SANDIA REPORT

SAND20001-1555
Unlimited Release
Printed June 2001

PorSalsa User's Manual

Mario J. Martinez, Polly L. Hopkins, and Paul C. Reeves

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

SAND2001-1555
Unlimited Release
Printed June 2001

PorSalsa User's Manual

Mario J. Martinez and Polly L. Hopkins
Multiphase Processes Department
mjmarti@sandia.gov

Paul C. Reeves
Data Exploitation Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0834

Abstract

This report describes the use of PorSalsa, a parallel-processing, finite-element-based, unstructured-grid code for the simulation of subsurface nonisothermal two-phase, two component flow through heterogeneous porous materials. PorSalsa can also model the advective-dispersive transport of any number of species. General source term and transport coefficient implementation greatly expands possible applications. Spatially heterogeneous flow and transport data are accomodated via a flexible interface.

Discretization methods include both Galerkin and control volume finite element methods, with various options for weighting of nonlinear coefficients. Time integration includes both first and second-order predictor/corrector methods with automatic time step selection. Parallel processing is accomplished by domain decomposition and message passing, using MPI, enabling seamless execution on single computers, networked clusters, and massively parallel computers.

Forward

This document represents the results of an LDRD project funded out of the Energy & Critical Infrastructures SBU LDRD area. The work included theoretical model development, numerical model implementation into an existing parallel-processing research platform, and evaluation of model capabilities via a variety of benchmark problems (included in this document).

The intent of this work was to develop an integrated capability to model a broad class of problems involving nonisothermal multiphase flow and transport in heterogeneous porous media. Expected applications include such diverse areas as microscale heatpipe design, air permeameter evaluation and design, analysis of the subsurface transport of jet fuels in fuel spill fires, and the analysis of flow and transport in intensely fractured granite for waste repository evaluation.

Acknowledgement

We would like to thank Teklu Hadgu (6821) for collaborating on testing and evaluation of heat-driven flow problems, and Jim Ramsey (6852) for collaborating on well models. We are grateful to Jim Ramsey and Alex Treadway (6821) for installing PorSalsa on the Geoscience and Environment Center Linux cluster. We must also thank the MPSalsa team (Shadid *et al*., 1999) for their help during this project. Finally, we appreciate the support of this work provided by our management, J. E. Johannes, A. C. Ratzel, B. P. Davies, and E. K. Webb.

The code and model development work was funded by the LDRD program at Sandia National Laboratories. Example problems related to repository design were funded through Work-For-Others programs led by Department 6115. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-ACO4-94AL85000.

Table of Contents

Introduction

History of Development

PorSalsa is a parallel-processing, finite-element-based, unstructured-grid code for the simulation of subsurface nonisothermal flow and transport through heterogeneous porous materials. PorSalsa is an outgrowth of the program MPSalsa (Shadid et al., 1996, 1999), which was developed to solve fluid mechanics problems involving chemical reactions. PorSalsa extends these capabilities to include the ability to also model two phase (liquid-gas), two component (air-water) subsurface thermal transport in highly heterogeneous porous media. PorSalsa can also model the advective/dispersive transport of any number of species (at trace concentrations) through porous materials. The transport model includes general source term descriptions in which the source terms can be dependent on any of the other species. This general implementation thereby allows the modeling of various complex processes such as radioactive decay chains, and transport in dual permeability representations of heterogeneous media.

The development philosophy adopted for PorSalsa has been to use MPSalsa as a platform, thereby leveraging the significant past and present development efforts that have gone into MPSalsa. MPSalsa has been, and continues to be, a prototype for the development of new and emerging methods for the solution of nonlinear PDEs on unstructured grids using massively parallel processing computers. The development philosophy for PorSalsa thus leverages new developments that are implemented into MPSalsa. PorSalsa and MPSalsa are maintained under the same software control system; developments in either code are readily transferable to the other. In the early stages of PorSalsa development, this strategy allowed the rapid implementation of a very general parallel processing capability for simulation of flows in porous media. This strategy continues to pay dividends today. As a result, all the modeling capability that exists in MPSalsa is also included in PorSalsa, but with the addition of the subsurface transport capabilities that are the specialty of PorSalsa.

Code Features

Physics

For completeness, the equations describing the transport of mass and energy in a porous medium, as implemented in PorSalsa, are presented below. The conservation of component α in the liquid and gas phases is described by,

$$\frac{\partial}{\partial t}\phi\left(Y_{\alpha l}\rho_l S_l + Y_{\alpha g}\rho_g S_g\right) + \nabla \bullet \left(Y_{\alpha l}\rho_l \mathbf{v}_l + Y_{\alpha g}\rho_g \mathbf{v}_g - \mathbf{J}_{\alpha g}\right) = Q_\alpha \qquad (1)$$

where subscripts *l* and *g* represent liquid and gas, respectively, ϕ is porosity, $Y_{\alpha\beta}$ is the mass fraction of component α in phase β (liquid or gas), *S* is saturation, ρ is density, and Q_α is a volumetric source. PorSalsa can model the transport of at most two components, water ($\alpha = w$) and air ($\alpha = a$), but only one need be invoked in a simulation. The Darcy fluxes for phase β are given by,

$$\boldsymbol{v}_\beta = -\frac{k_{r\beta}}{\mu_\beta}\boldsymbol{k} \bullet \left(\nabla P_\beta + \rho_\beta \boldsymbol{g}\right), \tag{2}$$

where $k_{r\beta}$ is relative permeability to phase β, $\boldsymbol{k}$ is the intrinsic permeability tensor of the porous medium, μ_β is phase viscosity, *P* is pressure and $\boldsymbol{g}$ the gravitational acceleration. Capillary pressure, $P_c = P_g - P_l$, is included in the model. The gas phase binary diffusion is modeled by,

$$\boldsymbol{J}_{\alpha g} = -\rho_g D_{\alpha g} \nabla Y_{\alpha g}, \tag{3}$$

in which the diffusion coefficient can depend on pressure and temperature.

The energy equation implemented in PorSalsa is

$$\frac{\partial}{\partial t}\left[\left((1-\phi)\rho_s e_s\right) + \phi\left(e_l \rho_l S_l + e_g \rho_g S_g\right)\right] + \nabla \bullet \boldsymbol{q} = Q_\alpha, \tag{4}$$

in which subscript *s* represents the solid porous matrix and *e* is internal energy. The total heat flux vector includes heat conduction and convection and heat transport due to binary diffusion,

$$\boldsymbol{q} = -\lambda_T \nabla T + \sum_\beta \rho_\beta \boldsymbol{v}_\beta h_\beta + \sum_\alpha h_{\alpha g} \boldsymbol{J}_{\alpha g} \tag{5}$$

where λ_T is a saturation-dependent effective thermal conductivity, *T* is temperature, h_β is phase enthalpy and $h_{\alpha g}$ is the enthalpy of component α in the gas phase. The user can activate any combination of the foregoing three transport equations.

