zonation Input Option = { Integer }

For: Number of Rock/Soil Zonation Domains
Rock/Soil or Scaling Group Name[a],
I-Start Index[b], I-End Index[c],
J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g],
If: Operation Mode Option = { **Water** } { **Fluid** } and
Execution Mode Option = { **Normal w/ Scaling** | **Restart w/ Scaling** }
Scaling Group[h],
Format: *Char*[a], *Integer*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f], *Integer*[g], *Char*[h],
Else:
Format: *Char*[a], *Integer*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f], *Integer*[g],
Elseif:
Endfor: Number of Rock/Soil Zonation Domains

Endif:

Endcard: Rock/Soil Zonation Card

B.17.1 Rock/Soil Zonation Card Examples

Extracted from a STOMP1 (Water) input file:
~Rock/Soil Zonation Card
1,
20/30 Ottawa Sand,1,1,1,1,1,92,

Extracted from a STOMP1 (Water) input file:
~Rock/Soil Zonation Card
IJK Indexing,

Extracted from a STOMP1 (Water) input file:
~Rock/Soil Zonation Card
7,
SP1,1,50,1,1,86,113,
SP2,1,50,1,1,63,85,
SM-ML1,1,50,1,1,69,69,
SM-SP1,1,50,1,1,46,62,
SP3,1,50,1,1,16,45,
SW1,1,50,1,1,1,15,
US,2,2,1,1,71,100,

Extracted from a STOMP1 (Water) input file:
~Rock/Soil Zonation Card
formatted file, zonation_a,
backfill,
hanford sand,
plio-pleistocene,
upper ringold,
middle ringold,

Extracted from a STOMP1 (Water) input file:
~Rock/Soil Zonation Card
5,
L5,1,1,1,1,1,40,R1,
L4,1,1,1,1,41,80,R1,
L3,1,1,1,1,81,100,R1,
L2,1,1,1,1,101,105,R1,
L1,1,1,1,1,106,130,R1,

B.18 Salt Transport Card

Card Title[a] { ~Salt Transport [Card] }
Format: *Char*[a]

If: Operational Mode Option Card = { **Water-Salt** } { **Water-Air-Salt** }
{ **Water-Air-Salt-Energy** }

Salt Name[a],
{ Sodium Chloride [Pitzer] | Sodium Nitrate [Pitzer] |
Sodium Thiosulfate [Pitzer] | Sodium Chloride Henry [Pitzer] |
Sodium Chloride Elder [Pitzer] }
Format: Char[a],
If: Salt Name = { Sodium Chloride Pitzer } { Sodium Nitrate Pitzer }
{ Sodium Thiosulfate Pitzer } { Sodium Chloride Henry Pitzer }
{ Sodium Chloride Elder Pitzer }
Number of Anions[a], Number of Cations[b],
Number of Positive Charges[c], Number of Negative Charges[d],
Pitzer Pairwise Ion-Interaction Parameter B0[e],
Pitzer Pairwise Ion-Interaction Parameter B1[f],
Pitzer Pairwise Ion-Interaction Parameter B2[h],
Pitzer Triplet Ion-Interaction Parameter[i],
Surface to Bulk Molality Ratio[j],
Format: *Integer*[a], *Integer*[b], *Integer*[c], *Integer*[d], *Real*[e], *Real*[f], *Real*[g], *Real*[h], *Real*[i],
Endif:

Effective Diffusion Option[a],

{ Constant } **Note:** $D^{s}_{\ell\, e} = D^{s}_{\ell}$

{ Conventional } **Note:** $D^{s}_{\ell\, e} = \square_{\ell}\, s_{\ell}\, n_{D}\, D^{s}_{\ell}$

{ Empirical [Kemper and van Schaik] } **Note:** $D^{s}_{\ell\, e} = D^{s}_{\ell}\, a \exp\left(b n_{D} s_{\ell}\right)$

If: Effective Diffusion Option = { Constant }
Aqueous-Phase Diffusion Coefficient @ 20 C[b], Units[c] (m^2/s),
Format: *Char*[a], *Real*[b], *Char*[c],
Elseif: Effective Diffusion Option = { Conventional }
Aqueous-Phase Diffusion Coefficient @ 20 C[b], Units[c] (m^2/s),
Format: *Char*[a], *Real*[b], *Char*[c],
Elseif: Effective Diffusion Option = { Empirical [Kemper and van Schaik] }
Format: *Char*[a],
Endif:

Salt Transport Card (cont'd)

For: Number of Rock/Soil Types
If: Effective Diffusion Option = { Constant } { Conventional }
Rock/Soil Name[a],

If: Rock/Soil Name = { IJK | JKI | KIJ } Indexing

Note: A parameter value input for the next input line can be replaced with an external file using the following formatting for ASCII files:

file: *filename*

or the following formattings for binary files:

binary file: *filename*

where; the external file will contain unique parameter values for each node (active or inactive) arranged according to the indexing scheme (i.e., IJK, JKI, or KIJ). Applicable units will be applied to all parameter values in the external file.

Endif:

Longitudinal Dispersivity[b], Units[c] (m),
Transverse Dispersivity[d], Units[e] (m),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e],*

Elseif: Effective Diffusion Option = { Empirical }
Rock/Soil Name[a],

If: Rock/Soil Name = { IJK | JKI | KIJ } Indexing

Note: A parameter value input for the next input line can be replaced with an external file using the following formatting for ASCII files:

file: *filename*

or the following formattings for binary files:

binary file: *filename*

where; the external file will contain unique parameter values for each node (active or inactive) arranged according to the indexing scheme (i.e., IJK, JKI, or KIJ). Applicable units will be applied to all parameter values in the external file.

Endif:

Salt Transport Card (cont'd)

Longitudinal Dispersivity[b], Units[c] (m),
Transverse Dispersivity[d], Units[e] (m),
Aqueous Molecular Diffusion Coefficient[f], Units[g] (m^2/s),
Constant a[h],
Constant b[i],
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Real[f], Char[g], Real[h], Real[i],*
Endif:
Endfor: Number of Rock/Soil Types
Endif:

If: Operational Mode Option Card = { **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy** }

For: Number of Rock/Soil Types
Rock/Soil Name[a],

If: Rock/Soil Name = { IJK | JKI | KIJ } Indexing

Note: A parameter value input can be replaced with an external file using the following formatting for ASCII files:

file: *filename*

or the following formattings for binary files:

binary file: *filename*

where; the external file will contain unique parameter values for each node (active or inactive) arranged according to the indexing scheme (i.e., IJK, JKI, or KIJ). Applicable units will be applied to all parameter values in the external file.

Endif:

Longitudinal Dispersivity[b], Units[c] (m),
Transverse Dispersivity[d], Units[e] (m),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e],*
Endfor: Number of Rock/Soil Types
Endif:

Endcard: Salt Transport Card

B.18.1 Salt Transport Card Examples

Extracted from a STOMP11 (Water-Salt) input file:

~Salt Transport Card
Sodium Nitrate,
Constant Diffusion,3.565e-6,m^2/s,
30/40 sand,0.0,m,0.0,m,

Extracted from a STOMP32 (Water-CO2-NaCl) input file

~Salt Transport Card
Eau Claire Carbonate,20.0,ft,5.0,ft,
Eau Claire Shale,20.0,ft,5.0,ft,
Lower Eau Claire,20.0,ft,5.0,ft,
Upper Mt. Simon,20.0,ft,5.0,ft,
Middle Mt. Simon,20.0,ft,5.0,ft,
Lower Mt. Simon,20.0,ft,5.0,ft,

B.19 Saturation Function Card

Card Title[a] { ~Saturation Function [Card] }
Format: *Char*[a]

If: Operational Mode Option = { **Water-Oil** } { **Water-Air-Oil** }
{ **Water-Air-Oil-Energy** } { **Water-Oil-Dissolved Oil** }

Fluid Pair Interfacial Tension

Note: Enter minimum of two fluid pair interfacial tension values. For nonspreading NAPLs (spreading coefficient < 0) enter all 3 values when Operational Mode Option = { **Water-Oil** } { **Water-Air-Oil** }

{ Gas-Aqueous Interfacial Tension[a], Units[b] (N/m),
NAPL-Aqueous Interfacial Tension[c], Units[d] (N/m),
Null[e], Null[f], }

{ Null[a], Null[b],
NAPL-Aqueous Interfacial Tension[c], Units[d] (N/m),
Gas-NAPL Interfacial Tension[e], Units[f] (N/m), }

{ Gas-Aqueous Interfacial Tension[a], Units[b] (N/m),
Null[c], Null[d],
Gas-NAPL Interfacial Tension[e], Units[f] (N/m), }

Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f],

Endif:

For: Number of Rock/Soil or Scaling-Group Types

If: Rock/Soil Name = { IJK | JKI | KIJ } Indexing

Note: The parameter input can be replaced with an external file using the following formatting for ASCII files:

file: *filename*

or the following formattings for binary files:

binary file: *filename*

where; the external file will contain unique parameter values for each node (active or inactive) arranged according to the indexing scheme (i.e., IJK, JKI, or KIJ).

Applicable units will be applied to all parameter values in the external file. An example input card is included in section B.19.1

Endif:

For: Number of Rock/Soil Types or Scaling Groups

If: Operational Mode Option = { **Water** } { **Fluid** } { **Water-Salt** } { **Water-Air-Salt** } { **Water-Air-Salt-Energy** }

Rock/Soil or Scaling Group Name[a],
Saturation Function[b],
{ [Entrapment | Extended | Fractured | Triple] van Genuchten |
[Entrapment | Extended | Fractured | Triple] Brooks and Corey |
Haverkamp | Tabular [Linear | Spline [Log]] [Water Content | Saturation] }

If: Saturation Function Option = { [Extended] van Genuchten }
α Parameter[c], Units[d] (1/m), n Parameter[e], Minimum Saturation[f],
m Parameter[g],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g],*

Elseif: Saturation Function Option = { Entrapment van Genuchten }
α Parameter[c], Units[d] (1/m), n Parameter[e], Minimum Saturation[f],
m Parameter[g], Effective Gas Residual Saturation[h],
Critical Trapping Number[i],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Real[h], Real[I],*

Elseif: Saturation Function Option = { Fractured van Genuchten }
Matrix α Parameter[c], Units[d] (1/m), Matrix n Parameter[e],
Matrix Minimum Saturation[f],
Fracture α Parameter[g], Units[h] (1/m), Fracture n Parameter[i],
Fracture Minimum Saturation[j],
Matrix m Parameter[k], Fracture m Parameter[l],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j], Real[k], Real[l],*

Elseif: Saturation Function Option = { Triple van Genuchten }
Main Drainage α Parameter[c], Units[d] (1/m),
Main Drainage n Parameter[e], Main Drainage Minimum Saturation[f],
Main Drainage m Parameter[g],
Boundary Wetting α Parameter[h], Units[i] (1/m),
Main Wetting α Parameter[j], Units[k] (1/m),
Main Wetting n Parameter[l], Main Wetting Minimum Saturation[m],
Main Wetting m Parameter[n],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Real[h], Real[i],*

Saturation Function Card (cont'd)

Real[j], Char[k], Real[l], Real[m], Real[n],

Elseif: Saturation Function Option = { [Extended] Brooks and Corey }
Entry Head[c], Units[d] (m), λ Parameter[e], Minimum Saturation[f],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f],*

Elseif: Saturation Function Option = { Entrapment Brooks and Corey }
Entry Head[c], Units[d] (m), λ Parameter[e], Minimum Saturation[f],
Effective Gas Residual Saturation[g],
Critical Trapping Number[h],

Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Real[h],*

Elseif: Saturation Function Option = { Fractured Brooks and Corey }
Matrix Entry Head[c], Units[d] (m), Matrix □ Parameter[e],
Matrix Minimum Saturation[f],
Fracture Entry Head[g], Units[h] (m), Fracture □ Parameter[i],
Fracture Minimum Saturation[j],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h],Real[i], Real[j],*

Elseif: Saturation Function Option = { Triple Brooks and Corey }
Main Entry Head[c], Units[d] (m),
Main Drainage □ Parameter[e], Main Drainage Minimum Saturation[f],
Boundary Wetting Entry Head[g], Units[h] (m),
Main Wetting Entry Head[i], Units[j] (m),
Main Wetting □ Parameter[k], Main Wetting Minimum Saturation[l],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Char[j], Real[k], Real[l],*

Elseif: Saturation Function Option = { Haverkamp }
Entry Head Parameter[c], Units[d] (m), □ Parameter[e], □ Parameter[f], Minimum Saturation[g],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*

Elseif: Saturation Function Option = { Tabular [Saturation] [Linear | Spline [Log]] }
Number of Table Entries[a],
Format: Char[a], Char[b], Integer[a]
For: Number of Table Entries
Air-Water Capillary Head[a], Units[b] (m), Aqueous Saturation[c],
Format: *Real[a], Char[b], Real[c],*
Endfor: Number of Table Entries

Elseif: Saturation Function Option = { Tabular Water Content [Linear | Spline [Log]] }
Number of Table Entries[a],
Format: Char[a], Char[b], Integer[a]
For: Number of Table Entries
Air-Water Capillary Head[a], Units[b] (m), Water Content[c],
Format: *Real[a], Char[b], Real[c],*

Saturation Function Card (cont'd)

Endfor: Number of Table Entries

Endif:

Elseif: Operational Mode Option: { **Water-Air** } { **Water-CO2-NaCl** }

Rock/Soil Name[a],
Saturation Function[b],
{ [Entrapment | Extended | Fractured] van Genuchten |
[Entrapment | Extended | Fractured] Brooks and Corey |
Haverkamp | Tabular [Linear | Spline [Log]] [Water Content | Saturation] }

If: Saturation Function Option = { [Extended] van Genuchten }
□ Parameter[c], Units[d] (1/m), n Parameter[e], Minimum Saturation[f],
m Parameter[g],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g],*

Elseif: Saturation Function Option = { Entrapment van Genuchten }
 Parameter[c], Units[d] (1/m), n Parameter[e], Minimum Saturation[f],
m Parameter[g], Effective Gas Residual Saturation[h],
Critical Trapping Number[i],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Real[h], Real[I],*

Elseif: Saturation Function Option = { Fractured van Genuchten }
Matrix Parameter[c], Units[d] (1/m), Matrix n Parameter[e],
Matrix Minimum Saturation[f],
Fracture Parameter[g], Units[h] (1/m), Fracture n Parameter[i],
Fracture Minimum Saturation[j],
Matrix m Parameter[k], Fracture m Parameter[l],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j], Real[k], Real[l],*

Elseif: Saturation Function Option = { [Extended] Brooks and Corey }
Entry Head[c], Units[d] (m), Parameter[e], Minimum Saturation[f],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f],*

Elseif: Saturation Function Option = { Entrapment Brooks and Corey }
Entry Head[c], Units[d] (m), Parameter[e], Minimum Saturation[f],
Effective Gas Residual Saturation[g],
Critical Trapping Number[h],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Real[h],*

Elseif: Saturation Function Option = { Fractured Brooks and Corey }
Matrix Entry Head[c], Units[d] (m), Matrix Parameter[e],
Matrix Minimum Saturation[f],
Fracture Entry Head[g], Units[h] (m), Fracture Parameter[i],

Saturation Function Card (cont'd)

Fracture Minimum Saturation[j],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j],*

Elseif: Saturation Function Option = { Haverkamp }
Entry Head Parameter[c], Units[d] (m), Parameter[e], Parameter[f], Minimum Saturation[g],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*

Elseif: Saturation Function Option = { Tabular [Saturation] [Linear | Spline [Log]] }
Number of Table Entries[a],
Format: Char[a], Char[b], Integer[a]
For: Number of Table Entries
Air-Water Capillary Head[a], Units[b] (m), Aqueous Saturation[c],
Format: *Real[a], Char[b], Real[c],*
Endfor: Number of Table Entries

Elseif: Saturation Function Option = { Tabular Water Content [Linear | Spline [Log]] }
Number of Table Entries[a],
Format: Char[a], Char[b], Integer[a]
For: Number of Table Entries
Air-Water Capillary Head[a], Units[b] (m), Water Content[c],
Format: *Real[a], Char[b], Real[c],*
Endfor: Number of Table Entries

Endif:

Saturation Function Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Energy** }

Rock/Soil Name[a],
Saturation Function[b],
{ [Extended | Fractured] van Genuchten |
[Extended | Fractured] Brooks and Corey |
Tabular [Linear | Spline] [Water Content | Saturation] }

If: Saturation Function Option = { [Extended] van Genuchten }
□ Parameter[c], Units[d] (1/m), n Parameter[e], Minimum Saturation[f],
m Parameter[g],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g],*