PorSalsa can also model the transport of any number of species through porous media. For each species, *k*, PorSalsa models the transport with the following equation,

$$\frac{\partial}{\partial t}\left(\theta\rho Y_k\right) + \nabla \bullet \left(\rho \boldsymbol{v} Y_k\right) = -\nabla \bullet \left(-\rho\theta \boldsymbol{D} \nabla Y_k\right) + W_k \dot{\omega}_k, \tag{6}$$

where $\theta = \phi S_l$ is the liquid moisture content, Y_k is the mass fraction of species *k*, W_k is the molecular weight, and $\dot{\omega}_k$ is the molar production rate of species *k*. The dispersive/diffusive tensor is given by

$$\theta D = \alpha_T |\boldsymbol{v}| \boldsymbol{I} + (\alpha_L - \alpha_T)\frac{\boldsymbol{v}\boldsymbol{v}}{|\boldsymbol{v}|} + D_m \theta \tau \boldsymbol{I} \tag{7}$$

where α_T and α_L are transverse and longitudinal dispersivities, D_m is the molecular diffusion coefficient, and τ is tortuosity. Some of the current capabilities and highlights embodied in these model equations include:

Two-phase, two-component thermal model (water, air, energy)

- Water material is calculated from steam table data, assuming equilibrium thermodynamic phase partitioning.
- Air is treated as an ideal gas, applying Henry's Law partitioning.
- Other EOS available (JP8 jet fuel; incompressible NAPL as a noncondensible).
- Energy transport includes conduction, latent and sensible energy transfer, including binary diffusion of heat.
- General specification of transport property dependence on solution vector (e.g., densities, viscosities, diffusion coefficients depend on temperature, pressure and phase volume fraction).
- Heterogeneous data can be specified for formation properties and transport models, including a linkage to GSLIB (Deutsch & Journel, 1998).

Species transport model (arbitrary number of species)

- Advective/diffusive transport.
- Hydrodynamic dispersion (full tensor description).
- General specification of sources with general dependence on other species (e.g. radioactive decay chains, dual porosity).
- Velocity field input options include: a) user-specified, b) solution file from another PorSalsa simulation.

Numerical Methods

Spatial Discretization

The numerical method applied for solving the initial-boundary value problem formed by the coupled system of equations is a finite-element-based method, enabling a general representation of complex geologic stratigraphy. The spatial discretization is accomplished either by a Galerkin finite element method (GFEM) or a control volume finite element method (CVFEM). The GFEM and the CVFEM can both utilize fully integrated isoparametric multilinear elements via Gauss-Legendre quadrature. The CVFEM can also be invoked using vertex quadrature and can apply upwind, midpoint and harmonic weighting of transmissivity coefficients for the flow equations. The CVFEM upwinding scheme was implemented primarily for problems with discontinuous solutions (e.g. phase boundaries), for which the GFEM method is unsuitable. Species transport equations are only implemented with a GFEM method. (An untested Galerkin Least Squares scheme is also available.)

Temporal Discretization

The spatial discretization results in a system of nonlinear ordinary differential equations with time as the continuous variable. This system of ordinary differential equations is integrated forward in time by a variable-step, backward-difference, predictor-corrector scheme first described by Gresho *et. al* (1980). Two time integration methods are implemented. A first order scheme employs a forward Euler (FE) predictor with a backward Euler (BE) corrector. A second order scheme employs an Adams-Bashforth predictor with a trapezoid rule corrector. The predictor/corrector scheme provides a method for estimating the local time truncation error, thereby providing a rational scheme for automatic time step control based on a user-specified truncation error tolerance. See Martinez *et al*. (1997) for details on these time integration methods.

Solution Procedures

Time and space discretization results in a system of nonlinear equations. The nonlinear system is solved for the solution variables by an "inexact" Newton iteration scheme; the term inexact refers to a numerical approximation of the Jacobian. The Jacobian can be computed efficiently via forward difference approximations, by exploiting the fact that most terms are sums of products of basis functions and grid variables. The inexact Newton scheme is a convenient method of determining the Jacobian because any new transport function or equation of state can be implemented without the need for the user to also program the gradient of the functions with respect to the solution vector.

The Newton iteration scheme generates a linear system of equations to be solved for each update vector. The systems are solved using a parallel-processing, preconditioned Krylov solver library called Aztec (Tuminaro *et al*., 1999). The library includes several parallel iterative solution methods, including the conjugate gradient method for symmetric positive definite systems and a number of related methods for nonsymmetric systems, e.g. the generalized minimum residual method (GMRES). The library includes several preconditioners (e.g., Jacobi, least-squares polynomial, incomplete LU decomposition), which can be "mixed and matched" with the Krylov methods. These options are given in the *Linear Solver Subsection* starting on page 23, however the user is referred to Tuminaro *et al*. for a more complete description.

Parallel Implementation

The entire numerical algorithm is implemented for distributed memory parallel computers, or networked systems, via domain decomposition and message-passing techniques. Message passing is accomplished using the MPI (Gropp *et al*., 1995) library, thus making the algorithm highly portable. In addition to the message passing techniques implemented in MPSalsa, some special data structures were necessary for the porous medium calculations. This is because our formulation requires a unique material type specified on a node-point basis. This requirement can be attributed to the capillary pressure vs. saturation constitutive model, which is non-unique at a material boundary, and our choice of primary variables. The domain decomposition itself is performed with a separate tool called **nem_slice** (Appendix E), which is itself built around the Chaco (Hendrickson, *et al*., 1995) graph partitioning code.

Getting Started

Files

The files necessary for performing a serial (single processor) simulation with PorSalsa include a problem definition input file, a finite element mesh file in the EXODUS II (Schoof & Yarberry, 1992) format, and an output file (which also serves as a restart file) for the primary variables, also in EXODUS II format. The primary variables used by PorSalsa for two-phase flow problems include $\left(\rho_w, P_a, T\right)^T$, where

$$\rho_w = \rho_l Y_{wl} S_l + \rho_g Y_{wg} (1 - S_l) \, . \qquad (8)$$

For advection-diffusion problems the primary variables are the mass fractions. If other secondary variables (described in ***Auxiliary Output Specifications*** on page 58) are desired from the simulation, an auxiliary output file is also needed.

A majority of this document is devoted to describing the various parts of a problem definition input file; this information can be found in ***Problem Definition Input File***, starting on page 14. Generation of mesh files and input/output files is described in ***Other Miscellaneous Files*** on page 70. If a simulation involves parallel processing then a load balance file is needed. The load balance file describes the mesh partitioning among the various processors. The procedure for generating this file is also described in the ***Other Miscellaneous Files*** chapter on page 70.