Elseif: Saturation Function Option = { Fractured van Genuchten }
Matrix □ Parameter[c], Units[d] (1/m), Matrix n Parameter[e],
Matrix Minimum Saturation[f],
Fracture □ Parameter[g], Units[h] (1/m), Fracture n Parameter[i],
Fracture Minimum Saturation[j],
Matrix m Parameter[k], Fracture m Parameter[l],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j], Real[k], Real[l],*

Elseif: Saturation Function Option = { [Extended] Brooks and Corey }
Entry Head[c], Units[d] (m), □ Parameter[e], Minimum Saturation[f],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f],*

Elseif: Saturation Function Option = { Fractured Brooks and Corey }
Matrix Entry Head[c], Units[d] (m), Matrix □ Parameter[e],
Matrix Minimum Saturation[f],
Fracture Entry Head[g], Units[h] (m), Fracture □ Parameter[i],
Fracture Minimum Saturation[j],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j],*

Elseif: Saturation Function Option = { Tabular [Saturation] [Linear | Spline] }
Number of Table Entries[a],
Format: Char[a], Char[b], Integer[a]
For: Number of Table Entries
Air-Water Capillary Head[a], Units[b] (m), Aqueous Saturation[c],
Format: *Real[a], Char[b], Real[c],*
Endfor: Number of Table Entries

Elseif: Saturation Function Option = { Tabular Water Content [Linear | Spline] }
Number of Table Entries[a],
Format: Char[a], Char[b], Integer[a]
For: Number of Table Entries
Air-Water Capillary Head[a], Units[b] (m), Water Content[c],

Saturation Function Card (cont'd)

Format: *Real[a], Char[b], Real[c],*
Endfor: Number of Table Entries

Endif:

Saturation Function Card (cont'd)

Elseif: Operational Mode Option = { **Water-Oil** } { **Water-Air-Oil** }

Rock/Soil Name[a],
Saturation Function[b]
{ [Residual [Geel] [Land]] [Entrapment] van Genuchten |
[Residual [Geel] [Land]] [Entrapment] Brooks and Corey }

If: Saturation Function Option = { van Genuchten }
 Parameter[c], Units[d] (1/m),
n Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
m Parameter[i],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i],*

Elseif: Saturation Function Option = { Brooks and Corey }
Entry Head[c], Units[d] (m),
 Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
Format: *Char[a], Char[b], Real[e], Char[d], Real[e], Real[f], Real[g], Char[h],*

Elseif: Saturation Function Option = { Entrapment van Genuchten }
 Parameter[c], Units[d] (1/m),
n Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
m Parameter[i], Effective NAPL Maximum Trapped Saturation[j],
Critical NAPL Trapping Number[k],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j], Real[k],*

Elseif: Saturation Function Option = { Entrapment Brooks and Corey }
Entry Head[c], Units[d] (m),
 Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
Effective NAPL Masxium Trapped Saturation[i],
Critical NAPL Trapping Number[j],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j],*

Elseif: Saturation Function Option = { Residual [Geel] [Land] van Genuchten }
 Parameter[c], Units[d] (1/m),
n Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),

Saturation Function Card (cont'd)

m Parameter[i], Effective Maximum Residual NAPL Saturation[j],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j],*

Elseif: Saturation Function Option = { Residual [Geel] [Land] Brooks and Corey }

Entry Head[c], Units[d] (m),
 Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
Effective Maximum Residual NAPL Saturation[i],
Format: *Char[a], Char[b], Real[e], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i],*

Elseif: Saturation Function Option = { Residual Entrapped [Geel] [Land]
van Genuchten }
 Parameter[c], Units[d] (1/m),
n Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
m Parameter[i], Effective Maximum Trapped NAPL Saturation[j],
Effective Maximum Residual NAPL Saturation[k],
Critical NAPL Trapping Number[l],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j], Real[k], Real[l],*

Elseif: Saturation Function Option = { Residual Entrapped [Geel] [Land]
Brooks and Corey }
Entry Head[c], Units[d] (m),
 Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
Effective Maximum Trapped NAPL Saturation[i],
Effective Maximum Residual NAPL Saturation[j],
Critical NAPL Trapping Number[k],
Format: *Char[a], Char[b], Real[e], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j], Real[k],*

Endif:

Saturation Function Card (cont'd)

Elseif: Operational Mode Option = {**Water-Air-Oil-Energy** }

Rock/Soil Name[a],
Saturation Function[b]
{ Nonhysteretic van Genuchten | Nonhysteretic Brooks and Corey }

If: Saturation Function Option = { Nonhysteretic van Genuchten }
 Parameter[c], Units[d] (1/m),
n Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
m Parameter[i],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i],*

Elseif: Saturation Function Option = { Nonhysteretic Brooks and Corey }
Entry Head[c], Units[d] (m),
 Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
Format: *Char[a], Char[b], Real[e], Char[d], Real[e], Real[f], Real[g], Char[h],*

Endif:

Saturation Function Card (cont'd)

Elseif: Operational Mode Option = { **Water-Oil-Dissolved Oil** }

Rock/Soil Name[a],
Saturation Function[b]
{ [Entrapment] van Genuchten | [Entrapment] Brooks and Corey }

If: Saturation Function Option = { van Genuchten }
 Parameter[c], Units[d] (1/m),
n Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
m Parameter[i],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i],*

Elseif: Saturation Function Option = { Brooks and Corey }
Entry Head[c], Units[d] (m),
 Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
Format: *Char[a], Char[b], Real[e], Char[d], Real[e], Real[f], Real[g], Char[h],*

Elseif: Saturation Function Option = { Entrapment van Genuchten }
 Parameter[c], Units[d] (1/m),
n Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
m Parameter[i], Effective NAPL Residual Saturation[j],
Effective Gas Residual Saturation[k], Critical NAPL Trapping Number[l],
Critical Gas Trapping Number[m],
Format: *Char[a], Char[b], Real[c], Char[d],*
Real[e], Real[f], Real[g], Char[h], Real[i], Real[j], Real[k], Real[l], Real[m],

Elseif: Saturation Function Option = { Entrapment Brooks and Corey }
Entry Head[c], Units[d] (m),
 Parameter[e],
Minimum Saturation[f],
Reference Fluid-Pair Interfacial Tension[g], Units[h] (N/m),
Effective NAPL Residual Saturation[i],
Effective Gas Residual Saturation[j], Critical NAPL Trapping Number[k],
Critical Gas Trapping Number[l],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j], Real[k], Real[l],*

Endif:

Saturation Function Card (cont'd)

Elseif: Operational Mode Option = { **Water-CO2-NaCl-Energy** }

Rock/Soil Name[a],
Saturation Function[b],
{ [Entrapment | Extended | Fractured] van Genuchten |
[Entrapment | Extended | Fractured] Brooks and Corey |
Haverkamp | Tabular [Linear | Spline] [Water Content | Saturation] }

If: Saturation Function Option = { [Extended] van Genuchten }
□ Parameter[c], Units[d] (1/m), n Parameter[e], Minimum Saturation[f],
m Parameter[g],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g],*

Elseif: Saturation Function Option = { Entrapment van Genuchten }
□ Parameter[c], Units[d] (1/m), n Parameter[e], Minimum Saturation[f],
m Parameter[g], Effective Gas Residual Saturation[h],
Critical Trapping Number[i],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Real[h], Real[i],*

Elseif: Saturation Function Option = { Fractured van Genuchten }
Matrix □ Parameter[c], Units[d] (1/m), Matrix n Parameter[e],
Matrix Minimum Saturation[f],
Fracture □ Parameter[g], Units[h] (1/m), Fracture n Parameter[i],
Fracture Minimum Saturation[j],
Matrix m Parameter[k], Fracture m Parameter[l],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j], Real[k], Real[l],*

Elseif: Saturation Function Option = { [Extended] Brooks and Corey }
Entry Head[c], Units[d] (m), □ Parameter[e], Minimum Saturation[f],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f],*

Elseif: Saturation Function Option = { Entrapment Brooks and Corey }
Entry Head[c], Units[d] (m), □ Parameter[e], Minimum Saturation[f],
Effective Gas Residual Saturation[g],
Critical Trapping Number[h],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Real[h],*

Elseif: Saturation Function Option = { Fractured Brooks and Corey }
Matrix Entry Head[c], Units[d] (m), Matrix □ Parameter[e],
Matrix Minimum Saturation[f],
Fracture Entry Head[g], Units[h] (m), Fracture □ Parameter[i],
Fracture Minimum Saturation[j],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Real[f], Real[g], Char[h], Real[i], Real[j],*

Saturation Function Card (cont'd)

Elseif: Saturation Function Option = { Haverkamp }
Entry Head Parameter[c], Units[d] (m), □ Parameter[e], □ Parameter[f], Minimum Saturation[g],
Format: *Char[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*

Elseif: Saturation Function Option = { Tabular Saturation [Linear | Spline] }
Number of Table Entries[a],
Format: Char[a], Char[b], Integer[a]
For: Number of Table Entries
Air-Water Capillary Head[a], Units[b] (m), Aqueous Saturation[c],
Format: *Real[a], Char[b], Real[c],*
Endfor: Number of Table Entries

Elseif: Saturation Function Option = { Tabular Water Content [Linear | Spline] }
Number of Table Entries[a],
Format: Char[a], Char[b], Integer[a]
For: Number of Table Entries
Air-Water Capillary Head[a], Units[b] (m), Water Content[c],
Format: *Real[a], Char[b], Real[c],*
Endfor: Number of Table Entries

Endif:

Endif:

Endfor: Number of Rock/Soil Types

Endcard: Saturation Function Card

B.19.1 Saturation Function Card Examples

Extracted from a STOMP1 (Water) input file:
~Saturation Function Card
IJK Indexing, Tabular Log-linear,12,file:tabh.dat,cm,file:tabs.dat,saturation,

Extracted from a STOMP1 (Water) input file:
~Saturation Function Card
Sand,Van Genuchten,1.563,1/m,5.4,0.15,,

Extracted from a STOMP1 (Water) input file:
#R1 is a scaling group
~Saturation Function Card
R1,van Genuchten,0.036,1/cm,1.756,0.092417,,

Extracted from a STOMP1 (Water) input file:
~Saturation Function Card
backfill,van Genuchten,9.885,1/m,2.928,0.0774,,
hanford sand,van Genuchten,0.0092,1/cm,1.8848,0.08366,,
plio-pleistocene,van Genuchten,0.0067,1/cm,1.8378,0.25953,,
upper ringold,van Genuchten,0.0029,1/cm,1.6285,0.21295,,
middle ringold,van Genuchten,0.0062,1/cm,1.6452,0.0686,,

Extracted from a STOMP1 (Water) input file:
~Saturation Function Card
Backfill,Entrapment van Genuchten,0.0210,1/cm,1.3740,0.07246,,0.2,2e-5,
H2 Sand,Entrapment van Genuchten,0.0117,1/cm,1.6162,0.11600,,0.2,2e-5,
H1 Gravelly Sand,Entrapment van Genuchten,0.0141,1/cm,1.3730,0.01505,,0.3,2e-5,
H3 Gravelly Sand,Entrapment van Genuchten,0.0197,1/cm,1.4194,0.05618,,0.2,2e-5,

Extracted from a STOMP4 (Water-Oil) input file:
~Saturation Function Card
72.0,dynes/cm,38.0,dynes/cm,,,
Sand,Brooks and Corey,10.12,cm,2.67,0.08,72.0,dynes/cm,

Extracted from a STOMP4 (Water-Oil) input file:
~Saturation Function Card
72.0,dynes/cm,44.4,dynes/cm,,,
Sand,Entrapment Van Genuchten,2.5,1/m,2.0,.1,72.0,dynes/cm,,0.25,2e-5,

Extracted from a STOMP4 (Water-Oil) input file:
~Saturation Function Card
72.0,dynes/cm,53/1,dynes/cm,24.7,dynes/cm,
Sand,Brooks and Corey w/ Residual,10.12,cm,2.67,0.08,72.0,dynes/cm,0.235,

B.20 Scaling Factor Card

Card Title[a] { ~Scaling [Factor Card] }
Format: *Char*[a]

If: Operational Mode Option Card = { **Water** } { **Fluid** }

If: Execution Mode Option = { Normal w/ Scaling } { Restart w/ Scaling }
Saturated Hydraulic Conductivity Scaling Function[a], { Log | Linear }
Diffusive Porosity Scaling Function[b], { Log | Linear }
Van Genuchten "alpha" or Brooks-Corey "psi" Scaling Function[c], { Log | Linear }
Van Genuchten "n" or Brooks-Corey "lambda" Scaling Function[d], { Log | Linear }
Residual Saturation Scaling Function[e], { Log | Linear }
Format: *Char*[a], *Char*[b], *Char*[c], *Char*[d], *Char*[e],
For: Number of Rock/Soil Types
Rock/Soil Type[a],
If: Rock/Soil Type contains {{ Fractured }} {{ DP }} {{ Dual }}
Saturated Hydraulic Conductivity Scaling Factor[b],
Saturated Hydraulic Conductivity Scaling Factor[c],
Diffusive Porosity Scaling Factor[d],
Diffusive Porosity Scaling Factor[e],
Van Genuchten "alpha" or Brooks-Corey "psi" Scaling Factor[f],
Van Genuchten "alpha" or Brooks-Corey "psi" Scaling Factor[g],
Van Genuchten "n" or Brooks-Corey "lambda" Scaling Factor[h],
Van Genuchten "n" or Brooks-Corey "lambda" Scaling Factor[i],
Residual Saturation Scaling Function Factor[j],
Residual Saturation Scaling Function Factor[k],
Format: *Char*[a], *Real*[b], *Real*[c], *Real*[d], *Real*[e], *Real*[f], *Real*[g], *Real*[h], *Real*[i], *Real*[j], *Real*[k],
If: Permeability Function Option = { Anisotropy Mualem }
Horizontal Pore-Scale Scaling Factor[l],
Horizontal Pore-Scale Scaling Factor[m],
Format: *Char*[a], *Real*[b], *Real*[c], *Real*[d], *Real*[e], *Real*[f], *Real*[g], *Real*[h], *Real*[i], *Real*[j], *Real*[k], *Real*[l], *Real*[m],
Endif:
Else:
Saturated Hydraulic Conductivity Scaling Factor[b],
Diffusive Porosity Scaling Factor[c],
Van Genuchten "alpha" or Brooks-Corey "psi" Scaling Factor[d],
Van Genuchten "n" or Brooks-Corey "lambda" Scaling Factor[e],
Residual Saturation Scaling Function Factor[f],
Format: *Char*[a], *Real*[b], *Real*[c], *Real*[d], *Real*[e], *Real*[f],
If: Permeability Function Option = { Anisotropy Mualem }
Horizontal Pore-Scale Scaling Factor[g],
Horizontal Pore-Scale Scaling Factor[h],
Format: *Char*[a], *Real*[b], *Real*[c], *Real*[d], *Real*[e], *Real*[f], *Real*[g], *Real*[h],
Endif:

Scaling Factor Card (cont'd)

Endfor:
Endfor:
Endif:

Endif:

Endcard: Scaling Factor Card

B.20.1 Scaling Factor Card Examples

Extracted from a STOMP1 (water) input file:
~Scaling Card
Linear,Linear,Linear,Linear,Linear,
L1,1.0000,1.000,1.000,1.000,1.000,
L2,0.0429,0.754,0.933,1.003,0.681,
L3,0.0256,0.580,1.512,1.100,0.681,
L4,0.0105,0.553,1.561,1.667,1.301,

B.21 Simulation Title Card

Card Title[a] { ~Simulation [Title Card] }
Format: *Char*[a]

Version Number[a],
Format: *Integer*[a],

Simulation Title[a],
Format: *Char*[a],

User Name[a],
Format: *Char*[a],

Company Name[a],
Format: *Char*[a],

Input Creation Date[a],
Format: *Char*[a],

Input Creation Time[a],
Format: *Char*[a],

Number of Simulation Note Lines[a],
Format: *Integer*[a],

For: Number of Simulation Note Lines

Simulation Notes[a]
Format: *Char*[a] *(maximum of 132 characters per line)*

Endfor: Number of Simulation Note Lines

Endcard: Simulation Title Card

B.21.1 Simulation Title Card Examples

Extracted from a STOMP1 (Water) input file:
~Simulation Title Card
1,
Scaling Method,
Zhang,
PNNL,
22 June 2001,
13:40,
2,
Rockhold et al.(1988)
Drainage in the Grass Site

Extracted from a STOMP1 (Water) input file:
~Simulation Title Card
1,
C-Tank Farm Simulation: Initial Condition,
ZF Zhang, VL Freedman, and MD White,
PNNL,
January 29 2003,
9:30,
3,
CASE 1, 1945-2050: This input file is used to simulate transport (U and TC)
using an initial concentration condition profile.
Cross section for Tank Row C-103, C-106, C-109, and C-112.