Sample Problems

The distribution of PorSalsa includes a subdirectory `Salsa/Sample_Problems` which contains subdirectories populated with complete problem set ups and output results. The contents will include a problem definition input file, mesh file, and files containing solution output which can be used to verify proper installation of PorSalsa.

Problem Definition Input File

In general, PorSalsa provides enormous leeway in the way in which the primary input file is configured. The input file is divided into distinct sections, each with a specific header followed by a listing of parameters and the values being assigned to them. Some of the sections are mandatory and must be present, while others are optional. They may, however, be listed in any order. The header must be spelled exactly as listed in the following sections and is used by the subroutines handling data input to find the beginnings of sections. If the exact header is not found then the section is either assumed to be missing (resulting in an error message for mandatory sections), or its parameters are assumed to take on default values (for optional sections). If the header is present but misspelled, then its parameter settings are ignored. The exact headers corresponding to the possible sections are as follows:

• "General Problem Specifications"	Mandatory
• "Solution Specifications"	Mandatory
• "Solver Specifications"	Mandatory
• "Chemistry Specifications"	Mandatory
• "Porous Flow Specifications"	Mandatory
• "Auxiliary Input Specifications"	Optional
• "Material ID Specifications"	Mandatory
• "Boundary Condition Specifications"	Mandatory
• "Initial Guess/Condition Specifications" -or- "Initial Condition/Guess Specifications"	Mandatory
• "Output Specifications"	Optional
• "Auxiliary Output Specifications"	Optional
• "Data Specification for User's Functions"	Optional

Comments may be inserted anywhere in the input file as long as the exact strings defining the section headings and parameters are not used. A "#" in the first column of a line denotes a comment line.

Formatting Key

Table 1 provides a listing of the formatting and notation used in the following sections to describe the contents of the problem definition input file. Font changes have been used to further emphasize meaning throughout the report. Courier font has been used to denote entries in the input file (*e.g.*, `Debug = 0`); input file section headings are in Courier font and enclosed in quotes. Arial font, in bold highlighting, has been used to denote utility programs and file types (*e.g*., **add_var, mesh_file.exoII**). Arial font, in normal highlighting, has been used to denote Aztec parameter settings (*e.g*., AZ_none).

The following notes refer to actual formatting within the problem definition input file:

- The section headings, parameter descriptions, and parameter options should be treated as case-sensitive.
- White space in section headings and parameter descriptions is necessary with the exception of that surrounding "=".
- There are no requirements for the ordering of sections and their parameter lists, except where parameters are clearly linked. An example of the latter is:

```
Number of auxiliary nodal variables = 2
Auxiliary nodal variable name(s):
  PERM
  PORO
```

In this case the first and second lines must be listed in this order, while the third and fourth lines must follow the second.

Many of the descriptions given in the following sections are taken from the original manual for MPSalsa (Salinger *et al*., 1996) from which PorSalsa was developed. The user may find more detailed information given there in some cases. As well, many of the parameters associated with the solver specifications pertain to the Aztec library of solver software. The reader is referred to the Aztec user's guide for further details regarding Aztec specifications and parameters (Tuminaro *et al*., 1999).

Physical Units

PorSalsa uses the standard SI system of units (meter – kilogram – second). Pressure values are to be input in Pascals; temperature in degrees Celsius. The physical units associated with parameters in the main input file are noted where appropriate.

Feature	*Indicator*	*Example*
Optional Section or Parameter	[]	`[Number of processors =` *`integer`*`]`
Exact Section Heading or Parameter Name	Normal	`Number of processors =` *`integer`*
User Specified Option or Value	Italic	`Number of processors =` *`integer`*
Choice of Predetermined Options	{|}	`Step Control = {`***`on`***`|`*`off`*`}`
Default Setting	Bold	`Step Control = {`***`on`***`|`*`off`*`}`
User Supplied Integer Value	*integer*	`Number of processors =` *`integer`*
User Supplied Real Value	*float*	`Initial Step Size =` *`float`*
User Supplied Character String	*string*	`Input FEM file =` *`string`*

Table 1: Syntax used in the subsequent sections to describe the contents of the problem definition input file.

General Specifications

The "`General Problem Specifications`" portion of the input file determines the type of problem being solved (*i.e.*, flow through porous media, *etc*.) and indicates other basic information such as the name of the mesh, output, and load balance files. Flags for some numerical options and debugging levels are also set here. This section is mandatory.

Example:

```
------------------------------------------------------------
      General Problem Specifications
------------------------------------------------------------
Problem type                     = porous_flow
Input FEM file                   = /tmp/m3d_0.genII
LB file                          = /tmp/m3d_0-m4-bKL.nemI
Output FEM file                  = /tmp/m3d_out.exoII
Number of processors             = 4
Cartesian or Cylindrical when 2D = Cartesian
Interpolation order              = linear
Stabilization                    = default
Debug                            = 0
```

Description and Usage

`Problem type = {`*`porous_flow`*`|`*`mass_conv_diff`*`}`

Only these two options are appropriate for flow through porous media. Simulation of the flow of air, water, and/or energy in the absence of dissolved species is specified via `porous_flow`. Simulation of advective – dispersive transport of dissolved species is specified with `mass_conv_diff`. The reader is referred to the MPSalsa manual for instructions pertaining to other types of simulations (Salinger *et al*., 1996, p. 15).

`Input FEM file =` *`string`*

This is the name of the EXODUS II file containing the finite element mesh. See the ***Other Miscellaneous Files*** section of this report for more information on this file. The path is optional if the file is located in the same directory as the PorSalsa executable.

`[LB file =` *`string`*`]`

This is the name of the load balance file generated by the utility program **nem_slice**, which is described in the ***Other Miscellaneous Files*** section of this report. The path is optional if the file is located in the same directory as the PorSalsa executable. This line of the input file is required only if more than one processor is specified.

`[Output FEM file =` *`string`*`]`

This is the name of the output EXODUS II file that will contain the computed results for the primary solution variables. The path is optional if the file is located in the same directory as the PorSalsa executable.

`[Number of processors =` *`integer`*`]`

The number of processors defined here must be consistent with the load balance file, and with the number of processors requested at the time of execution (Appendix A). The default value is ***`1`***.

`[Cartesian or Cylindrical when 2D =` ***`Cartesian`***`]`

Only Cartesian coordinates have been implemented for porous medium problems. Since this is the default value, this line is optional.

`[Interpolation order = {`***`linear`***`|`*`quadratic`*`}]`

This line specifies the interpolation order for all quantities in the finite-element model. Only ***`linear`*** interpolation is appropriate for `porous_flow` problem types.

`[Stabilization =` ***`default`***`]`

Only ***`default`*** stabilization is appropriate for `porous_flow` problem types. Since this is the default value, this line is optional. For other types of problems the reader is referred to Salinger *et al*. (1996, p. 16)

`[Debug =` *`integer`*`]`

This line specifies how much information will be output to *stdout* during execution. It also determines how much information is output from the linear solver library (Aztec). The higher the number, the greater the amount of information written. The value of `integer` must lie in the range [`0-10`], with **`2`** being the default.

`Debug = 0`	Minimal information is printed to *stdout*; only a summary of important flags and entries into important code segments are printed.
`Debug > 0`	Along with the above information, timing information and summary information on the global FE mode (not the local processor FE model),

node sets, side sets, and boundary conditions are printed. The solver library prints out residual summaries as well.

`Debug > 6`	Along with the above information, summary information on the local processor FE model is printed. Processor-based vector quantities such as residual, initial guesses, and solutions are included. Processor-based communication summaries and local-to-global mapping information are also printed.
`Debug > 9`	Along with the above information, information on the local matrix is printed. This can be a significant amount of information and is really meant to debug smaller problems in detail.

Solution Specifications

The parameters specified in the "`Solution Specifications`" section of the input file direct the temporal component of the solution process. This section is mandatory whether or not the problem is steady-state.

Example:

```
------------------------------------------------------------
     Solution Specifications
------------------------------------------------------------
Solution Type                      = transient
Order of integration/continuation = 1
Step Control                       = on
Relative Time Integration Error    = 1.0e-3
Step Control Aggressiveness        = 2.0
Initial Step Size                  = 4.0e2
Maximum Step Size                  = 4.0e4
Maximum Number of Steps            = 1000
Maximum Time or Parameter Value    = 2.592e6
Mass Matrix Formulation Weight     = lumped
Mass Fraction Clipping             = off
```

Description and Usage

`Solution Type =` *`string`*

This parameter, which is mandatory, specifies the type of solution desired. Currently, *`string`* may take on the following values:

`steady`	Steady-state solution, *i.e.*, the transient terms of the governing equations are not computed.
`transient`	Transient solution, *i.e.*, the complete time-dependent governing equations are solved in a time-accurate manner by keeping the integration error under a specified tolerance.

`pseudo`	Transient pseudo-steady solution, *i.e.*, the complete time-dependent governing equations are solved to a quasi-steady state by aggressively increasing the time step, regardless of the time integration error.
`continuation`	Used to solve a series of steady-state solutions as a function of a parameter using the function contained in the file "rf_user_continuation.c". This can be used to investigate how altered material properties or boundary conditions influence the solution behavior. See Salinger et al. (1996, p. 18) for more details.
`tp_continuation`	Continuation of a steady-state solution.
`optimization`	Optimization of a steady-state solution.
`steady_deltat`	Steady-state problem with transient Jacobian preconditioning.

`[Order of integration/continuation =` {***1***|*2*}|{*0*|***1***}`]`

This parameter has different options or meanings depending on whether the solution type is time-dependent (*`transient`* or *`pseudo`*) or *`continuation`*. The default value is ***1*** in either case. This specification is ignored if `Solution Type =` *`steady`*.

For `transient` or `pseudo` solutions

1	First-Order Forward-Euler/Backward-Euler Predictor Corrector Integration.
2	Second-Order Adams-Bashforth/Trapezoidal-Rule scheme. This scheme starts with a pair of first-order steps to begin.

For `continuation` solutions

0	The solution at the previous step is used as an initial guess for the current step
1	First-Order (Euler-Newton) continuation. In this case, the tangent to the previous solution with respect to the continuation parameter is calculated numerically, and is used to calculate an initial guess for the current solution.

`[Step Control =` {***on***|*off*}`]`

This parameter is read for non-*`steady`* solutions. When `Step Control` is *on*, the step size will be adjusted after successful steps. For *`transient`* solutions, the step size is chosen as a function of the value of the `Relative Time Integration Error` (see below). For *`pseudo`* and *`continuation`* solutions, the step size will always be increased following a successful step, with the increase depending on the rate of the number of Newton iterations needed for convergence divided by the maximum number of Newton iterations allowed. If the value of `Step Control` is *`off`*, the step size remains unchanged. For any of the solution types and either value of `Step Control`, the step size is cut in half after a failed step (*i.e.*, when a converged solution is not found in the maximum number of Newton iterations).

`[Relative Time Integration Error =` *`float`*`]`

This parameter is only required for `transient` solutions. A value of the time integration error is calculated using the difference between the predicted and corrected value of the solution by the method of Gresho *et al*., 1980, see Martinez *et al*. (1997) for details. If this estimated error is less than twice the value set in the input file, the ratio of the input error value and the estimated error are used to pick the next time step size. The value of the `Relative Time Integration Error` must be greater than the `Solution Relative Error Tolerance`, which is input in the Solver Specifications section to set the convergence criterion for the linear solver. The default value is ***10^{-3}***.

`[Step Control Aggressiveness =` *`float`*`]`

This controls the aggressiveness of the time step adjustments for *`pseudo`*, *`continuation`*, *`steady_deltat`*, and *`tp_continuation`* solutions. The default value is ***`2.0`***.

`[Initial Parameter Value =` *`float`*`]`

This value is only used for *`continuation`* solutions. It specifies the initial value of the continuation parameter. The default value is ***`none`***, which results in an error for *`continuation`* solutions.

`[Initial Step Size =` *`float`*`]` [sec]

This value is only used for non-*`steady`* solutions. It sets the size of the initial time step for *`transient`* or *`pseudo`* solutions, and sets the initial parameter step size for *`continuation`* solutions. If `Step Control` is *`off`*, then this step size stays constant throughout the solution as long as each step converges. The default value is ***`none`***, which will result in an error for *`transient`*, *`pseudo`*, and *`continuation`* runs.

`[Maximum Step Size =` *`float`*`]` [sec]

This value is only used for non-*`steady`* solutions. It sets the maximum size of a time step for non-*`steady`* solutions. The default value is ***`1e10`***.

`[Maximum Number of Steps =` *`integer`*`]`

This value is only used for non-*`steady`* solutions. The program terminates when the number of time steps exceeds this value. The default value is ***`1000`*** steps.

`[Maximum Time or Parameter Value =` *`float`*`]` [sec]

This value is only used for non-*`steady`* solutions. The program terminates when the simulation time exceeds this value. The default is no maximum simulation time.

`[Mass Matrix Formulation Weight =` {*`consistent`*|*`lumped`*|*`float`*}`]`

This parameter defines the mass matrix formulation, and is valid only for transient and pseudo solution types.