Extracted from a STOMP1 (Water) input file:
~Simulation Title Card
1,
Field Test #12,
MD White,
PNNL,
Monday April 5 1996,
12:34,
2,
Simulation of field test #12 at Edwards AFB.
Starting time 1/3/96 14:19, Ending time 1/8/96 9:19.

Extracted from a STOMP3 (Water-Air-Energy) input file:
~Simulation Title Card
1,
Evaporation/Condensation Heat Pipe,
MD White,
PNNL,
June 18 1994,
10:04 AM PDT,
3,
This application problem follows the heat-pipe problem solved
semi-analytically by Udell and Fitch. The Soil moisture retention
function has been changed to a modified van Genuchten function.

Simulation Title Card Examples (cont'd)

Extracted from a STOMP4 (Water-Oil) input file:

```
~Simulation Title Card
1,
DW2 experiment for Partioning Tracer Study,
Mart Oostrom,
PNNL,
Jan-00,
15:05,
3,
Partitioning tracer test for detection, estimation, and remediation performance
assessment of subsurface nonaqueous phase liquids.  Jin et al.
Water Resources Research, Vol. 31, No. 5, Pages 1201-1211, May 1995.
```

Extracted from a STOMP4 (Water-Oil) input file:

```
~Simulation Title Card
1,
Brooklawn Site: Disposal Scenario,
Mark  White,
Tel: (509) 372-6070, E-mail: mark.white@pnl.gov,
Pacific Northwest National Laboratory,
03-Mar-00,
4,
Simulation to establish an initial aqueous flow field in a cross-section
extending southward along the E. 6150 rdinate line from N. 5000 to N. 4000,
extending vertically from ground surface to a depth of -170 ft-msl, using
the steady-state calibration results.
```

Extracted from a STOMP32 (Water-CO2-NaCl) input file:

```
~Simulation Title Card
1,
BP_UTCOMP Comparison (Base Case),
M.D.  White,
Pacific Northwest Laboratory,
30 October 2002,
04:06 PM PST,
23,
CC*****************************************************************
CC                                    *
CC  BRIEF DESCRIPTION OF DATA SET: UTCOMP ( VERSION UTCOMP-3.5 )     *
CC                                    *
CC*****************************************************************
CC Radial geometry                                *
CC Mt. Simon Sst. Run BPIN1 2D Full scale,radial model            *
CC (phase 1=inactivewater,phase 2 = water/oil,phase 3=co2/gas       *
CC LENGTH(FT):              INJECTION FLUID: CO2              *
CC HEIGHT(FT):              INJECTION RATE: cont. pre.         *
CC WIDTH(FT): variable       W/O REL. PERM:                   *
CC POROSITY: variable        G/O REL. PERM: lindeburg          *
CC ABS. PERM(MD): variable    3-PHASE REL. PERM: water endpt.=1.0   *
CC TEMP(F):   .0          WETTIBILITY:                      *
```

Simulation Title Card Examples (cont'd)

```
CC PRESSURE(PSI): .    psi/ft  W/O CAP. PRESSURE:                 *
CC SOR:                    G/O CAP. PRESSURE:                 *
CC SWC:                      DISPLACEMENT TYPE: HORIZONTAL        *
CC stop injection after 20 years run for 40 yrs,h2o k endpoint=1    *
CC ****NON-IDEAL MIXING, NO GRAVITY, WITH Pc, WITH X-FLOW ****        *
CC FILE NAME:                                          *
CC CREATED BY Neeraj Gupta                                  *
CC MODIFIED BY Neeraj Gupta,                                 *
CC*****************************************************************
```

B.22 Solute/Fluid Interactions Card

Card Title[a] { ~Solute/Fluid [Interactions Card] }
Format: *Char*[a]

Number of Solutes
Format: *Integer*[a],

For: Number of Solutes

If: Operational Mode Option Card = { **Water** } { **Fluid** }
Solute Name[a],
Effective Diffusion Option[b],

{ Constant | **Note:** $D^C_{\ell\,e} = \overline{D}^C_{\ell\,e}$

Conventional | **Note:** $D^C_{\ell\,e} = \tau_\ell s_\ell n_D D^C_\ell$

Empirical [Kemper and van Schaik] } **Note:** $D^C_{\ell\,e} = D^C_\ell \, a \exp(b n_D s_\ell)$
Model parameters for the Empirical option are entered in the Solute/Porous Medium Interaction Card

If: Effective Diffusion Option = { Empirical }
Solute-Aqueous Partition Option[c],

{ Continuous | **Note:** $C_\ell = \dfrac{C}{n_D s_\ell + (1 - n_T)\rho_s K_{s\ell}}$

Noncontinuous } **Note:** $C_\ell = \dfrac{C}{n_D s_\ell + s_\ell (1 - n_T)\rho_s K_{s\ell}}$

IfDef: Radioactive
Half-Life[d], Units[e] (s),
Format: *Char*[a], *Char*[b], *Char*[c], *Real*[d], *Char*[e],
ElseifDef: Reactive
Number of Reactions[d],
For: Number of Reactions:
First-Order Decay Constant[e], Units[f] (s),
Endfor: Number of Reactions
Format: *Char*[a], *Char*[b], *Char*[c], *Integer*[d],<*Real*[e], *Char*[f],>
EndifDef:

Else:
Aqueous-Phase Molecular Diffusion Coefficient @ 20 C[c], Units[d] (m^2/s),
Solute Partition Option[e],

{ Continuous | **Note:** $C_\ell = \dfrac{C}{n_D s_\ell + (1 - n_T)\rho_s K_{s\ell}}$

Noncontinuous } **Note:** $C_\ell = \dfrac{C}{n_D s_\ell + s_\ell (1 - n_T)\rho_s K_{s\ell}}$

IfDef: Radioactive
Half-Life[f], Units[g] (s),
Format: *Char[a], Char[b], Real[c], Char[d], Char[e], Real[f], Char[g],*
If: Operational Mode Option ={ Water Courant Vadose Transport }
Aqueous-Phase Cut-off Conc[h], Units[i] (1/m^3),
Format: *Char[a], Char[b], Real[c], Char[d], Char[e], Real[f], Char[g], Real[h], Char[i],*
Endif:
ElseifDef: Reactive
Number of Reactions[f],
For: Number of Reactions:
First-Order Reaction Rate Constant[g], Units[h] (s),
Endfor: Number of Reactions
Format: *Char[a], Char[b], Real[c], Char[d], Char[e], Integer[f], <Real[g], Char[h],>*
EndifDef:
Endif:

Elseif: Operational Mode Option Card = { **Water-Salt** }
Solute Name[a],
Effective Diffusion Option[b],

{ Constant | **Note:** $D^C_{\ell\,e} = \overline{D}^C_{\ell\,e}$

Conventional | **Note:** $D^C_{\ell\,e} = \tau_\ell s_\ell n_D D^C_\ell$

Empirical [Kemper and van Schaik] } **Note:** $D^C_{\ell\,e} = D^C_\ell\, a \exp(b n_D s_\ell)$
Model parameters for the Empirical option are entered in the Solute/Porous Medium Interaction Card

If: Effective Diffusion Option = { Empirical }
Solute-Aqueous Partition Option[c],

{ Continuous | **Note:** $C_\ell = \dfrac{C}{n_D s_\ell + (1 - n_T)\rho_s K_{s\ell}}$

Solute/Fluid Interactions Card (cont'd)

Noncontinuous } **Note:** $C_\ell = \dfrac{C}{n_D s_\ell + s_\ell (1 - n_T) \rho_s K_{s\ell}}$

IfDef: Radioactive
Half-Life[d], Units[e] (s),
Format: *Char[a], Char[b], Char[c], Real[d], Char[e],*
ElseifDef: Reactive
Number of Reactions[d],
For: Number of Reactions:
First-Order Reaction Rate Constant[e], Units[f] (s),
Endfor: Number of Reactions
Format: *Char[a], Char[b], Char[c], Integer[d],<Real[e], Char[f],>*
EndifDef:
Else:
Aqueous-Phase Molecular Diffusion Coefficient @ 20 C[c], Units[d] (m^2/s),
Solute Partition Option[e],

{ Continuous | **Note:** $C_\ell = \dfrac{C}{n_D s_\ell + (1 - n_T) \rho_s K_{s\ell}}$

Noncontinuous } **Note:** $C_\ell = \dfrac{C}{n_D s_\ell + s_\ell (1 - n_T) \rho_s K_{s\ell}}$

IfDef: Radioactive
Half-Life[f], Units[g] (s),
Format: *Char[a], Char[b], Real[c], Char[d], Char[e], Real[f], Char[g],*
ElseifDef: Reactive
Number of Reactions[f],
For: Number of Reactions:
First-Order Decay Constant[g], Units[h] (s),
Endfor: Number of Reactions
Format: *Char[a], Char[b], Real[c], Char[d], Char[e], Integer[f], <Real[g], Char[h],>*
EndifDef:
Endif:

Elseif: Operational Mode Option = { **Water-Air** } { **Water-Air-Energy** }
{ **Water-Air-Salt** } { **Water-Air-Salt-Energy** }
Solute Name[a],
Aqueous-Phase Molecular Diffusion Coefficient @ 20 C[b], Units[c] (m^2/s),
Gas-Phase Molecular Diffusion Coefficient @ 20 C[d], Units[e] (m^2/s),
Gas-Aqueous Partition Function Option[f]

{ Constant | **Note:** $K_{g\ell} = \overline{K}_{gl}$

Solute/Fluid Interactions Card (cont'd)

Temperature Dependent } **Note:** $\ln(K_{g\ell}) = a + \dfrac{b}{T} + c\ln(T) + dT + eT^2$

If: Gas-Aqueous Partition Function Option = { Constant }
Gas-Aqueous Partition Coefficient[g], Units[h] (m^3/m^3),

Solute-Aqueous Partition Option[i],

{ Continuous | **Note:** $C_\ell = \frac{C}{n_D s_\ell + (1 - n_T)\rho_s K_{s\ell}}$

Noncontinuous } **Note:** $C_\ell = \frac{C}{n_D s_\ell + s_\ell (1 - n_T)\rho_s K_{s\ell}}$

IfDef: Radioactive
Half-Life[j], Units[k] (s),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Char[h], Char[i], Real[j], Char[k],*
ElseifDef: Reactive
Number of Parent Reactions[j],
For: Number of Reactions
First-Order Reaction Rate Constant[k], Units[l] (s),
Endfor: Number of Reactions
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Char[h], Char[i], Integer[j], <Real[k], Char[l],>*
EndifDef:

Elseif: Gas-Aqueous Partition Function Option = { Temperature Dependent }
Constant a[g], Constant b[h], Constant c[i], Constant d[j], Constant e[k],
Solute-Aqueous Partition Option[l],

{ Continuous | **Note:** $C_\ell = \frac{C}{n_D s_\ell + (1 - n_T)\rho_s K_{s\ell}}$

Noncontinuous } **Note:** $C_\ell = \frac{C}{n_D s_\ell + s_\ell (1 - n_T)\rho_s K_{s\ell}}$

IfDef: Radioactive
Half-Life[m], Units[n] (s),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Real[h], Real[i], Real[j], Real[k], Char[l], Real[m], Char[n],*
ElseifDef: Reactive
Number of Parent Reactions[m],
For: Number of Reactions
First-Order Reaction Rate Constant[n], Units[o] (s),

Solute/Fluid Interactions Card (cont'd)

Endfor: Number of Reactions
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Real[h], Real[i], Real[j], Real[k], Integer[l], Char[m], <Real[n], Char[o],>*
EndifDef:
Endif:

Elseif: Operational Mode Option = { **Water-Oil** }
Solute Name[a], Aqueous-Phase Molecular Diffusion Coefficient @ 20 C[b], Units[c] (m^2/s),
NAPL Molecular Diffusion Coefficient @ 20 C[d], Units[e] (m^2/s),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e],*

Aqueous-NAPL Adsorpton Function[a], { Linear | Freundlich | Langmuir }
If: Aqueous-NAPL Adsorpton Function = { Linear }
Linear k[b],
Format: *Char[a], Real[b],*
Elseif: Aqueous-NAPL Adsorpton Function = { Freundlich }
k parameter[b], n parameter[c],
Format: *Char[a], Real[b], Real[c],*
Elseif: Aqueous-NAPL Adsorpton Function = { Langmuir }
a parameter[b], b parameter[c], Units[d] (m^3),
Format: *Char[a], Real[b], Real[c], Char[d],*
Endif:
IfDef: Radioactive
Half-Life[a], Units[b] (s),
Format: *Real[a], Char[b],*
ElseifDef: Reactive
Number of Parent Reactions[a]
For: Number of Reactions
First-Order Reaction Rate Constant[b], Units[c] (s),
Endfor: Number of Reactions
Format: *Integer[a], <Real[b], Char[c],>*
EndifDef:

Elseif: Operational Mode Option = { **Water-Oil-Dissolved Oil** }
Solute Name[a], Aqueous-Phase Molecular Diffusion Coefficient[b], Units[c] (m^2/s),
NAPL Molecular Diffusion Coefficient[d] @ 20 C, Units[e] (m^2/s),
Aqueous-NAPL Partition Coefficient[f], Units[g] (m^3/m^3),
IfDef: Radioactive
Half-Life[h], Units[i] (s),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Real[f], Char[g], Real[h], Char[i],*
ElseifDef: Reactive
Number of Parent Reactions[h],
For: Number of Reactions
First-Order Reaction Rate Constant[i], Units[j] (s),
Endfor: Number of Reactions
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Real[f], Char[g],Integer[h], <Real[i], Char[j],>*

Solute/Fluid Interactions Card (cont'd)

EndifDef:

Elseif: Operational Mode Option{ **Water-Air-Oil** } { **Water-Air-Oil-Energy** }
Solute Name[a],
Aqueous-Phase Molecular Diffusion Coefficient[b] @ 20 C, Units[c] (m^2/s),
Gas-Phase Molecular Diffusion Coefficient[d] @ 20 C, Units[e] (m^2/s),
NAPL Molecular Diffusion Coefficient[f] @ 20 C, Units[g] (m^2/s),
Gas-Aqueous Partition Function Option[h]

{ Constant | **Note:** $K_{g\ell} = \overline{K}_{gl}$

Temperature Dependent } **Note:** $\ln(K_{g\ell}) = a + \frac{b}{T} + c\ln(T) + dT + eT^2$

If: Gas-Aqueous Partition Function Option = { Constant }

Gas-Aqueous Partition Coefficient[i], Units[j] (m^3/m^3),
Aqueous-NAPL Partition Coefficient[k], Units[l] (m^3/m^3),
IfDef: Radioactive
Half-Life[m], Units[n] (s),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Char[h], Real[i], Char[j], Real[k], Char[l], Real[m], Char[n],*
ElseifDef: Reactive
Number of Parent Reactions[m],
For: Number of Reactions
First-Order Reaction Rate Constant[n], Units[o] (s),
Endfor: Number of Reactions
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Char[h], Real[i],Char[j], Real[k], Char[l], Integer[m], <Real[n], Char[o],>*
EndifDef:
Elseif: Gas-Aqueous Partition Function Option = { Temperature Dependent }
Constant a[i], Constant b[j], Constant c[k], Constant d[l], Constant e[m],
Aqueous-NAPL Partition Coefficient[n], Units[o] (m^3/m^3),
IfDef: Radioactive
Half-Life[p], Units[q] (s)
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Char[h], Real[i], Real[j], Real[k], Real[l],Real[m],*
Real[n], Char[o], Real[p], Char[q],
ElseifDef: Reactive
Number of Parent Reactions[p],
For: Number of Reactions
First-Order Reaction Rate Constant[q], Units[r] (s),
Endfor: Number of Reactions
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Char[h], Real[i], Real[j], Real[k], Real[l], Real[m], Real[n], Char[o], Integer[p], <Real[q], Char[r],>*
EndifDef:

Solute/Fluid Interactions Card (cont'd)

Endif:

Elseif: Operational Mode Option = { **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy** }
Solute Name[a],
Aqueous-Phase Molecular Diffusion Coefficient @ 20 C[b], Units[c] (m^2/s),
Gas-Phase Molecular Diffusion Coefficient @ 20 C[d], Units[e] (m^2/s),
Gas-Aqueous Partition Function Option[f]

{ Constant | **Note:** $K_{g\ell} = \overline{K}_{gl}$

Temperature Dependent } **Note:** $\ln\left(K_{g\ell}\right) = a + \frac{b}{T} + c\ln\left(T\right) + dT + eT^2$

If: Gas-Aqueous Partition Function Option = { Constant }
Gas-Aqueous Partition Coefficient[g], Units[h] (m^3/m^3),
IfDef: Radioactive
Half-Life[i], Units[j] (s),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Char[h], Real[i], Char[j],*
ElseifDef: Reactive
Number of Parent Reactions[j],