`[Mass Fraction Clipping =` {*`on`*|***`off`***}`]`

This parameter applies to transient and pseudo solutions of mass transfer problem types (e.g. *mass_conv_diff*). When `Mass Fraction Clipping` is on, computed values of mass fraction greater than one are set to 1.0, and those less than zero are set to 0.0.

Solver Specifications

The "`Solver Specifications`" section is used to define the parameters needed for the linear and non-linear solvers; there is a subsection for each solver with headings of "`nonlinear solver subsection:`" and "`linear solver subsection:`" respectively. Note the colon terminating each of the subsection headings. This section is mandatory, but many of the parameters may be invoked as default values.

Example:

```
------------------------------------------------------------
     Solver Specifications
------------------------------------------------------------
Override Default Linearity Choice = nonlinear

---------------- nonlinear solver subsection: --------------
Number of Newton Iterations                       = 8
Use Modified Newton Iteration                     = no
Enable backtracking for residual reduction        = no
Choice for Inexact Newton Forcing Term            = 0
Calculate the Jacobian Numerically                = no
Solution Relative Error Tolerance                 = 1.0e-4
Solution Absolute Error Tolerance                 = 1.0e-5

---------------- linear solver subsection: -----------------
Solution Algorithm               = gmres
Convergence Norm                 = 3
Preconditioner                   = no_overlap_ilu
Scaling                          = none
Orthogonalization                = classical
Size of Krylov subspace          = 64
Maximum Linear Solve Iterations  = 1000
Linear Solver Normalized Residual Tolerance   = 1.0e-7
```

Description and Usage

`Override Default Linearity Choice =` {***`default`***|*`linear`*|*`nonlinear`*}

> Problems involving porous media are typically nonlinear due to the nature of the equations of state and other constitutive relationships. This parameter should be set to *`nonlinear`* when solving porous medium problems. In the case of *mass_conv_diff* problem types , if the problem being solved is in fact linear, *linear* may be specified to override the nonlinear default. Note that there is no real penalty by specifying `nonlinear` when solving a linear problem.

Nonlinear Solver Subsection

Parameters of the "`nonlinear solver subsection:`" control the Newton iteration scheme. All of the parameter definition lines are optional since default values will be invoked, and include the following:

`[Number of Newton Iterations = `*`integer`*`]`

This parameter sets the maximum number of Newton iterations allowed during a nonlinear solve. If the maximum is reached and the convergence criterion has not been met, the nonlinear solve fails. For steady-state problems, this results in a termination with a fatal error. For transient problems, a convergence error is reported for the current time step, the time step size is then halved and the solution again attempted. The default value is **`25`**.

`[Use Modified Newton Iteration = {`*`yes`*`|`***`no`***`} [`*`auto`*`|`*`integer`*`] ]`

Modified Newton iteration is not implemented for porous medium problems. The default is ***`no`***.

`[Enable backtracking for residual reduction = {`*`yes`*`|`*`no`*`|`***`default`***`}]`

When a Newton iteration causes the norm of the residual to increase rather than decrease, backtracking will not accept the update. Instead, the algorithm looks in the same direction as the solution update from the Newton iteration. Performing residual calculations along the solution path given by the direction, it finds the solution that minimizes the residual. The algorithm in PorSalsa (same as for MPSalsa) follows the work of Eisenstat & Walker (1996). Backtracking has been shown in some cases to help converge to a steady-state when Newton's method without backtracking failed. The ***`default`*** flag disables backtracking for `transient` runs, but enables backtracking for all other solution types (*`pseudo`*, *`steady`*, and *`continuation`*).

`[Choice for Inexact Newton Forcing Term = `*`integer`*`]`

If backtracking is enabled, then these forcing term choices correspond to those in Eisenstat & Walker (1996). The algorithm attempts to save computations by solving the linear system to a coarser tolerance than the requested tolerance (`Linear Solver Normalized Residual Tolerance`) during the iteration process. The various forcing term choices govern this intermediate tolerance level. A value of 4 sets the criteria to a constant equal the `Linear Solver Normalized Residual Tolerance`. This choice requires the linear solve to meet this tolerance during the nonlinear iterations. The default value is ***`0`***.

`[Calculate the Jacobian Numerically = {`*`yes`*`|`*`no`*`|`***`default`***`}]`

For the *porous_flow* problem type this should be set to *`no`*. For *`mass_conv_diff`* , ***`default`*** results in an analytical Jacobian.

`[Solution Relative Error Tolerance = `*`float`*`]`

See Below. Default = ***`10`***$^{-3}$.

`[Solution Absolute Error Tolerance = `*`float`*`]`

Default = ***`10`***$^{-8}$.

The relative and absolute error tolerances are used in calculating the convergence criterion for the update vector in the nonlinear solver. The criterion is

$$\frac{1}{N}\sum_{i=1}^{N}\frac{|\delta_i|}{\varepsilon_R|x_i|+\varepsilon_A}<1.0$$

Where ε_R and ε_A are the relative and absolute tolerances specified above, δ_i is the update for the unknown x_i, and N is the total number of unknowns. The quantity on the left side of this inequality is what is output from the solver as the "update norm".

Linear Solver Subsection

The "`linear solver subsection:`" defines linear solver features as implemented by Aztec, see Tuminaro et al. (1999) for more details. All of its parameter lines are optional since acceptable default values will be invoked. Descriptions of the possible parameter options are given below together with their corresponding Aztec settings (in Arial font, *e.g.*, AZ_gmres):

[Solution Algorithm = {***gmres***|*tfqmr*|*cg*|*cgs*|*cgstab*|*lu*}]

gmres	Restarted General Minimized Residual Method. AZ_gmres
tfqmr	Transpose-Free Quasi Minimum Residual Method. AZ_tfqmr
cg	Conjugate Gradient Method. AZ_cg
cgs	Conjugate Gradient Squared Method. AZ_cgs
cgstab	Stabilized Biconjugate Gradient Method. AZ_bicgstab
lu	Full sparse LU factorization (only available on 1 processor). AZ_lu

For *porous_flow* type problems the matrix is seldom symmetric except for some simple diffusion problems. For this reason, conjugate-gradient solvers are not appropriate for most simulations involving porous media.

[Convergence Norm = {***0***|*1*|*2*|*3*|*4*}]

This defines the computational form of the norm used to measure convergence of the linear solver. Note that the default value of **0** corresponds to the norm in the GMRES method.