For: Number of Reactions
First-Order Reaction Rate Constant[k], Units[l] (s),
Endfor: Number of Reactions
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Char[h], Char[i], Integer[j], <Real[k], Char[l],>*
EndifDef:

Elseif: Gas-Aqueous Partition Function Option = { Temperature Dependent }
Constant a[g], Constant b[h], Constant c[i], Constant d[j], Constant e[k],
IfDef: Radioactive
Half-Life[l], Units[m] (s),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Real[h], Real[i], Real[j], Real[k], Real[l], Char[m],*
ElseifDef: Reactive
Number of Parent Reactions[l],
For: Number of Reactions
First-Order Reaction Rate Constant[m], Units[n] (s),
Endfor: Number of Reactions
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Char[f], Real[g], Real[h],Real[i], Real[j], Real[k], Integer[l], <Real[m], Char[n],>*
EndifDef:
Endif:

Endif:

Solute/Fluid Interactions Card (cont'd)

Endfor: Number of Solutes

IfDef: Radioactive
Number of Chain Decay Lines[a],
Format: *Integer[a],*
For: Number of Chain Decay Lines
Parent Solute Name[a], Progeny Solute Name[b], Chain Decay Fraction[c],
Format: *Char[a], Char[b], Real[c],*
Endfor: Number of Chain Decay Lines
ElseifDef: Reactive
Number of Reaction Lines[a],
Format: *Integer[a],*
For: Number of Reaction Lines
Parent Solute Name[a], Progeny Solute Name[b], Reaction Number[c],
Reaction Stoichiometry Fraction[d],
Format: *Char[a], Char[b], Real[c], Real[d],*
Endfor: Number of Reaction Lines
EndifDef

Endcard: Solute/Fluid Interactions Card

B.22.1 Solute/Fluid Interaction Card Examples

Extracted from a STOMP1 (Water) input file:
~Solute/Fluid Interaction Card
1,
TCE,Conventional,9.6283e-6,cm^2/s,Continuous,1000,yr,
0,

Extracted from a STOMP1 (Water) input file:
~Solute/Fluid Interaction Card
4,
Tc-99,Empirical,Noncontinuous,1.e20,yr,
U-238,Empirical,Noncontinuous,1.e20,yr,
Np-237,Empirical,Noncontinuous,1.e20,yr,
Pu-239,Empirical,Noncontinuous,1.e20,yr,
0,

Extracted from a STOMP1 (Water) input file:
~Solute/Fluid Interaction Card
7,
Dithionite,Conventional,1.e-5,cm^2/s,Continuous,2,5,hr,18,hr,
Sulfite,Conventional,1.e-5,cm^2/s,Continuous,0,
Bisulfite,Conventional,1.e-5,cm^2/s,Continuous,0,
Thiosulfate,Conventional,1.e-5,cm^2/s,Continuous,0,
Fe(III),Conventional,1.e-5,cm^2/s,Continuous,0,
Fe(II),Conventional,1.e-5,cm^2/s,Continuous,0,
H+,Conventional,1.e-5,cm^2/s,Continuous,0,
6,
Dithionite,Fe(III),1,-2.0,
Dithionite,Fe(II),1,2.0,
Dithionite,Sulfite,1,2.0,
Dithionite,H+,1,4.0,
Dithionite,Bisulfite,2,1.0,
Dithionite,Thiosulfate,2,0.5,

Extracted from a STOMP4 (Water-Oil) input file:
~Solute/Fluid Interactions Card
Note different format for Water-Oil mode!
3,
Tritium,1.0e-10,m^2/s,1.0e-10,m^2/s,
Linear Isotherm,0.0,
1.e+8,yr,
IPA,1.0e-10,m^2/s,1.0e-10,m^2/s,
Linear Isotherm,0.04,
1.e+8,yr,
DMB,1.0e-10,m^2/s,1.0e-10,m^2/s,
Linear Isotherm,2.76,
1.e+8,yr,
0,

B.23 Solute/Porous Media Interactions Card

Card Title[a] { ~Solute/Porous [Media Interactions Card] }
Format: *Char*[a]

For: Number of Rock/Soil Types

Rock/Soil or Scaling Group Name[a],

If: Rock/Soil or Scaling Group Name = { IJK | JKI | KIJ } Indexing

Note: A parameter value input can be replaced with an external file using the following formatting for ASCII files:

file: *filename*

or the following formattings for binary files:

binary file: *filename*

where; the external file will contain unique parameter values for each node (active or inactive) arranged according to the indexing scheme (i.e., IJK, JKI, or KIJ). Applicable units will be applied to all parameter values in the external file.

Endif:

Longitudinal Dispersivity[b], Units[c] (m),
Transverse Dispersivity[d], Units[e] (m)
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Char*[e],

If: Operational Mode Option = { **Water** } { **Fluid** }
For: Number of Solutes
Solute Name[a],
Solid-Aqueous Partition Coefficient[b], Units[c] (m^3/kg),
If: Effective Diffusion Option: Solute/Fluid Interactions Card = { Empirical }
Aqueous Molecular Diffusion Coefficient[d], Units[e] (m^2/s),
Solute/Fluid Interactions Card: Constant a[f],
Solute/Fluid Interactions Card: Constant b[g],
[Macrodispersity Enhancement Factor[h],]
Format: *Char*[a], *Real*[b], *Char*[c], *Real*[d], *Char*[e], *Real*[f], *Real*[g], *[Real*[h]*,]*
Else:
[Macrodispersivity Enhancement Factor[d],]
Format: *Char*[a], *Real*[b], *Char*[c], *[Real*[d]*,]*
Endif:
Endfor: Number of Solutes

Elseif: Operational Mode Option = { **Water-Salt** }
For: Number of Solutes
Solute Name[a],
Solid-Aqueous Partition Coefficient[b], Units[c] (m^3/kg),

If: Effective Diffusion Option: Solute/Fluid Interactions Card = { Empirical }
Aqueous Molecular Diffusion Coefficient[d], Units[e] (m^2/s),
Solute/Fluid Interactions Card: Constant a[f],
Solute/Fluid Interactions Card: Constant b[g],
Format: *Char[a], Real[b], Char[c], Real[d], Char[e], Real[f], Real[g],*
Else:
Format: *Char[a], Real[b], Char[c],*
Endif:
Endfor: Number of Solutes

Elseif: Operational Mode Option Card = { **Water-Air** }
{ **Water-Air-Energy** } { **Water-Oil** } { **Water-Air-Oil** } { **Water-Air-Oil-Energy** }
{ **Water-Oil-Dissolved Oil** } { **Water-Air-Salt** } { **Water-Air-Salt-Energy** }
{ **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy** }
For: Number of Solutes
Solute Name[a],
Solid-Aqueous Partition Coefficient[b], Units[c] (m^3/kg),
Format: *Char[a], Real[b], Char[c],*
Endfor: Number of Solutes
Endif:

Endfor: Number of Rock/Soil Types

Endcard: Solute/Porous Media Interactions Card

B.23.1 Solute/Porous Media Interactions Card Examples

Extracted from a STOMP1 (Water) input file:
~Solute/Porous Media Interaction Card
Backfill Soil,,,,,
Tc-99,0.,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
U-238,0.67,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Np-237,3.0,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Pu-239,21.0,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Hanford Sand,,,,,
Tc-99,0.,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
U-238,0.67,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Np-237,3.0,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Pu-239,21.0,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Hanford Gravel,,,,,
Tc-99,0.,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
U-238,0.67,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Np-237,3.0,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Pu-239,21.0,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Ringold Formation,,,,,
Tc-99,0.,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
U-238,0.67,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Np-237,3.0,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Pu-239,21.0,cm^3/g,7.8894e-2,m^2/yr,0.005,10.,
Glass Waste,,,,,
Tc-99,0.,cm^3/g,3.1557e-7,m^2/yr,1.,0.,
U-238,0.,cm^3/g,3.1557e-7,m^2/yr,1.,0.,
Np-237,0.,cm^3/g,3.1557e-7,m^2/yr,1.,0.,
Pu-239,0.,cm^3/g,3.1557e-7,m^2/yr,1.,0.,

Extracted from a STOMP1 (Water) input file:
~Solute/Porous Media Interaction Card
Soil,1,m,0.1,m,
Dithionite,0.0,m^3/kg,
Sulfite,0.0,m^3/kg,
Bisulfite,0.0,m^3/kg,
Thiosulfate,0.0,m^3/kg,
Fe(III),0.0,m^3/kg,
Fe(II),0.0,m^3/kg,
H+,0.0,m^3/kg,

Extracted from a STOMP1 (Water) input file:
~Solute/Porous Media Interaction Card
Backfill,150,cm,15,cm,
U:0.01, 0.01,ml/g,1.067,
U:0.03, 0.03,ml/g,1.067,
U:0.10, 0.10,ml/g,1.067,
U:0.30, 0.30,ml/g,1.067,
U:0.60, 0.60,ml/g,1.067,
U:1.00, 1.00,ml/g,1.067,
Tc, 0.00,ml/g,1.0,
Solute/Porous Media Interaction Card (cont'd)

H2 Sand,150,cm,15,cm,
U:0.01, 0.01,ml/g,1.063,
U:0.03, 0.03,ml/g,1.063,
U:0.10, 0.10,ml/g,1.063,
U:0.30, 0.30,ml/g,1.063,
U:0.60, 0.60,ml/g,1.063,
U:1.00, 1.00,ml/g,1.063,
Tc, 0.00,ml/g,1.0,
H1 Gravelly Sand,100,cm,10,cm,
U:0.01, 0.01,ml/g,1.120,
U:0.03, 0.03,ml/g,1.120,
U:0.10, 0.10,ml/g,1.120,
U:0.30, 0.30,ml/g,1.120,
U:0.60, 0.60,ml/g,1.120,
U:1.00, 1.00,ml/g,1.120,
Tc, 0.00,ml/g,1.0,
H3 Gravelly Sand,100,cm,10,cm,
U:0.01, 0.01,ml/g,1.062,
U:0.03, 0.03,ml/g,1.062,
U:0.10, 0.10,ml/g,1.062,
U:0.30, 0.30,ml/g,1.062,
U:0.60, 0.60,ml/g,1.062,
U:1.00, 1.00,ml/g,1.062,
Tc, 0.00,ml/g,1.0,

B.24 Solution Control Card

Card Title[a] {~Solution [Control Card] }
Format: *Char*[a]

Execution Mode Option[a],
If: Operational Mode Option = { **Water** } { **Fluid** }
Execution Mode Option[a],
{ Normal [No Flow] [Dynamic Domain] [Second Order] [Scaling] [Inverse] |
Restart [No Flow] [Dynamic Domain] [File] [Mode] [Second Order] |
[Scaling] [Inverse] | Initial Conditions }
Else:
Execution Mode Option[a],
{ Normal [Second Order] |
Restart [File] [Second Order] |
Initial Conditions }
Endif:

If: Execution Mode Option = { Restart [File] }
Restart File Name[b],
Format: *Char*[a], *Char*[b],
Else:
Format: *Char*[a],
Endif:

Operational Mode Option[a],
{ Water [[[Leonard] TVD | [Roe] Superbee | First-Order [Upwind]] [Electrolyte]
[Courant [Vadose]] Transport] |
Fluid [[[Leonard] TVD | [Roe] Superbee | First-Order [Upwind]] [Courant]
Transport] |
Water-Air [[[Leonard] TVD | [Roe] Superbee | First-Order [Upwind]] [Courant]
Transport] |
Water-Air-Energy [[[Leonard] TVD | [Roe's] Superbee | First-Order [Upwind]] [Courant]
Transport] |
Water-Oil [LFL] [[[Leonard] TVD | [Roe] Superbee | First-Order [Upwind]] [Courant]
Transport] |
Water-Air-Oil [[[Leonard] TVD | [Roe] Superbee | First-Order [Upwind]] [Courant]
Transport] |
Water-Air-Oil-Energy [[[Leonard] TVD | [Roe] Superbee | First-Order [Upwind]]
[Courant] Transport] |
Water-Oil-Dissolved Oil [LFL] [[[Leonard] TVD | [Roe's] Superbee |
First-Order [Upwind]] [Courant] Transport] |
Water-Salt [LFL] [[[Leonard] TVD | [Roe] Superbee | First-Order [Upwind]]
[Courant [Vadose]] Transport] [Osmotic] [Surface [Tension]] |
Water-Air-Salt [LFL] [[[Leonard] TVD | [Roe] Superbee |
First-Order [Upwind]] [Courant] Transport] [Osmotic] [Surface [Tension]] |

Solution Control Card (cont'd)

Water-Air-Salt-Energy [LFL] [[[Leonard] TVD | [Roe] Superbee |
First-Order [Upwind]] [Courant] Transport] [Osmotic] [Surface [Tension]] |
Water-CO2-NaCl [LFL] [[[Leonard] TVD | [Roe] Superbee |
First-Order [Upwind]] [Courant] Transport] [Osmotic] [Surface [Tension]] |

Water-CO2-NaCl-Energy [LFL] [[[Leonard] TVD | [Roe] Superbee |
First-Order [Upwind]] [Courant] Transport] [Osmotic] [Surface [Tension]] }

If: Maximum Courant Number is specified (optional):
Maximum Courant Number[b],
Format: *Char[a], Real[b],*
Else:
Format: *Char[a],*
Endif:

If: Execution Mode Option = { Initial Conditions }
Endcard: Solution Control Card
Endif:

Number of Execution Time Periods[a],
Format: *Integer[a],*

For: Number of Execution Time Periods
If: Execution Mode Option = { Normal }
Initial Time[a], Units[b] (s), Final Time[c], Units[d] (s),
Initial Time Step[e], Units[f] (s),
Maximum Time Step[g], Units[h] (s),
Time Step Acceleration Factor[i],
Maximum Number of Newton-Raphson Iterations[j],
Convergence Criterion[k],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Integer[j], Real[k],*
ElseIf: Execution Mode Option = { Restart }
Initial Time[a], Units[b] (s), Final Time[c], Units[d] (s),
Initial Time Step[e], Units[f] (s),
Maximum Time Step[g], Units[h] (s),
Time Step Acceleration Factor[i],
Maximum Number of Newton-Raphson Iterations[j],
Convergence Criterion[k],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Integer[j], Real[k],*
Endif:
Endfor: Number of Execution Time Periods

Maximum Number of Time Steps[a]
Format: *Integer[a],*

If: Operational Mode Option = { **Water-Air** } { **Water-Air-Energy** }
Solution Control Card (cont'd)

Aqueous Diffusion Option[a],
{ Zero }
{ Constant }
{ Variable }

If: Aqueous Diffusion Option = { Constant }
Dissolved Air Diffusion Coefficient[b], Units[c] (m^2/s),
Format: *Char[a], Real[b], Char[c],*
Else:
Format: *Char[a],*

Endif:
Endif:

If: Operational Mode Option = { **Water-Oil** } { **Water-Oil-Dissolved Oil** }
Aqueous Diffusion Option[a],
{ Zero }
{ Constant }
{ Variable }
If: Aqueous Diffusion Option = { Constant }
Dissolved Oil Diffusion Coefficient[b], Units[c] (m^2/s),
Format: *Char[a], Real[b], Char[c],*
Else:
Format: *Char[a],*
Endif:
Endif:

If: Operational Mode Option = { **Water-Air-Oil** } { **Water-Air-Oil-Energy** }
Aqueous Diffusion Option[a],
{ Zero }
{ Constant }
{ Variable }
If: Aqueous Diffusion Option = { Constant }
Dissolved Air Diffusion Coefficient[b], Units[c] (m^2/s),
Dissolved Oil Diffusion Coefficient[d], Units[e] (m^2/s),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e],*
Else:
Format: *Char[a],*
Endif:
Endif:

If: Operational Mode Option = { **Water-Air-Salt** } { **Water-Air-Salt-Energy** }
Aqueous Diffusion Option[a],
{ Zero }
{ Constant }
{ Variable }
Solution Control Card (cont'd)

If: Aqueous Diffusion Option = { Constant }
Dissolved Air Diffusion Coefficient[b], Units[c] (m^2/s),
Dissolved Salt Diffusion Coefficient[d], Units[e] (m^2/s),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e],*
Else:
Format: *Char[a],*
Endif:
Endif:

If: Operational Mode Option = { **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy** }
Aqueous Diffusion Option[a],
{ Zero }
{ Constant }
{ Variable }
If: Aqueous Diffusion Option = { Constant }
Dissolved CO_2 Diffusion Coefficient[b], Units[c] (m^2/s),

Dissolved Salt Diffusion Coefficient[d], Units[e] (m^2/s),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e],*
Else:
Format: *Char[a],*
Endif:
Endif:

If: Operational Mode Option = { **Water-Air** } { **Water-Air-Energy** } { **Water-Air-Salt** } { **Water-Air-Salt-Energy** } { **Water-CO2-NaCl** } { **Water-CO2-NaCl-Energy** }
Gas Diffusion Option[b],
{ Zero }
{ Constant }
{ Variable }
{ Enhanced }
If: Gas Diffusion Option = { Constant }
Water Vapor Diffusion Coefficient[b], Units[c] (m^2/s),
Format: *Char[a], Real[b], Char[c],*
Elseif: Gas Diffusion Option = { Enhanced }
Clay Mass Fraction[b],
Format: *Char[a], Real[b],*
Endif:
Endif:

If: Operational Mode Option = { **Water-Air-Oil** } { **Water-Air-Oil-Energy** }
Gas Diffusion Option[b],
{ Zero }
{ Constant }
{ Variable }
{ Enhanced }

Solution Control Card (cont'd)

If: Gas Diffusion Option = { Constant }
Water Vapor Diffusion Coefficient[b], Units[c] (m^2/s),
Oil Vapor Diffusion Coefficient[d], Units[e] (m^2/s),
Format: *Char[a], Real[b], Char[c], Real[d], Char[e],*
Elseif: Gas Diffusion Option = { Enhanced }
Clay Mass Fraction[b],
Format: *Char[a], Real[b],*
Endif:
Endif:

If: Operational Mode Option = { **Fluid** }
Fluid Density[a], Unit[b] (kg/m^3),
Fluid Viscosity[c], Unit[d] (Pa s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:

If: Operational Mode Option = { **Water w/ Electrolyte Solute Transport** }
{ Fluid w/ Electrolyte Solute Transport }
Electrolyte Name[a],
Format: *Char[a],*
Electrolyte Density Function Option[a] = { Leijnse | Fourth }
If: Electrolyte Density Function Option= { Leijnse }

Exponential Coefficient[b],
Format: *Char[a], Real[b],*
Elseif: Electrolyte Density Function Option= { Fourth }
Units[b] (1/m^3), Polynomial "a" Coefficient[c], Polynomial "b" Coefficient[d],
Polynomial "c" Coefficient[e], Polynomial "d" Coefficient[f],
Format: *Char[a], Char[b], Real[c] ,Real[d] ,Real[e], Real[f],*
Endif:
Electrolyte Viscosity Function Option[a] = { Leijnse | Fourth }
If: Electrolyte Viscosity Function Option= { Leijnse }
Units[b] (1/m^3), Leijnse "a" Coefficient[c], Leijnse "b" Coefficient[d],
Leijnse "c" Coefficient[e], Leijnse "d" Coefficient[f],
Format: *Char[a], Char[b], Real[c], Real[d], Real[e], Real[f],*
Elseif: Electrolyte Viscosity Function Option= { Fourth }
Units[b] (1/m^3), Polynomial "a" Coefficient[c], Polynomial "b" Coefficient[d],
Polynomial "c" Coefficient[e], Polynomial "d" Coefficient[f],
Format: *Char[a], Char[b], Real[c], Real[d], Real[e], Real[f],*
Endif:
Endif:

Number of Interfacial Averaging Variables[a]
Format: *Integer[a],*

For: Number of Interfacial Averaging Variables
Solution Control Card (cont'd)

Surface Variable Option[a],
{ Aqueous Density | Aqueous Relative Permeability |
Aqueous Viscosity } { Dissolved Air Diffusion |
Dissolved Salt Diffusion } { Dissolved Oil Diffusion |
Effective Permeability |
Gas Density | Gas Relative Permeability |
Gas Viscosity | Hydraulic Dispersion |
Intrinsic Permeability | NAPL Density |
NAPL Relative Permeability | NAPL Viscosity |
Solute Diffusion | Thermal Conductivity |
Oil Vapor Diffusion | Water Vapor Diffusion }
Interfacial Averaging Scheme Option[b]
{ Harmonic | Geometric | Arithmetic | Upwind }
Format: *Char[a], Char[b],*
Endfor: Number of Interfacial Averaging Variables

Endcard: Solution Control Card

B.24.1 Solution Control Card Examples

Extracted from STOMP1 (Water) input file:
~Solution Control Card
Restart,
Water w/transport vadose courant,
1,
2050,yr,12000,yr,0.01,yr,1.0,yr,1.25,8,1.e-06,
20000,
0,

Extracted from STOMP1 (Water) input file:
~Solution Control Card
Normal,
Water w/ Solute Transport,
1,
0,min,6816.95,min,1,s,20,min,1.25,8,1.e-06,
1000,
0,

Extracted from STOMP1 (Water) input file:
~Solution Control Card
Normal w/ Scaling w/ Inverse,
Water,
1,
0,s,1980000,s,0.1,s,1800,s,1.25,8,1.e-06,
1000,
0,

Extracted from STOMP1 (Water) input file:
~Solution Control Card
Normal,
Water w/TVD transport w/electrolyte courant,0.20,
2,
0,hr,22.2,hr,0.0001,hr,0.1,hr,1.25,8,1.E-6,
22.2,hr,94,hr,0.001,hr,0.1,hr,1.25,8,1.E-6,
100000,
TDS,
Fourth-Order,1/L,1.0,0.000825,,,
Fourth-Order,1/L,1.0,,,,
0,

Extracted from STOMP2 (Wate-Air) input file:
~Solution Control Card
Restart file, TheRestart,
Water-Air Transport,
1,
0,day,200,day,10,s,50,day,1.25,8,1.e-6,
10000,
Constant Aqueous Diffusion, 1.8e-05,cm^2/s,
Constant Gas Diffusion, 1.5e-02,cm^2/s,
,

Solution Control Card Examples (cont'd)

Extracted from STOMP3 (Water-Air-Energy) input file:

```
~Solution Control Card
Restart Mode,1,
Water-Air-Energy,
1,
0,yr,500,yr,1,sec,50,yr,1.25,8,1.e-06,
10000,
Variable,
Variable,
0,
```

Extracted from STOMP3 (Water-Air-Energy) input file:

```
~Solution Control Card
Normal,
Water-Air-Energy,
1,
0,day,876.6,day,10,S,100,day,1.25,16,1.E-06,
1000,
Variable Aqueous Diffusion,
Variable Gas Diffusion,
# Interfacial average defaults
3,
Gas Relative Permeability, Arithmetic,
Aqueous Relative Permeability, Harmonic,
Intrinsic Permeability, Geometric,
```

Extracted from STOMP4 (Water-Oil) input file:

```
~Solution Control Card
Normal,
Water-Oil,
2,
0,s,10,d,1,s,1,d,1.25,8,1.e-6,
5,d,10,d,1,s,1,d,1.25,8,1.e-6,
1000,
Variable Aqueous Diffusion,
0,
```

Extracted from STOMP5 (Water-Air-Oil) input file:

```
#  ------------------------------------------------------------------
~Solution Control Card
#  ------------------------------------------------------------------
Normal,
Water-Oil-Air,,
1,
0,yr,1954,yr,1,yr,200,yr,1.25,16,1.e-6,
,,,,100000,
Variable,
Constant,0.9e-6,m^2/s,0.9e-6,m^2/s,
0,
```

Solution Control Card Examples (cont'd)

Extracted from STOMP11 (Water-Salt) input file:

```
~Solution Control Card
Normal,
Water-Salt w/ Surface Tension Effects,
1,
0,day,1.0,hr,1.e-5,min,10.0,min,1.25,8,1.e-06,
1000,
0,
```

Extracted from STOMP32 (Water-CO2-NaCl) input file:

```
~Solution Control Card
Normal,
H2O-NaCl-CO2,
1,
0,day,14600,day,1,hr,40,day,1.25,16,1.e-06,
10000,
Constant Aqueous Diffusion,0.0,ft^2/day,0.0,ft^2/day,
Constant Aqueous Diffusion,4.0e-3,ft^2/day,4.0e-3,ft^2/day,
0,
```

Extracted from STOMP33 (Water-CO2-NaCl-Energy) input file:

```
~Solution Control Card
Restart File,restart.1644,
H2O-NaCl-CO2-Energy,
1,
,,2.0,hr,1.e-6,hr,2.0,hr,1.25,24,1.e-06,0.6,
100000,
Variable Aqueous Diffusion,
Variable Gas Diffusion,
0,
```

B.25 Source Card

Card Title[a] { ~Source [Card] }
Format: *Char*[a]

Number of Source Domains[a],
Format: Integer[a],

For: Number of Source Domains

If: Operational Mode Option = { **Water** } { **Fluid** }
Source Type Option[a]
{ Aqueous Volumetric [Density] | Aqueous Mass [Density] |
Flow Well[p] [Packer [No Volume]] [No Volume] |
Slug Well[p] [No Volume] | Pulse Well[p] [No Volume] |
Pressure Well[p] [No Volume] |
Solute [Density], Solute Name }
I-Start Index[b], I-End Index[c], J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g], Number of Source Times[h] ,
Format: *Char*[a], *Integer*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f], *Integer*[g], *Integer*[h],

Elseif: Operational Mode Option = { **Water-Air** }
Source Type Option[a]
{ Aqueous Volumetric | Aqueous Mass | Flow Well [p] |
Gas Volumetric [Mass Fraction] [Relative Humidity] |
Gas Mass [Mass Fraction] [Relative Humidity] |
Solute [Density], Solute Name }
I-Start Index[b], I-End Index[c], J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g], Number of Source Times[h] ,
Format: *Char*[a], *Integer*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f], *Integer*[g], *Integer*[h]

Elseif: Operational Mode Option = { **Water-Air-Energy** }
Source Type Option[a]
{ Power [Density] | Aqueous Volumetric | Aqueous Mass |
Gas Volumetric [Mass Fraction] [Relative Humidity] |
Gas Mass [Mass Fraction] [Relative Humidity] |
Root-Water Uptake | Solute [Density], Solute Name }
If: Source Type Option = { Root-Water Uptake }
Plant Name[b], I-Start Index[c], I-End Index[d], J-Start Index[e], J-End Index[f],
K-Start Index[g], K-End Index[h], Number of Source Times[i] ,
Format: *Char*[a], *Char*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f], *Integer*[g], *Integer*[h], *Integer*[i],
Else:
I-Start Index[b], I-End Index[c], J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g], Number of Source Times[h] ,
Format: *Char*[a], *Integer*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f], *Integer*[g], *Integer*[h],
Endif:

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-Oil** }
Source Type Option[a]
{ Aqueous Volumetric | Aqueous Mass |
Flow Well[p] | [[Elevation] [Depth] [Head] Well [p] |

NAPL Volumetric | NAPL Mass |
Solute [Density], Solute Name }
I-Start Index[b], I-End Index[c], J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g], Number of Source Times[h] ,
Format: *Char[a], Integer[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h],*

Elseif: Operational Mode Option = { **Water-Air-Oil** }
Source Type Option[a]
{ Aqueous Volumetric | Aqueous Mass |
NAPL Volumetric | NAPL Mass |
Gas Volumetric [Mass Fraction] [Relative Humidity] |
Gas Mass [Mass Fraction] [Relative Humidity] |
Solute [Density], Solute Name }
I-Start Index[b], I-End Index[c], J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g], Number of Source Times[h] ,
Format: *Char[a], Integer[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h],*

Elseif: Operational Mode Option = { **Water-Air-Oil-Energy** }
Source Type Option[a]
{ Power [Density] | Aqueous Volumetric | Aqueous Mass |
NAPL Volumetric | NAPL Mass] |
Gas Volumetric [Mass Fraction] [Relative Humidity] |
Gas Mass [Mass Fraction] [Relative Humidity] |
Solute [Density], Solute Name }
I-Start Index[b], I-End Index[c], J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g], Number of Source Times[h] ,
Format: *Char[a], Integer[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h],*

Elseif: Operational Mode Option = { **Water-Oil-Dissolved Oil** }
Source Type Option[a]
{ Aqueous Volumetric | Aqueous Mass | Dissolved Oil [Density] |
NAPL Volumetric | NAPL Mass |
Solute [Density], Solute Name }
I-Start Index[b], I-End Index[c], J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g], Number of Source Times[h] ,
Format: *Char[a], Integer[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h],*

Elseif: Operational Mode Option = { **Water-Salt** }
Source Type Option[a]
{ Aqueous Volumetric [Density] | Aqueous Mass [Density] |
Salt [Density] | Solute [Density], Solute Name }
If: Source Type Option = { Aqueous Volumetric [Density] }

Source Card (cont'd)

Dissolved Salt Source Option[b],
{ Dissolved Salt Aqueous Concentration | Dissolved Salt Mass Fraction |
Dissolved Salt Relative Saturation | Null }
I-Start Index[c], I-End Index[d], J-Start Index[e], J-End Index[f],
K-Start Index[g], K-End Index[h], Number of Source Times[i] ,
Format: *Char[a], Char[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i],*
Elseif: Source Type Option = { Aqueous Mass [Density] }
Dissolved Salt Source Option[b],
{ Dissolved Salt Aqueous Concentration | Dissolved Salt Mass Fraction |

Dissolved Salt Relative Saturation | Null }
I-Start Index[c], I-End Index[d], J-Start Index[e], J-End Index[f],
K-Start Index[g], K-End Index[h], Number of Source Times[i] ,
Format: *Char*[a], *Char*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f], *Integer*[g], *Integer*[h], *Integer*[i],
Endif:

Elseif: Operational Mode Option = { **Water-Air-Salt** }
Source Type Option[a]
{ Aqueous Volumetric | Aqueous Mass |
Gas Volumetric | Gas Mass |
Salt [Density] | Solute [Density], Solute Name }
If: Source Type Option = { Aqueous Volumetric }
Dissolved Salt Source Option[b],
{ Dissolved Salt Aqueous Concentration | Dissolved Salt Mass Fraction |
Dissolved Salt Relative Saturation | Null }
I-Start Index[c], I-End Index[d], J-Start Index[e], J-End Index[f],
K-Start Index[g], K-End Index[h], Number of Source Times[i] ,
Format: *Char*[a], *Char*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f], *Integer*[g], *Integer*[h], *Integer*[i],
Elseif: Source Type Option = { Aqueous Mass }
Dissolved Salt Source Option[b],
{ Dissolved Salt Aqueous Concentration | Dissolved Salt Mass Fraction |
Dissolved Salt Relative Saturation | Null }
I-Start Index[c], I-End Index[d], J-Start Index[e], J-End Index[f],
K-Start Index[g], K-End Index[h], Number of Source Times[i] ,
Format: *Char*[a], *Char*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f], *Integer*[g], *Integer*[h], *Integer*[i],
Elseif: Source Type Option = { Gas Volumetric [Mass Fraction] [Relative Humidity] }
I-Start Index[b], I-End Index[c], J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g], Number of Source Times[h] ,
Format: *Char*[a], *Integer*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f], *Integer*[g], *Integer*[h],
Elseif: Source Type Option = { Gas Mass [Mass Fraction] [Relative Humidity] }
I-Start Index[b], I-End Index[c], J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g], Number of Source Times[h] ,
Format: *Char*[a], *Integer*[b], *Integer*[c], *Integer*[d], *Integer*[e], *Integer*[f], *Integer*[g], *Integer*[h],
Endif:

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Salt-Energy** }
Source Type Option[a]
{ Power [Density] | Aqueous Volumetric |
Aqueous Mass | Gas Volumetric [Mass Fraction] [Relative Humidity] |
Gas Mass [Mass Fraction] [Relative Humidity] |
Salt [Density] | Solute [Density], Solute Name }
If: Source Type Option = { Aqueous Volumetric }
Dissolved Salt Source Option[b],
{ Dissolved Salt Aqueous Concentration | Dissolved Salt Mass Fraction |
Dissolved Salt Relative Saturation | Null }
I-Start Index[c], I-End Index[d], J-Start Index[e], J-End Index[f],
K-Start Index[g], K-End Index[h], Number of Source Times[i] ,
Format: *Char[a], Char[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i],,*
Elseif: Source Type Option = { Aqueous Mass }
Dissolved Salt Source Option[b],
{ Dissolved Salt Aqueous Concentration | Dissolved Salt Mass Fraction |
Dissolved Salt Relative Saturation | Null }
I-Start Index[c], I-End Index[d], J-Start Index[e], J-End Index[f],
K-Start Index[g], K-End Index[h], Number of Source Times[i] ,
Format: *Char[a], Char[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i],*
Elseif: Source Type Option = { Gas Volumetric [Mass Fraction] [Relative Humidity] }
I-Start Index[b], I-End Index[c], J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g], Number of Source Times[h] ,
Format: *Char[a], Integer[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h],*
Elseif: Source Type Option = { Gas Mass [Mass Fraction] [Relative Humidity] }
I-Start Index[b], I-End Index[c], J-Start Index[d], J-End Index[e],
K-Start Index[f], K-End Index[g], Number of Source Times[h] ,
Format: *Char[a], Integer[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h],*
Endif:

Elseif: Operational Mode Option = { **Water-CO2-NaCl** }
Source Type Option[a]
{ Aqueous Volumetric | Aqueous Mass |
Gas Volumetric | Gas Mass |
Salt [Density] | Solute [Density], Solute Name }
If: Source Type Option = { Aqueous Volumetric }
Dissolved Salt Source Option[b],
{ Dissolved Salt Aqueous Concentration | Dissolved Salt Mass Fraction |
Dissolved Salt Relative Saturation | Null }
Dissolved CO_2 Source Option[c],
{ Dissolved CO_2 Aqueous Concentration | Dissolved CO_2 Mass Fraction |
Dissolved CO_2 Relative Saturation | Null }
I-Start Index[d], I-End Index[e], J-Start Index[f], J-End Index[g],
K-Start Index[h], K-End Index[i], Number of Source Times[j] ,
Format: *Char[a], Char[b], Char[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i], Integer[j],*

Source Card (cont'd)

Elseif: Source Type Option = { Aqueous Mass }
Dissolved Salt Source Option[b],
{ Dissolved Salt Aqueous Concentration | Dissolved Salt Mass Fraction |
Dissolved Salt Relative Saturation | Null }

Dissolved CO_2 Source Option[c],
{ Dissolved CO_2 Aqueous Concentration | Dissolved CO_2 Mass Fraction |
Dissolved CO_2 Relative Saturation | Null }
I-Start Index[d], I-End Index[e], J-Start Index[f], J-End Index[g],
K-Start Index[h], K-End Index[i], Number of Source Times[j] ,
Format: *Char[a], Char[b], Char[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i], Integer[j],*
Elseif: Source Type Option = { Gas Volumetric }
Water Vapor Source Option[b],
{ Water Vapor Gas Relative Humidity | Water Vapor Gas Mass Fraction | Null }
I-Start Index[c], I-End Index[d], J-Start Index[e], J-End Index[f],
K-Start Index[g], K-End Index[h], Number of Source Times[i] ,
Format: *Char[a], Char[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i],*
Elseif: Source Type Option = { Gas Mass }
Water Vapor Source Option[b],
{ Water Vapor Gas Relative Humidity | Water Vapor Gas Mass Fraction | Null }
I-Start Index[c], I-End Index[d], J-Start Index[e], J-End Index[f],
K-Start Index[g], K-End Index[h], Number of Source Times[i] ,
Format: *Char[a], Char[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i],*
Endif:

Elseif: Operational Mode Option = { **Water-CO2-NaCl-Energy** }
Source Type Option[a]
{ Power [Density] | Aqueous Volumetric | Aqueous Mass |
Gas Volumetric | Gas Mass |
Salt [Density] | Solute [Density], Solute Name }
If: Source Type Option = { Aqueous Volumetric }
Dissolved Salt Source Option[b],
{ Dissolved Salt Aqueous Concentration | Dissolved Salt Mass Fraction |
Dissolved Salt Relative Saturation | Null }
Dissolved CO_2 Source Option[c],
{ Dissolved CO_2 Aqueous Concentration | Dissolved CO_2Mass Fraction |
Dissolved CO_2 Relative Saturation | Null }
I-Start Index[d], I-End Index[e], J-Start Index[f], J-End Index[g],
K-Start Index[h], K-End Index[i], Number of Source Times[j] ,
Format: *Char[a], Char[b], Char[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i], Integer[j],*
Elseif: Source Type Option = { Aqueous Mass }
Dissolved Salt Source Option[b],
{ Dissolved Salt Aqueous Concentration | Dissolved Salt Mass Fraction |
Dissolved Salt Relative Saturation | Null }
Dissolved CO_2 Source Option[c],
{ Dissolved CO_2 Aqueous Concentration | Dissolved CO_2 Mass Fraction |
Dissolved CO_2 Relative Saturation | Null }

Source Card (cont'd)

I-Start Index[d], I-End Index[e], J-Start Index[f], J-End Index[g],
K-Start Index[h], K-End Index[i], Number of Source Times[j] ,
Format: *Char[a], Char[b], Char[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i], Integer[j],*
Elseif: Source Type Option = { Gas Volumetric }
Water Vapor Source Option[b],
{ Water Vapor Gas Relative Humidity | Water Vapor Gas Mass Fraction | Null }
I-Start Index[c], I-End Index[d], J-Start Index[e], J-End Index[f],
K-Start Index[g], K-End Index[h], Number of Source Times[i] ,
Format: *Char[a], Char[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i],*

Elseif: Source Type Option = { Gas Mass }
Water Vapor Source Option[b],
{ Water Vapor Gas Relative Humidity | Water Vapor Gas Mass Fraction | Null }
I-Start Index[c], I-End Index[d], J-Start Index[e], J-End Index[f],
K-Start Index[g], K-End Index[h], Number of Source Times[i] ,
Format: *Char[a], Char[b], Integer[c], Integer[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i],*
Endif:

Endif:

Source Card (cont'd)

If: Operational Mode Option = { **Water** } { **Fluid** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type Option = { Aqueous Volumetric }
Aqueous Volumetric Rate[c], Units[d] (m^3/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Aqueous Mass }
Aqueous Mass Rate[c], Units[d] (kg/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Aqueous Volumetric Density }
Aqueous Volumetric Density Rate[c], Units[d] (m^3/s m^3),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Aqueous Mass Density }
Aqueous Mass Density Rate[c], Units[d] (kg/s m^3),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Slug Well [No Volume] } or
{ Pressure Well [No Volume] }
Well Bottom Pressure[c], Units[d] (Pa),
Borehole Radius[e], Units[f] (m),
Casing Radius[g], Units[h] (m),
For: Number of Solutes
Solute Aqueous Conc[i], Units[j] (1/m^3),
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], < Real[l], Char[j], >*
Elseif: Source Type Option = { Pulse Well [No Volume] }
Well Bottom Pressure[c], Units[d] (Pa),
Borehole Radius[e], Units[f] (m),
Casing Radius[g], Units[h] (m),
System Volume[i], Units[j] (m^3),
System Compressibility[k], Units[l] (1/Pa),
For: Number of Solutes
Solute Aqueous Conc[m], Units[n] (1/m^3),
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Char[j], Real[k], Char[l], < Real[m], Char[n], >*
Elseif: Source Type Option = { Flow Well [No Volume] }
Aqueous Volumetric Rate[c], Units[d] (m^3/s),
Borehole Radius[e], Units[f] (m),
Casing Radius[g], Units[h] (m),
Well Bottom Pressure Constraint[i], Units[j] (Pa),
For: Number of Solutes
Solute Aqueous Conc[k], Units[l] (1/m^3),
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Char[j], < Real[k], Char[l], >*

Source Card (cont'd)

Elseif: Source Type Option = { Flow Well Packer [No Volume] }
Aqueous Volumetric Rate[c], Units[d] (m^3/s),
Borehole Radius[e], Units[f] (m),
Casing Radius[g], Units[h] (m),

System Volume[i], Units[j] (m^3),
For: Number of Solutes
Solute Aqueous Conc[m], Units[n] (1/m^3),
Endfor: Number of Solutes
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Char[j], Real[k], Char[l], < Real[m], Char[n], >*
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] (1/m^3 s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:
Endfor: Number of Source Times

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type Option = { Aqueous Volumetric }
Pressure[c], Units[d] (Pa), Aqueous Volumetric Rate[e], Units[f] (m^3/s),
Dissolved-Air Relative Saturation[g],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*
Elseif: Source Type Option = { Aqueous Mass }
Pressure[c], Units[d] (Pa), Aqueous Mass Rate[e], Units[f] (kg/s),
Dissolved-Air Relative Saturation[g],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*
Elseif: Source Type Option = { Gas Mass w/ Mass Fraction }
Pressure[c], Units[d] (Pa), Gas Mass Rate[e], Units[f] (kg/s),
Water Vapor Mass Fraction[g],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*
Elseif: Source Type Option = { Gas Mass w/ Relative Humidity }
Pressure[c], Units[d] (Pa), Gas Mass Rate[e], Units[f] (kg/s),
Water Vapor Relative Humidity[g],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*
Elseif: Source Type Option = { Gas Volumetric w/ Mass Fraction }
Pressure[c], Units[d] (Pa), Gas Volumetric Rate[e], Units[f] (m^3/s),
Water Vapor Mass Fraction[g],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*
Elseif: Source Type Option = { Gas Volumetric w/ Relative Humidity }
Pressure[c], Units[d] (Pa), Gas Volumetric Rate[e], Units[f] (m^3/s),
Water Vapor Relative Humidity[g],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*
Elseif: Source Type Option = { Flow Well }
Aqueous Volumetric Rate[c], Units[d] (m^3/s), Borehole Radius[e], Units[f] (m),
Water Level Constraint[g], Units[h] (m), Water Level Control Parameter[i], Units[j] (m),
Dissolved-Air Relative Saturation[k],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Char[j], Real[k],*
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] (1/s m^3),
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:

Endfor: Number of Source Times

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Energy** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type Option = { Power }
Power[c], Units[d] (W),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Power Density }
Power Density[c], Units[d] (W/m^3),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Root-Water Uptake }
Source Plant Areal Density[c], Units[d] (plants/m^2),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Aqueous Volumetric }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Aqueous Volumetric Rate[g], Units[h] (m^3/s), Dissolved Relative Saturation[i],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i],*
Elseif: Source Type Option = { Aqueous Mass }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Aqueous Mass Rate[g], Units[h] (kg/s), Dissolved Air Relative Saturation[i],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i],*
Elseif: Source Type Option = { Gas Mass w/ Mass Fraction }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Gas Mass Density Rate[g], Units[h] (kg/s), Water Vapor Mass Fraction[i],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i]*
Elseif: Source Type Option = { Gas Mass w/ Relative Humidity }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Gas Mass Density Rate[g], Units[h] (kg/s), Water Vapor Relative Humidity[i],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i]*
Elseif: Source Type Option = { Gas Volumetric w/ Mass Fraction }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Gas Volumetric Density Rate[g], Units[h] (m^3/s), Water Vapor Mass Fraction[i],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i]*
Elseif: Source Type Option = { Gas Volumetric w/ Relative Humidity }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Gas Volumetric Rate[g], Units[h] (m^3/s), Water Vapor Relative Humidity[i],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i]*
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s),
Format: *Real[a], Char[b], Real[c], Char[d]*
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] (1/s m^3),
Format: *Real[a], Char[b], Real[c], Char[d]*
Endif:

Endfor: Number of Source Times

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-Oil** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type Option = { Aqueous Volumetric }

Aqueous Volumetric Rate[c], Units[d] (m^3/s), Dissolved Oil Mass Fraction[e],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e],*
Elseif: Source Type Option = { Aqueous Mass }
Aqueous Mass Rate[c], Units[d] (kg/s), Dissolved Oil Mass Fraction[e],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e],*
Elseif: Source Type Option = { NAPL Volumetric }
NAPL Volumetric Rate[c], Units[d] (m^3/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { NAPL Mass }
NAPL Mass Rate[c], Units[d] (kg/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Flow Well }
Aqueous Volumetric Rate[c], Units[d] (m^3/s), Borehole Radius[e], Units[f] (m),
Liquid Level Constraint[g], Units[h] (m), Dissolved-Oil Relative Saturation[i],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i],*
Elseif: Source Type Option = { [Elevation] [Head] [Depth] Well }
Liquid Elevation in Well[c], Units[d] (m),
Borehole Radius[e], Units[f] (m),
Dissolved-Oil Relative Saturation[g],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] (1/s m^3),
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:

Endfor: Number of Source Times

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Oil** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type= { Aqueous Volumetric }
Aqueous Volumetric Rate[c], Units[d] (m^3/s),
Dissolved Air Mass Fraction[g], Dissolved Oil Mass Fraction[f],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Elseif: Source Type Option = { Aqueous Mass }
Aqueous Mass Rate[c], Units[d] (kg/s),
Dissolved Air Mass Fraction[e], Dissolved Oil Mass Fraction[f],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Elseif: Source Type Option = { Gas Mass w/ Mass Fraction }
Gas Mass Rate[c], Units[c] (kg/s),
Water Vapor Mass Fraction[e], Oil Vapor Mass Fraction[f],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Elseif: Source Type Option = { Gas Mass w/ Relative Humidity }
Gas Mass Rate[c], Units[d] (kg/s),
Water Vapor Relative Humidity[e], Oil Vapor Relative Humidity[f],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Elseif: Source Type Option = { Gas Volumetric w/ Mass Fraction }
Gas Volumetric Rate[c], Units[d] (m^3/s),
Water Vapor Mass Fraction[e], Oil Vapor Mass Fraction[f],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Elseif: Source Type Option = { Gas Volumetric w/ Relative Humidity }
Gas Volumetric Rate[c], Units[d] (m^3/s),
Water Vapor Relative Humidity[e], Oil Vapor Relative Humidity[f],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Elseif: Source Type Option = { NAPL Volumetric }
NAPL Volumetric Rate[c], Units[d] (m^3/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { NAPL Mass }
NAPL Mass Rate[c], Units[d] (kg/s),
NAPL Volumetric Rate[c], Units[d] (m^3/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s) ,
NAPL Volumetric Rate[c], Units[d] (m^3/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] (1/m^3 s) ,
NAPL Volumetric Rate[c], Units[d] (m^3/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:

Endfor: Number of Source Times

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Oil-Energy** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type Option = { Power }

Power[c], Units[d] (W),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Power Density }
Power Density[c], Units[d] (W/m^3),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type= { Aqueous Volumetric }
Temperature[c], Units[d], Pressure[e], Units[f] (Pa),
Aqueous Volumetric Rate[g], Units[h] (m^3/s),
Dissolved Air Mass Fraction[i], Dissolved Oil Mass Fraction[j],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Elseif: Source Type Option = { Aqueous Mass }
Temperature[c], Units[d], Pressure[e], Units[f] (Pa),
Aqueous Mass Rate[g], Units[h] (kg/s),
Dissolved Air Mass Fraction[i], Dissolved Oil Mass Fraction[j],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Elseif: Source Type Option = { Gas Mass w/ Mass Fraction }
Temperature[c], Units[d], Pressure[e], Units[f] (Pa),
Gas Mass Rate[g], Units[h] (kg/s),
Water Vapor Mass Fraction[i], Oil Vapor Mass Fraction[j],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Elseif: Source Type Option = { Gas Mass w/ Relative Humidity }
Temperature[c], Units[d], Pressure[e], Units[f] (Pa),
Gas Mass Rate[g], Units[h] (kg/s),
Water Vapor Relative Humidity[i], Oil Vapor Relative Humidity[j],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Elseif: Source Type Option = { Gas Volumetric w/ Mass Fraction }
Temperature[c], Units[d], Pressure[e], Units[f] (Pa),
Gas Volumetric Rate[g], Units[h] (m^3/s),
Water Vapor Mass Fraction[i], Oil Vapor Mass Fraction[j],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Elseif: Source Type Option = { Gas Volumetric w/ Relative Humidity }
Temperature[c], Units[d], Pressure[e], Units[f] (Pa),
Gas Volumetric Rate[g], Units[h] (m^3/s),
Water Vapor Relative Humidity[i], Oil Vapor Relative Humidity[j],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Elseif: Source Type Option = { NAPL Volumetric }
Temperature[c], Units[d], Pressure[e], Units[f] (Pa),
NAPL Volumetric Rate[g], Units[h] (m^3/s),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h],*
Elseif: Source Type Option = { NAPL Mass }
Temperature[c], Units[d], Pressure[e], Units[f] (Pa), NAPL Mass Rate[g], Units[h] (kg/s),

Source Card (cont'd)

Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h],*
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] ($1/m^3$ s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:

Endfor: Number of Source Times

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-Oil-Dissolved Oil** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type Option = { Aqueous Volumetric }
Aqueous Volumetric Rate[c], Units[d] (m^3/s),
Dissolved Oil Conc[e], Units[f] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Elseif: Source Type Option = { Aqueous Mass }
Aqueous Mass Rate[c], Units[d] (kg/s),
Dissolved Oil Conc[e], Units[f] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Elseif: Source Type Option = { Dissolved Oil }
Dissolved Oil Mass Rate[c], Units[d] (kg/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Dissolved Oil Density }
Dissolved Oil Mass Density Rate[c], Units[d] (kg/s m^3),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { NAPL Volumetric }
NAPL Volumetric Rate[c], Units[d] (m^3/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { NAPL Mass }
NAPL Mass Rate[c], Units[d] (kg/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] (1/m^3 s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:
Endfor: Number of Source Times

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-Salt** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type Option = { Aqueous Volumetric }
Aqueous Volumetric Rate[c], Units[d] (m^3/s),
If: Dissolved Salt Source Option = { Aqueous Concentration }
Dissolved Salt Aqueous Concentration[e], Units[f] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[e],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e],*
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[e],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e],*
Else:
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:
Elseif: Source Type Option = { Aqueous Volumetric Density }
Aqueous Volumetric Density Rate[c], Units[d] (m^3/s m^3),
If: Dissolved Salt Source Option = { Aqueous Concentration }
Dissolved Salt Aqueous Concentration[e], Units[f] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[e],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e],*
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[e],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e],*
Else:
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:
Elseif: Source Type Option = { Aqueous Mass }
Aqueous Mass Rate[c], Units[d] (kg/s),
If: Dissolved Salt Source Option = { Aqueous Concentration }
Dissolved Salt Aqueous Concentration[e], Units[f] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[e],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e],*
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[e],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e],*
Else:
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:

Source Card (cont'd)

Elseif: Source Type Option = { Aqueous Mass Density }
Aqueous Mass Density Rate[c], Units[d] (kg/s m^3),
If: Dissolved Salt Source Option = { Aqueous Concentration }
Dissolved Salt Aqueous Concentration[e], Units[f] (kg/m^3),

Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f],
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[e],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e],
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[e],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e],
Else:
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Endif:
Elseif: Source Type Option = { Salt }
Salt Mass Rate[c], Units[d] (kg/s)
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Elseif: Source Type Option = { Salt Density }
Salt Mass Density Rate[c], Units[d] (kg/m^3 s)
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] (1/s m^3),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Endif:

Endfor: Number of Source Times

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Salt** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type Option = { Aqueous Volumetric }
Pressure[c], Units[d] (Pa), Aqueous Volumetric Rate[e], Units[f] (m^3/s),
Dissolved-Air Relative Saturation[g],
If: Dissolved Salt Source Option = { Aqueous Concentration }
Dissolved Salt Aqueous Concentration[h], Units[i] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h],Char[i],*
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h],*
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h],*
Else:
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*
Endif:
Elseif: Source Type Option = { Aqueous Mass }
Pressure[c], Units[d] (Pa), Aqueous Mass Rate[e], Units[f] (kg/s),
Dissolved-Air Relative Saturation[g],
If: Dissolved Salt Source Option = { Aqueous Concentration }
Dissolved Salt Aqueous Concentration[h], Units[i] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h],Char[i],*
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h],*
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h],*
Else:
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*
Endif:
Elseif: Source Type Option = { Gas Mass w/ Mass Fraction }
Pressure[c], Units[d] (Pa), Gas Mass Rate[e], Units[f] (kg/s),
Water Vapor Mass Fraction[g],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*
Elseif: Source Type Option = { Gas Mass w/ Relative Humidity }
Pressure[c], Units[d] (Pa), Gas Mass Rate[e], Units[f] (kg/s),
Water Vapor Relative Humidity[g],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*
Elseif: Source Type Option = { Gas Volumetric w/ Mass Fraction }
Pressure[c], Units[d] (Pa), Gas Volumetric Rate[e], Units[f] (m^3/s),
Water Vapor Mass Fraction[g],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*

Source Card (cont'd)

Elseif: Source Type Option = { Gas Volumetric w/ Relative Humidity }
Pressure[c], Units[d] (Pa), Gas Volumetric Rate[e], Units[f] (m^3/s),
Water Vapor Relative Humidity[g],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g],*

Elseif: Source Type Option = { Salt }
Salt Mass Rate[c], Units[d] (kg/s)
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Salt Density }
Salt Mass Density Rate[c], Units[d] (kg/m^3 s)
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] (1/s m^3),
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:

Endfor: Number of Source Times

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-Air-Salt-Energy** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type Option = { Power }
Power[c], Units[d] (W),
Format: *Real[a], Char[b], Real[c], Char[d],Real[e],*
Elseif: Source Type Option = { Power Density }
Power Density[c], Units[d] (W/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e],*
Elseif: Source Type Option = { Aqueous Volumetric }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Aqueous Volumetric Rate[g], Units[h] (m^3/s), Dissolved Air Mass Fraction[i],
If: Dissolved Salt Source Option = { Aqueous Concentration }
Dissolved Salt Aqueous Concentration[j], Units[k] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j], Char[k],*
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Else:
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i],*
Endif:
Elseif: Source Type Option = { Aqueous Mass }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Aqueous Mass Rate[g], Units[h] (kg/s), Dissolved Air Mass Fraction[i],
If: Dissolved Salt Source Option = { Aqueous Concentration }
Dissolved Salt Aqueous Concentration[j], Units[k] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j], Char[k],*
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Else:
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i],*
Endif:
Elseif: Source Type Option = { Gas Mass w/ Mass Fraction }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Gas Mass Density Rate[g], Units[h] (kg/s), Water Vapor Mass Fraction[i],
Format: *Real[a], Char[b], Real[c], Char[d],Real[e], Char[f], Real[g], Char[h], Real[i],*
Elseif: Source Type Option = { Gas Mass w/ Relative Humidity }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),

Source Card (cont'd)

Gas Mass Density Rate[g], Units[h] (kg/s), Water Vapor Relative Humidity[i],
Format: *Real[a], Char[b], Real[c], Char[d],Real[e], Char[f], Real[g], Char[h], Real[i],*
Elseif: Source Type Option = { Gas Volumetric w/ Mass Fraction }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),

Gas Volumetric Density Rate[g], Units[h] (m^3/s), Water Vapor Mass Fraction[i],
Format: *Real[a], Char[b], Real[c], Char[d],Real[e], Char[f], Real[s], Char[h], Real[i],*
Elseif: Source Type Option = { Gas Volumetric w/ Relative Humidity }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Gas Volumetric Rate[g], Units[h] (m^3/s), Water Vapor Relative Humidity[i],
Format: *Real[a], Char[b], Real[c], Char[d],Real[e], Char[f], Real[s], Char[h], Real[i],*
Elseif: Source Type Option = { Salt }
Salt Mass Rate[c], Units[d] (kg/s)
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Salt Density }
Salt Mass Density Rate[c], Units[d] (kg/m^3 s)
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] (1/s m^3),
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:
Format: *Real[a], Char[b], Real[c], Char[d],Real[e], Char[f], Real[s], Char[h], Real[i],*

Endfor: Number of Source Times

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-CO2-NaCl** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type Option = { Aqueous Volumetric }
Pressure[c], Units[d] (Pa), Aqueous Volumetric Rate[e], Units[f] (m^3/s),
If: Dissolved Salt Source Option = { Aqueous Concentration }
Dissolved Salt Aqueous Concentration[g], Units[h] (kg/m^3),
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[i], Units[j] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Char[j],*
Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[i],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i],*
Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[i],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i],*
Endif:
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[g],
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[h], Units[i] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h], Char[i],*
Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h],*
Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h],*
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[g],
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[h], Units[i] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h], Char[i],*
Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h],*
Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[h],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Real[h],*
Else:
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f],*
Endif:
Elseif: Source Type Option = { Aqueous Mass }
Pressure[c], Units[d] (Pa), Aqueous Mass Rate[e], Units[f] (kg/s),
If: Dissolved Salt Source Option = { Aqueous Concentration }

Source Card (cont'd)

Dissolved Salt Aqueous Concentration[g], Units[h] (kg/m^3),
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[i], Units[j] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Char[j],*

Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[i],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i],
Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[i],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i],
Endif:
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[g],
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[h], Units[i] (kg/m^3),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Real*[h], *Char*[i],
Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[h],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Real*[h],
Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[h],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Real*[h],
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[g],
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[h], Units[i] (kg/m^3),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Real*[h], *Char*[i],
Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[h],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Real*[h],
Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[h],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Real*[h],
Else:
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f],
Endif:
Elseif: Source Type Option = { Gas Mass }
Pressure[c], Units[d] (Pa), Gas Mass Rate[e], Units[f] (kg/s),
If: Water Vapor Source Option = { Mass Fraction }
Water Vapor Mass Fraction[g],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g],
Elseif: Water Vapor Source Option = { Relative Saturation }
Water Vapor Relative Humidity[g],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g],
Else:

Source Card (cont'd)

Null[g],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g],
Endif:
Elseif: Source Type Option = { Gas Volumetric }
Pressure[c], Units[d] (Pa), Gas Volumetric Rate[e], Units[f] (m^3/s),
If: Water Vapor Source Option = { Mass Fraction }
Water Vapor Mass Fraction[g],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g],
Elseif: Water Vapor Source Option = { Relative Saturation }
Water Vapor Relative Humidity[g],

Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g],
Else:
Null[g],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g],
Endif:
Elseif: Source Type Option = { Salt }
Salt Mass Rate[c], Units[d] (kg/s)
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Elseif: Source Type Option = { Salt Density }
Salt Mass Density Rate[c], Units[d] (kg/m^3 s)
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] (1/s m^3),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Endif:

Endfor: Number of Source Times

Source Card (cont'd)

Elseif: Operational Mode Option = { **Water-CO2-NaCl-Energy** }
For: Number of Source Times
Source Time[a], Units[b] (s),
If: Source Type Option = { Power }
Power[c], Units[d] (W),
Format: *Real[a], Char[b], Real[c], Char[d],*
Elseif: Source Type Option = { Power Density }
Power Density[c], Units[d] (W/m^3),
Format: *Real[a], Char[b], Real[c], Char[d],*
Endif:
If: Source Type Option = { Aqueous Volumetric }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Aqueous Volumetric Rate[g], Units[h] (m^3/s),
If: Dissolved Salt Source Option = { Aqueous Concentration }
Dissolved Salt Aqueous Concentration[i], Units[j] (kg/m^3),
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[k], Units[l] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Char[j], Real[k], Char[l],*
Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[k],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Char[j], Real[k],*
Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[k],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Char[j], Real[k],*
Endif:
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[i],
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[j], Units[k] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j], Char[k],*
Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[j],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[j],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h],Real[i], Real[j],*
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[i],
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[j], Units[k] (kg/m^3),
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j], Char[k],*
Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[j],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*

Source Card (cont'd)

Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[j],
Format: *Real[a], Char[b], Real[c], Char[d], Real[e], Char[f], Real[g], Char[h], Real[i], Real[j],*
Else:

Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f],
Endif:
Elseif: Source Type Option = { Aqueous Mass }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Aqueous Volumetric Rate[g], Units[h] (m^3/s),
If: Dissolved Salt Source Option = { Aqueous Concentration }
Dissolved Salt Aqueous Concentration[i], Units[j] (kg/m^3),
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[k], Units[l] (kg/m^3),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Char*[j], *Real*[k], *Char*[l],
Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[k],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Char*[j], *Real*[k],
Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[k],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Char*[j], *Real*[k],
Endif:
Elseif: Dissolved Salt Source Option = { Mass Fraction }
Dissolved Salt Mass Fraction[i],
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[j], Units[k] (kg/m^3),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Real*[j], *Real*[k],
Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[j],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Real*[j],
Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[j],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Real*[j],
Elseif: Dissolved Salt Source Option = { Relative Saturation }
Dissolved Relative Saturation[i],
If: Dissolved CO_2 Source Option = { Aqueous Concentration }
Dissolved CO_2 Aqueous Concentration[j], Units[k] (kg/m^3),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Real*[j], *Real*[k],
Elseif: Dissolved CO_2 Source Option = { Mass Fraction }
Dissolved CO_2 Mass Fraction[j],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Real*[j],
Elseif: Dissolved CO_2 Source Option = { Relative Saturation }
Dissolved CO_2 Relative Saturation[j],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Real*[j],
Else:

Source Card (cont'd)

Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f],
Endif:
Elseif: Source Type Option = { Gas Mass }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Aqueous Volumetric Rate[g], Units[h] (m^3/s),
If: Water Vapor Source Option = { Mass Fraction }
Water Vapor Mass Fraction[i],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i],
Elseif: Water Vapor Source Option = { Relative Saturation }
Water Vapor Relative Humidity[i],

Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i],
Else:
Null[i],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i],
Endif:
Elseif: Source Type Option = { Gas Volumetric }
Temperature[c], Units[d] (C), Pressure[e], Units[f] (Pa),
Aqueous Volumetric Rate[g], Units[h] (m^3/s),
If: Water Vapor Source Option = { Mass Fraction }
Water Vapor Mass Fraction[i],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i],
Elseif: Water Vapor Source Option = { Relative Saturation }
Water Vapor Relative Humidity[i],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i],
Else:
Null[i],
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i],
Endif:
Elseif: Source Type Option = { Salt }
Salt Mass Rate[c], Units[d] (kg/s)
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Elseif: Source Type Option = { Salt Density }
Salt Mass Density Rate[c], Units[d] (kg/m^3 s)
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Elseif: Source Type Option = { Solute }
Solute Rate[c], Units[d] (1/s),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Elseif: Source Type Option = { Solute Density }
Solute Density Rate[c], Units[d] (1/s m^3),
Format: *Real*[a], *Char*[b], *Real*[c], *Char*[d],
Endif:
Endfor: Number of Source Times
Endfor: Number of Source Domains

Endcard: Source Card

B.25.1 Source Card Examples

Extracted from STOMP1 (Water) input file:
~Source Card
1,
Aqueous Volumetric,2,2,1,1,71,71,7,
0,min,0.8320722,gal/min,
66.046844,min,1.0441576,gal/min,
135.56322,min,1.0438042,gal/min,
205.09055,min,0.98627841,gal/min,
274.60406,min,0.99331003,gal/min,
344.13425,min,0.91488999,gal/min,
413.64774,min,1.0228366,gal/min,

Extracted from STOMP2 (Water-Air) input file:
~Source Card
1,
Gas Volumetric w/ Mass Fraction,29,31,1,1,7,9,2,
0,min,101326.2,Pa,0.555556,l/min,,
10,min,101326.2,Pa,5,l/min,,

Extracted from STOMP3 (Water-Air-Energy) input file:
~Source Card
2,
Root-Water Uptake,Sage Brush,1,1,1,1,7,10,1,
0.0,yr,0.1,plant/m^2,

Extracted from STOMP11 (Water-Salt) input file:
~Source Card
2,
Aqueous Volumetric, Dissolved Salt Aqueous Concentration,30,30,1,1,119,120,2,
0.0,min,1.0,ml/min,312.6,kg/m^3,
5.0,min,1.0,ml/min,312.6,kg/m^3,
Aqueous Volumetric,,60,60,1,1,119,120,2,
0.0,min,1.0,ml/min,
5.0,min,1.0,ml/min,

Extracted from STOMP33 (Water-CO2-NaCl) input file:
~Source Card
1,
Gas Mass Rate,Water-Vapor Mass Fraction,1,1,1,1,1,9,2,
0,day,,,0.352413,kg/s,0.0,
7300,day,,,0.352413,kg/s,0.0,

Extracted from STOMP33 (Water-CO2-NaCl-Energy) input file:
~Source Card
1,
Gas Volumetric Rate,Water-Vapor Mass Fraction,3,3,1,1,3,3,2,
0,hr,25.0,C,138.0,bar,4.2908,cm^3/min,0.0,
0.2,hr,25.0,C,138.0,bar,4.2908,cm^3/min,0.0,

B.26 Surface Flux Card

Card Title[a] { ~Surface [Flux Card] }
Format: *Char*[a]

Number of Surface Flux Inputs[a]
Format: Integer[a],

Note: The number of suface flux inputs may be written to one or more user specified files. For each user specified file, the following input line has to be specified. An example has been included in the Surface Flux Card Examples section. The sum of the "Number of Surface Flux Inputs in File" has to be equal to the "Number of Surface Flux Inputs".