0	$\left\|r^k\right\|_2 / \left\|r^0\right\|_2$	AZ_r0
1	$\left\|r^k\right\|_2 / \left\|b\right\|_2$	AZ_rhs
2	$\left\|r^k\right\|_2 / \left\|A\right\|_\infty$	AZ_Anorm
3	$\left\|r^k\right\|_2 / \left(\left\|A\right\|_\infty \left\|x^k\right\|_1 + \left\|b\right\|_\infty\right)$	AZ_sol
4	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{r_i^k}{\varepsilon_R\left\|x_i\right\| + \varepsilon_A}\right)^2}$	AZ_weighted

`[Preconditioner = {`*`see list of recognized options below`*`}]`

This parameter determines the preconditioner, if any, used in the linear solver. The corresponding Aztec options are listed in Table 2. Many of the following descriptions of the preconditioner options are terse since preconditioner theory is outside the scope of this document. The options are better explained in the Aztec 2.1 User's Guide (Tuminaro *et al*., 1999).

none	No preconditioner applied.
jacobi	Jacobi preconditioner.
block_jacobi	Block Jacobi preconditioner.
sgs	Symmetric Gauss-Seidel.
poly	Polynomial preconditioner, with the order and type of polynomial specified by the following two lines of parameter settings.
ilu	Sparse incomplete LU factorization, ilu(k) with a drop tolerance.
icc	Incomplete Cholesky, similar to ilu but using icc(k) insteady of ilu(k).
no_overlap_ilu	Same as ilu with no overlapping between processors.
diag_overlap_ilu	ilu(k) with overlapping of diagonal blocks between processors.
real_overlap_ilu	ilu(k) with one level of overlap between processors.
ilut	Similar to ilu(k) but using Saad's (Saad, 1996) ILUT with a threshold tolerance instead of LU factorization.
no_overlap_ilut	ilut with no overlapping between processors.
diag_overlap_ilut	ilut with overlapping of diagonal blocks between processors.
real_overlap_ilut	ilut with one level of overlap between processors.
rilu	Similar to ilu(k) with a relaxation parameter.
no_overlap_rilu	`rilu` with no overlapping between processors.

diag_overlap_rilu	`rilu` with overlapping of diagonal blocks between processors.
real_overlap_rilu	`rilu` with one level of overlap between processors.
lu	Sparse LU factorization with drop tolerance.
no_overlap_lu	Sparse LU factorization with no overlapping between processors.
diag_overlap_lu	Sparse LU factorization with overlapping of diagonal blocks between processors.
real_overlap_lu	Sparse LU factorization with one level of overlap between processors.
bilu	Block `ilu(k)`.
no_overlap_bilu	`bilu` with no overlapping between processors.
diag_overlap_bilu	`bilu` with overlapping of diagonal blocks between processors.
real_overlap_bilu	`bilu` with one level of overlap between processors.
user	User-specified preconditioner (`AZ_user_precond`), see Aztec User's Guide.

`[Polynomial Order =` *`integer`*`]`

When a polynomial preconditioner is specified, this parameter line defines the order of the polynomial. The default value is ***3***.

`[Polynomial Type = {`*`LS`*`,` *`NS`*`}]`

When a polynomial preconditioner is specified, this parameter line defines the type of polynomial preconditioning.

LS	Least squares polynomial, `integer` choices are `{0,1,2,3,4,5,6,7,8,9}`. AZ_ls
NS	Neumann series, `integer` choices are `{0,1,2,...infinity}` AZ_Neumann

`[Level of Subdomain Overlapping = {`***`AZ_none`***`|`*`AZ_diag`*`|`*`AZ_full`*`}]`

This parameter determines the submatrices factored with the domain decomposition algorithm. For many of the preconditioning options this value is already preset.

`[Drop Tolerance for LU & ILUT =` *`float`*`]`

This parameter specifies the drop tolerance used in conjunction with LU and ILUT preconditioning. The default value is ***`0.0`***. For many of the preconditioning options, this value is already preset. AZ_drop

Preconditioner Option	*Preconditioner Type*	*Overlapping Degree*	*Subdomain Solver Type*
none	AZ_none	AZ_none	n/a
jacobi	AZ_Jacobi	AZ_none	n/a
block_jacobi	AZ_Jacobi	AZ_none	n/a
sgs	AZ_sym_GS	AZ_none	n/a
Poly	AZ_ls | AZ_Neumann	AZ_none	n/a
icc	AZ_dom_decomp	AZ_none	AZ_icc
ilu	AZ_dom_decomp	AZ_none	AZ_ilu
no_overlap_ilu	AZ_dom_decomp	AZ_none	AZ_ilu
diag_overlap_ilu	AZ_dom_decomp	AZ_diag	AZ_ilu
real_overlap_ilu	AZ_dom_decomp	AZ_full	AZ_ilu
ilut	AZ_dom_decomp	AZ_none	AZ_ilut
no_overlap_ilut	AZ_dom_decomp	AZ_none	AZ_ilut
diag_overlap_ilut	AZ_dom_decomp	AZ_diag	AZ_ilut
real_overlap_ilut	AZ_dom_decomp	AZ_full	AZ_ilut
rilu	AZ_dom_decomp	AZ_none	AZ_rilu
no_overlap_rilu	AZ_dom_decomp	AZ_none	AZ_rilu
diag_overlap_rilu	AZ_dom_decomp	AZ_diag	AZ_rilu
real_overlap_rilu	AZ_dom_decomp	AZ_full	AZ_rilu
lu	AZ_dom_decomp	AZ_none	AZ_lu
no_overlap_lu	AZ_dom_decomp	AZ_none	AZ_lu
diag_overlap_lu	AZ_dom_decomp	AZ_diag	AZ_lu
real_overlap_lu	AZ_dom_decomp	AZ_full	AZ_lu
bilu	AZ_dom_decomp	AZ_none	AZ_bilu
no_overlap_bilu	AZ_dom_decomp	AZ_none	AZ_bilu
diag_overlap_bilu	AZ_dom_decomp	AZ_diag	AZ_bilu
real_overlap_bilu	AZ_dom_decomp	AZ_full	AZ_bilu

Table 2: Preconditioner options and their corresponding Aztec options.

[Fill Factor for ILUT = *float*]

This parameter sets the fill factor for ILUT preconditioning. The default value is ***1.0***.

[Enable RCM Reordering for ILUT = {***yes***|*no*|*default*}]

Enable reverse Cuthill-McKee ordering. Note that *default* is equivalent to *yes*.

`[RILU Relaxation Parameter =` *`float`*`]`

Default = ***`1.0`***.

`[Scaling =` {*`block_jacobi`*|*`sym_diag`*|***`row_sum`***|*`none`*}`]`

Determines how the linear system is scaled.

`[Orthogonalization =` {***`classical`***|*`modified`*}`]`

Sets the orthogonalization scheme for GMRES

`classical`	Classical Gramm-Schmidt orthogonalization . AZ_classic
`modified`	Modified Gramm-Schmidt orthogonalization. AZ_modified

`[Size of Krylov subspace =` *`integer`*`]`

Default = ***`64`***.

`[Maximum Linear Solve Iterations =` *`integer`*`]`

This sets the maximum number of iterations for any linear solve. When this maximum is reached before the residual has been reduced by the specified amount (as specified by the `Choice for Inexact Newton Forcing Term` and `Linear Solver Normalized Residual Tolerance` values), the linear solver terminates and an error condition is returned to the calling program. For nonlinear problems, the solution is accepted nonetheless and the next Newton step is started. For restarted *`gmres`*, this number is usually picked to be a small integer multiple (2 or 3) of the Krylov subspace size. Default = ***`300`***.

`[Linear Solver Normalized Residual Tolerance =` *`float`*`]`

Tolerance for the `Convergence Norm` in solving the linear system. Default = $\mathbf{10^{-4}}$.

Chemistry Specifications

The "`Chemistry Specifications`" section is mandatory. The parameters of this section are not relevant to *`porous_flow`* problem types, and attempts to include irrelevant equation terms will result in an error. That is, the parameters directing inclusion of source terms and thermal diffusion must be set to `off`. However, these parameters are relevant to the species transport equations (problem type *`mass_conv_diff`*). See Salinger et al. (1996) for further details.

Example:

```
-------------------------------------------------------------
     Chemistry Specifications
-------------------------------------------------------------
Energy equation source terms      = off
Species equation source terms     = off
Pressure (atmospheres)            = 1.0
Thermal Diffusion                 = off
Multicomponent Transport          = mixture_avg
Chemkin file                      = chem.asc
Surface chemkin file              = surf.asc
Transport chemkin file            = tran.asc
```

Description and Usage

`[Energy equation source terms = {`*`on`*`|`***`off`***`}]`

This flag controls inclusion of the energy source term due to chemical reactions. The value of the source term is controlled via the *`Q_VOLUME`* or *`Q_VOLUME_VAR`* data input line in the `Material Property Specifications` section of the input file.

`[Species equation source terms = {`*`on`*`|`***`off`***`}]`

This flag controls the chemical reactions in the interior of the domain. The value of the source term is controlled via the *`Y_VOLUME`* or *`Y_VOLUME_VAR`* data input line in the `Material Property Specifications` section of the input file. Surface reactions are defined separately through the `Boundary Condition Specifications` section.

`[Pressure (atmospheres) = `*`float`*`]`

This parameter specifies the thermodynamic pressure. Default = ***`1.0`*** (this value is in atmospheres and corresponds to 1.01325e+06 Pascal)**.**

`[Thermal Diffusion = {`*`on`*`|`***`off`***`}]`

This flag controls inclusion of thermal diffusion, also known as the Soret effect.

`[Multicomponent Transport = {`***`mixture_avg`***`|`*`mixture_avg_vc`*`|`*`dixon_lewis`*`}]`

The only formulation currently available for multicomponent transport is mixture-averaged diffusion..

`[Chemkin file = `*`string`*`]`

This specifies the chemkin file to use. If no value is specified then a default of **`chem.asc`** is set. If *`chem.bin`* or *`chem.bin.ws`* are specified then the default file (**`chem.asc`**) is also used.

`[Surface chemkin file = `*`string`*`]`

This specifies the surface chemkin file to use. If no value is specified then a default of ***`surf.asc`*** is set. If *`surf.bin`* or *`surf.bin.ws`* are specified then the default file (***`surf.asc`***) is also used.

`[Transport chemkin file = `*`string`*`]`

This specifies the chemkin file to use. If no value is specified then a default of **`tran.asc`** is set. If *`tran.bin`* or *`tran.bin.ws`* are specified then the default file (**`tran.asc`**) is also used.

Porous Flow Specifications

The "`Porous Flow Specifications`" section is used to identify additional numerical options specific to the solution of *porous_flow* problem types. It allows the user to specify whether or not binary gas diffusion terms are included in the governing equations, and also allows the user to invoke the <u>C</u>ontrol <u>V</u>olume <u>F</u>inite <u>E</u>lement <u>M</u>ethod (CVFEM) in lieu of Galerkin finite elements. This can be helpful in simulations involving a moving boundary between one- and two-phase systems and for convection dominated systems. In general, the CVFEM method will often require fewer iterations than Galerkin finite elements at the expense of numerical diffusion.

Example:

```
------------------------------------------------------------
     Porous Flow Specifications
------------------------------------------------------------
Binary Diffusion                = on
CVFEM                           = on
Flux Weighting                  = upwind
Lobatto Quadrature              = on
```

Description and Usage

`[Binary Diffusion =` {*`on`*|***`off`***}`]`

Flag to determine whether or not the binary gas diffusion terms are computed.

`[CVFEM =` {*`on`*|***`off`***}`]`

Flag to determine whether <u>C</u>ontrol <u>V</u>olume <u>F</u>inite <u>E</u>le<u>m</u>ents are used in lieu of Galerkin finite elements. If CVFEM is *on*, then the `Flux Weighting` and `Lobatto Quadrature` specifications will influence the solution, even if the default values are used. Also, the mass lumping option is invoked regardless of the `Mass Matrix Formulation Weight` parameter identified in the Solver Specifications section.

`[Flux Weighting =` {***`upwind`***|*`midpoint`*|*`harmonic`*}`]`

This parameter controls how the fluxes through the control volumes are computed. Note that this parameter is relevant only if the CVFEM is invoked, otherwise it is ignored.

`upwind`	This may introduce numerical diffusion into the solution.
`midpoint`	This is a second-order accurate solution and recovers the Galerkin finite elements solution.
`harmonic`	This option will over-weight the smaller transmissivities and will tend to slow down an infiltrating front.

`[Lobatto Quadrature =` ***`on`***|*`off`*`]`

Flag to determine whether numerical integration (quadrature) is performed using Lobatto quadrature, or Gaussian quadrature. The default value of ***`on`*** invokes Lobatto quadrature, which may be helpful in reducing negative transmissivities that can arise with hexahedral elements. Note that this parameter is relevant only if the CVFEM is invoked, otherwise it is ignored.

Auxiliary Input Specifications

This "`Auxiliary Input Specifications`" section is used to define auxiliary variables, which are nodal variables within an existing EXODUS II file. These values may be used to define material properties including intrinsic permeability, porosity, and parameters associated with a number of relative permeability models. Auxiliary variables may also define a velocity field for problems involving species transport. This capability allows the user to create a fully heterogeneous material by supplying properties generated with an external geostatistical software package such as GSLIB (Deutsch & Journel, 1998). This section is optional.