Number of Surface Flux Inputs in File[a], Filename[b],
Format: Integer[a], Char[b],

For: Number of Surface Flux Inputs:

If: Operational Mode Option = { **Water** } { **Fluid** }
Surface Flux Type Option[a],
{ Aqueous Volumetric Flux | Aqueous Mass Flux | Solute Flux, Solute Name }

Elseif: Operation Mode Option = { **Water-Air** }
Surface Flux Type Option[a],
{ Aqueous Volumetric Flux | Aqueous Mass Flux | Gas Volumetric Flux |
Gas Mass Flux | Solute Flux, Solute Name }

Elseif: Operation Mode Option = { **Water-Air-Energy** }
Surface Flux Type Option[a],
{ Aqueous Volumetric Flux | Aqueous Mass Flux | Gas Volumetric Flux |
Gas Mass Flux | Gas Advective Heat Flux | Gas Advective Water Mass Flux |
Gas Advective Air Mass Flux | Gas Diffusive Heat Flux |
Gas Diffusive Water Mass Flux | Gas Diffusive Air Mass Flux |
Solute Flux, Solute Name | Heat Flux | Condensate Water Mass }

Elseif: Operation Mode Option = { **Water-Oil** } { **Water-Oil-Dissolved Oil** }
Surface Flux Type Option[a],
{ Aqueous Volumetric Flux | Aqueous Mass Flux | NAPL Volumetric Flux |
NAPL Mass Flux | Dissolved Oil Flux | Water Mass Flux |
Solute Flux, Solute Name }

Elseif: Operation Mode Option = { **Water-Air-Oil** }
Surface Flux Type Option[a],
{ Aqueous Volumetric Flux | Aqueous Mass Flux | Gas Volumetric Flux |
Gas Mass Flux | NAPL Volumetric Flux | NAPL Mass Flux |
Gas-Advective Oil Mass | Gas-Diffusive Oil Mass | Gas-Total Oil Mass |

Surface Flux Card (cont'd)

Dissolved-Oil | Solute Flux, Solute Name }

Elseif: Operation Mode Option = { **Water-Air-Oil-Energy** }
Surface Flux Type Option[a],

{ Aqueous Volumetric Flux | Aqueous Mass Flux | Gas Volumetric Flux |
Gas Mass Flux | NAPL Volumetric Flux | NAPL Mass Flux |
Heat Flux | Solute Flux, Solute Name }

Elseif: Operation Mode Option = { **Water-Salt** }
Surface Flux Type Option[a],
{ Aqueous Volumetric Flux | Aqueous Mass Flux | Salt Mass Flux |
Solute Flux, Solute Name }

Elseif: Operation Mode Option = { **Water-Air-Salt** }
Surface Flux Type Option[a],
{ Aqueous Volumetric Flux | Aqueous Mass Flux | Gas Volumetric Flux |
Gas Mass Flux | Salt Mass Flux | Solute Flux, Solute Name }

Elseif: Operation Mode Option = { **Water-Air-Salt-Energy** }
Surface Flux Type Option[a],
{ Aqueous Volumetric Flux | Aqueous Mass Flux | Gas Volumetric Flux |
Gas Mass Flux | Salt Mass Flux | Heat Flux | Solute Flux, Solute Name }

Elseif: Operation Mode Option = { **Water-CO2-NaCl** }
Surface Flux Type Option[a],
{ Aqueous Volumetric Flux | Aqueous Mass Flux | Gas Volumetric Flux |
Gas Mass Flux | Salt Mass Flux | CO_2-Mass Flux |
Aqueous-Phase CO_2 Mass Flux | Gas-Phase CO_2 Mass Flux |
CO_2-Mass Flux | Solute Flux, Solute Name }

Elseif: Operation Mode Option = { **Water-CO2-NaCl-Energy** }
Surface Flux Type Option[a],
{ Aqueous Volumetric Flux | Aqueous Mass Flux | Gas Volumetric Flux |
Gas Mass Flux | Salt Mass Flux | CO_2-Mass Flux |
Aqueous-Phase CO_2 Mass Flux | Gas-Phase CO_2 Mass Flux |
CO_2-Mass Flux | Heat Flux | Solute Flux, Solute Name }

Endif:

If: Surface Flux Type Option = { Heat Flux }
Units[b] (W), Units[c] (J),
ElseIf: Surface Flux Type Option = { Volumetric Flux }
Units[b] (m^3/s), Units[c] (m^3),
Elseif: Surface Flux Type Option = { Mass Flux } { Dissolved Oil }
Units[b] (kg/s), Units[c] (kg),
Elseif: Surface Flux Type Option = { Solute Flux }

Surface Flux Card (cont'd)

Units[b] (sol/s), Units[c] (sol),
Endif:

Surface Flux Orientation Option[d]
{ West } { East }
{ South } { North }
{ Top } { Bottom }
I-Start Index[e], I-End Index[f],
J-Start Index[g], J-End Index[h],

K-Start Index[i], K-End Index[j],
Format: Char[a], Char[b], Char[c], Char[d], Integer[e], Integer[f], Integer[g], Integer[h], Integer[i], Integer[j],

Endfor: Number of Surface Flux Inputs

Endcard: Surface Flux Card

B.26.1 Surface Flux Card Examples

Extracted from STOMP1 (Water) input file:

```
~Surface Flux Card
5,
Aqueous Volumetric,gal/min,gal,East,2,2,1,1,14,33,
Aqueous Volumetric,gal/min,gal,East,2,2,1,1,71,100,
Aqueous Volumetric,gal/min,gal,East,52,52,1,1,1,113,
Solute Flux,TCE,1/min,,East,2,2,1,1,14,33,
Solute Flux,TCE,1/min,,East,2,2,1,1,71,100,
```

Extracted from STOMP1 (Water) input file:

```
~Surface Flux Card
4,
Solute Flux,Tc-99,1/yr,,Bottom,1,10,1,1,1,1,
Solute Flux,U-238,1/yr,,Bottom,1,10,1,1,1,1,
Solute Flux,Np-237,1/yr,,Bottom,1,10,1,1,1,1,
Solute Flux,Pu-239,1/yr,,Bottom,1,10,1,1,1,1,
```

Extracted from STOMP1 (Water) input file:

```
~Surface Flux Card
13,
8,bottom_flux_z16m.srf,
Aqueous Volumetric Flux,L/day,L,bottom,1,180,1,1,17,17,
Solute Flux,U:0.01,1/day,, bottom,1,180,1,1,17,17,
Solute Flux,U:0.03,1/day,, bottom,1,180,1,1,17,17,
Solute Flux,U:0.10, 1/day,, bottom,1,180,1,1,17,17,
Solute Flux,U:0.30,1/day,, bottom,1,180,1,1,17,17,
Solute Flux,U:0.60,1/day,, bottom,1,180,1,1,17,17,
Solute Flux,U:1.00, 1/day,, bottom,1,180,1,1,17,17,
Solute Flux,Tc,1/day,, bottom,1,180,1,1,17,17,
5,east_flux_x61m.srf,
Aqueous Volumetric Flux,L/day,L,East, 61,61,1,1,1,18,
Solute Flux,U:0.01,1/day,, East, 61,61,1,1,1,18,
Solute Flux,U:0.03,1/day,, East, 61,61,1,1,1,18,
Solute Flux,U:0.10,1/day,, East, 61,61,1,1,1,18,
Solute Flux,U:0.30,1/day,, East, 61,61,1,1,1,18,
```

Extracted from STOMP4 (Wate-Oil) input file:

```
~Surface Flux Card
1,
NAPL Volumetric Flux,cm^3/min,cm^3,Top,1,1,1,1,10,10,
```

Extracted from STOMP5 (Wate-Air-Oil) input file:

```
#   ------------------------------------------------------------------
~Surface Flux Card
#   ------------------------------------------------------------------
25,
# bottom of trench
NAPL Volumetric Flux,m^3/yr,m^3,top,13,15,13,17,79,79,
# top of pplz
```

Surface Flux Card Examples (cont'd)

NAPL Volumetric Flux,m^3/yr,m^3,top,1,27,1,34,60,60,
bottom of pplc
NAPL Volumetric Flux,m^3/yr,m^3,top,1,27,1,34,38,38,
plane through minimum elevation of water table
NAPL Volumetric Flux,m^3/yr,m^3,top,1,27,1,34,27,27,
South boundary of model domain
NAPL Volumetric Flux,m^3/yr,m^3,south,1,27,1,1,1,85,
North boundary of model domain
NAPL Volumetric Flux,m^3/yr,m^3,north,1,27,34,34,1,85,
West boundary of model domain
NAPL Volumetric Flux,m^3/yr,m^3,west,1,1,1,34,1,85,
East boundary of model domain
NAPL Volumetric Flux,m^3/yr,m^3,east,27,27,1,34,1,85,
Gas-total oil mass flux,kg/yr,kg,top,1,27,1,34,85,85,
Gas-total oil mass flux,kg/yr,kg,south,1,27,1,1,61,85,
Gas-total oil mass flux,kg/yr,kg,south,1,27,1,1,39,60,
Gas-total oil mass flux,kg/yr,kg,south,1,27,1,1,1,38,
Gas-total oil mass flux,kg/yr,kg,north,1,27,34,34,61,85,
Gas-total oil mass flux,kg/yr,kg,north,1,27,34,34,39,60,
Gas-total oil mass flux,kg/yr,kg,north,1,27,34,34,1,38,
Gas-total oil mass flux,kg/yr,kg,west,1,1,1,34,61,85,
Gas-total oil mass flux,kg/yr,kg,west,1,1,1,34,39,60,
Gas-total oil mass flux,kg/yr,kg,west,1,1,1,34,1,38,
Gas-total oil mass flux,kg/yr,kg,east,27,27,1,34,61,85,
Gas-total oil mass flux,kg/yr,kg,east,27,27,1,34,39,60,
Gas-total oil mass flux,kg/yr,kg,east,27,27,1,34,1,38,
Dissolved oil mass flux,kg/yr,kg,top,1,27,1,34,85,85,
Dissolved oil mass flux,kg/yr,kg,south,1,27,1,1,1,85,
Dissolved oil mass flux,kg/yr,kg,north,1,27,34,34,1,85,
Dissolved oil mass flux,kg/yr,kg,west,1,1,1,34,1,85,
Dissolved oil mass flux,kg/yr,kg,east,27,27,1,34,1,85,

B.27 Thermal Properties Card

Card Title[a] { ~Thermal [Properties Card] }
Format: *Char*[a]

For: Number of Rock/Soil Types
Rock/Soil Name[a],
Thermal Conductivity Function Option[b],
{ Constant } { Parallel } { Linear } { Somerton } { Campbell }

If: Thermal Conductivity Function Option = { Constant }
X-Dir. Thermal Conductivity[c], Units[d] (W/m K),
Y-Dir. Thermal Conductivity[e], Units[f] (W/m K),
Z-Dir. Thermal Conductivity[g], Units[h] (W/m K), Specific Heat[i], Units[j] (J/kg K),
Format: *Char*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Char*[j],
Elseif: Thermal Conductivity Function Option = { Parallel }
X-Dir. Rock/Soil Grain Thermal Conductivity[c], Units[d] (W/m K),
Y-Dir. Rock/Soil Grain Thermal Conductivity[e], Units[f] (W/m K),
Z-Dir. Rock/Soil Grain Thermal Conductivity[g], Units[h] (W/m K),
Specific Heat[h], Units[i] (J/kg K),
Format: *Char*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h], *Real*[i], *Char*[j],
Elseif: Thermal Conductivity Function Option = { Linear }
X-Dir. Rock/Soil Unsaturated Thermal Conductivity[c], Units[d] (W/m K),
Y-Dir. Rock/Soil Unsaturated Thermal Conductivity[e], Units[f] (W/m K),
Z-Dir. Rock/Soil Unsaturated Thermal Conductivity[g], Units[h] (W/m K),
X-Dir. Rock/Soil Water Saturated Thermal Conductivity[i], Units[j] (W/m K),
Y-Dir. Rock/Soil Water Saturated Thermal Conductivity[k], Units[l] (W/m K),
Z-Dir. Rock/Soil Water Saturated Thermal Conductivity[m], Units[n] (W/m K),
Specific Heat[o], Units[p] (J/kg K),
Format: *Char*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h],
Real[i], *Char*[j], *Real*[k], *Char*[l], *Real*[m], *Char*[n], *Real*[o], *Char*[p],
Elseif: Thermal Conductivity Function Option = { Somerton }
X-Dir. Rock/Soil Unsaturated Thermal Conductivity[c], Units[d] (W/m K),
Y-Dir. Rock/Soil Unsaturated Thermal Conductivity[e], Units[f] (W/m K),
Z-Dir. Rock/Soil Unsaturated Thermal Conductivity[g], Units[h] (W/m K),
X-Dir. Rock/Soil Water Saturated Thermal Conductivity[i], Units[j] (W/m K),
Y-Dir. Rock/Soil Water Saturated Thermal Conductivity[k], Units[l] (W/m K),
Z-Dir. Rock/Soil Water Saturated Thermal Conductivity[m], Units[n] (W/m K),
Specific Heat[o], Units[p] (J/kg K),
Format: *Char*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g], *Char*[h],
Real[i], *Char*[j], *Real*[k], *Char*[l], *Real*[m], *Char*[n], *Real*[o], *Char*[p],
Elseif: Thermal Conductivity Function Option = { Campbell }
Parameter a[c] (0.734), Units[d] (W/m K), Parameter b[e] (1.45), Units[f] (W/m K),
Parameter c[g] (2.01), Parameter d[h] (0.204), Units[i] (W/m K),
Parameter e[j] (4.0), Specific Heat[k], Units[l] (J/kg K),
Format: *Char*[a], *Char*[b], *Real*[c], *Char*[d], *Real*[e], *Char*[f], *Real*[g],

Thermal Properties Card (cont'd)

Real[h], *Char*[i], *Real*[j], *Real*[k], *Char*[l],
Endif:

Endfor: Number of Rock/Soil Types

Endcard: Thermal Properties Card

B.27.1 Thermal Properties Card Examples

Extracted from a STOMP3 (Water-Air-Energy) input file:
~Thermal Properties Card
Soil,Constant,0.5,W/m K,0.5,W/m K,0.5,W/m K,750,J/kg K

Extracted from a STOMP3 (Water-Air-Energy) input file:
~Thermal Properties Card
Silt Loam-Gravel Admix,Somerton,0.25,W/m K,0.25,W/m K,0.25,W/m K,2.0,W/m K,2.0,W/m K,2.0,W/m K,750,J/kg K,
Gravel Drainage,Somerton,0.25,W/m K,0.25,W/m K,0.25,W/m K,2.0,W/m K,2.0,W/m K,2.0,W/m K,750,J/kg K,
Gravel Filter,Somerton,0.25,W/m K,0.25,W/m K,0.25,W/m K,2.0,W/m K,2.0,W/m K,2.0,W/m K,750,J/kg K,
Riprap,Somerton,0.25,W/m K,0.25,W/m K,0.25,W/m K,2.0,W/m K,2.0,W/m K,2.0,W/m K,750,J/kg K,
Compacted Silt Loam,Somerton,0.25,W/m K,0.25,W/m K,0.25,W/m K,2.0,W/m K,2.0,W/m K,2.0,W/m K,750,J/kg K,

B.28 Ucode Control Card

Card Title[a] { ~Ucode [Control Card] }
Format: *Char*[a]

If: Operational Mode Option Card = { **Water** } { **Fluid** }

If: Execution Mode Option = { Normal w/ Inverse } { Restart w/ Inverse }

Ucode Phase[a], { 1 | 2 | 3 | 11 | 22 | 33 | 44 | 45 }
Note: 1 = Forward Modeling
2 = Sensitivities at Starting Parameters
3 = Perfrom Regression
11 = Calculates Sum of Squares
22 = Sensitivities at Starting Parameters using Central Differences
33 = Calculate Model Linearity
44 = Calculate Prediction Intervals
45 = Calculate Differnences and Prediction Intervals

Ucode Differencing Index[b], { 1 | 2 }
Note: 1 = Forward Differencing (Recommended)
2 = Central Differencing

Ucode Tolerance[c], Ucode Sum-of-Squared Residual Factor[d],
Ucode Quasi-Newton Updating Index[e], { 0 | 1 }
0 = No Quasi-Newton Updating
1 = Quasi-Newton Updating

Maximum Change Factor[f],
Format: *Integer*[a], *Integer*[b], *Real*[c], *Real*[d], *Integer*[e], *Real*[f],

Ucode Path and Name of Inverse Code[a],
Format: *Char*[a],
Ucode Number of Application Models[a],
Format: *Integer*[a],
For: Number of Application Models
Ucode Application Model Execution Commands[a],
Format: *Char*[a],
Endfor:

Ucode Scale Sensitivities Index[a], { 0 | 1 | 2 | 3 }
Note: 0 = No Scaling is Applied and Unscaled Sensitivities are Printed
1 = Dimensionless Scaled Sensitivities are Printed
2 = One-Percent Scaled Sensitivities are Printed
3 = Both Dimensionless and One-Percent Scaled Sensitivities are Printed

Ucode Control Card (cont'd)

Ucode Print Intermediate Index[b], { 0 | 1 }
Note: 0 = No Printing for Intermediate Iterations
1 = Printing for Intermediate Iterations

Ucode Print Graph Index[c], { 0 | 1 }
Note: 0 = Do not Print Post-Processing Files
1 = Print Post-Processing Files

Number of Residual Sets[d],
Format: Integer[a], Integer[b], Integer[c], Integer[d],
Endif:

Endif:

Endcard: Ucode Control Card

B.28.1 UCode Control Card Examples

Extracted from a STOMP1 (Water) input file:

```
~UCode Control Card
1,1,0.01,0.01,0,20,1.0,
..\bin\MRDRIVE,
1,
batch,
3,0,1,1,
```

Contents Appendix B