Example:

```
------------------------------------------------------------
     Auxiliary Input Specifications
------------------------------------------------------------
Auxiliary input file          = /tmp/mccord_auxinp.exoII
Auxiliary variable time index = 1
Number of auxiliary nodal variables = 7
Auxiliary nodal variable name(s):
  PERM
  PORO
  ALPHA
  BETA
  SL_RES
  SG_RES
  KREL_G_OPT
```

Description and Usage

[`Auxiliary input file =` *`string`*]

This is the name of the EXODUS II file containing the nodal values of the variables of interest. This EXODUS II file may be generated using the utility **add_var** described in the ***Other Miscellaneous Files*** section of this report, or any other program that generates nodal variables in the EXODUS II format. The mesh definition in this file must match that defined in the `Input FEM file` of the "`General Problem Specifications`" section. The path is optional if the file is located in the same directory as the PorSalsa executable.

[`Auxiliary variable time index =` *`integer`*]

This specifies which time index in the EXODUS II database contains the desired values for the auxiliary input variables. The default value is ***`1`***.

[`Number of auxiliary nodal variables =` *`integer`*]

This indicates how many auxiliary nodal variables are to be read from the auxiliary input file, and this number must match the number of names specified below. Default = ***`0`***.

`[Auxiliary nodal variable name(s): {`*`See Options Below`*`}]`

This line is followed by a list of variable names with one name per line. The number of names must match that of `Number of auxiliary nodal variables`. The quantities that may be defined by auxiliary variables are shown below. The first entry on a line represents the default nodal variable name.

Intrinsic Permeability

PERM	Isotropic Intrinsic Permeability, k	[m^2]
PERM_1	Orthotropic Intrinsic Permeability, k_{xx}	[m^2]
PERM_2	Orthotropic Intrinsic Permeability, k_{yy}	[m^2]
PERM_3	Orthotropic Intrinsic Permeability, k_{zz}	[m^2]

Porosity

PORO	Porosity	[-]

Unsaturated Material Properties

SL_RES	Residual Liquid Saturation, S_{wr}	[-]
SG_RES	Residual Gas Saturation, S_{gr}	[-]
ALPHA	van Genuchten Model Parameter, α	[m^{-1}]
BETA	van Genuchten Model Parameter, β	[-]
KREL_G_OPT	van Genuchten Model, Option for Computing Relative Permeability to the Gas Phase	
SIGMA	Cubic Model Parameter, σ	[N/m]
COEF1	Cubic Model Coefficient 1, c_1	[-]
COEF2	Cubic Model Coefficient 2, c_2	[-]
COEF3	Cubic Model Coefficient 3, c_3	[-]
P_ENTRY	Brooks-Corey Model, Entry Pressure Head, P_{entry}	[m]
LAMBDA	Brooks-Corey Model Parameter, λ	[-]

Note: See the subsection on ***Characteristic Curves*** (page 61) for a description of these models and their associated parameters.

Velocity

VEL_X	x-component of velocity	[m/s]
VEL_Y	y-component of velocity	[m/s]
VEL_Z	z-component of velocity	[m/s]

Notes:

- The auxiliary variables listed in this section of the input file correspond to those specified in the “`Material ID Specifications`” section, as described below.
- Comment lines may be placed within the auxiliary variable name listing. Any line beginning with the character “#” is interpreted as a comment line.
- If orthotropic permeability is specified {*`PERM_1`* & *`PERM_2`* [& *`PERM_3`*]}, then the number of permeability parameters must correspond to the dimensionality of the problem.
- The following section titled ***Material ID Specifications*** gives a description of the available unsaturated material property models available in PorSalsa. Specification of a particular model is done in the “`Material ID Specifications`” section of the input file. Each model has a required set of parameters, all of which must be present, but in any order. If homogeneous unsaturated material properties are desired then the parameter values may be specified via “`Data Specification for User’s Functions`” section of the input file rather than via auxiliary variables.

Material ID Specifications

The “`Material ID Specifications`” section is used to define the material properties associated with all materials in the spatial domain. This includes intrinsic material properties such as permeability, porosity, and matrix compressibility, as well as the models to be used for unsaturated material properties (capillary pressure – saturation – relative permeability functions). For *porous_flow* problem types, this section is also where the user identifies which of the governing equations to solve. Currently, each material block must solve the same set of governing equations, however the user may specify unique initial conditions for each material in the system

Multiple material subsections may exist, each with a list of parameter specifications. A material identification number is associated with each material definition. The material definition subsection includes a list of element block identification numbers defined in the EXODUS II mesh file that are to be assigned this material type. Each element block in the EXODUS II mesh must be assigned to a material.

Notes:

- Comment lines may be placed anywhere following the `Number of Material Properties` parameter specification line. Any line beginning with the character “#” is interpreted as a comment line.
- Blank lines and white space are ignored.
- An initial condition value must be given for each of the governing equations. However, the user may override this value using the “`Initial Guess/Condition Specifications`” section.

Example 1: Water Transport Using Homogeneous Material Properties

```
------------------------------------------------------------
     Material ID Specifications
------------------------------------------------------------
Number of Materials              = 1

POROUS_MEDIUM                    = 0   "Zone_1"
         G_VECTOR        =  0.0 0.0 9.8
         PERM            =  5.8e-12
         POROSITY        =  0.278
         MATRIX_COMPRESS =  8.e-8
         T_INIT          =  20.
         PA_INIT         =  9.766e4
         WATER
           SAT_INIT      =  0.308
         ELEM_BLOCK_IDS  =  1
END Material ID Specifications
```

Example 2: Water Transport Using Auxiliary Input

```
------------------------------------------------------------
     Material ID Specifications
------------------------------------------------------------
Number of Materials              = 2

POROUS_MEDIUM            = 0   "Zone_1"
         G_VECTOR               =  0.0 0.0 9.8
         PERM                   =  AUX_INP=PERM
         REL_PERM_FUNC          =  VG_krel_auxinp
         POROSITY               =  AUX_INP=PORO
         MATRIX_COMPRESS        =  8.e-8
         CAP_PRESS_FUNC         =  VG_cap_press_auxinp
         WATER
         T_INIT                 =  20.
         PA_INIT                =  9.766e4
         SAT_INIT               =  0.308
         ELEM_BLOCK_IDS         =  10
END
POROUS_MEDIUM            = 1   "Zone_2"
         PERM                   =  AUX_INP=PERM
         REL_PERM_FUNC          =  VG_krel_auxinp
         POROSITY               =  AUX_INP=PORO
         MATRIX_COMPRESS        =  8.e-8
         CAP_PRESS_FUNC         =  VG_cap_press_auxinp
         WATER
         SAT_INIT               =  0.370
         T_INIT                 =  20.
         PA_INIT                =  9.766e4
         ELEM_BLOCK_IDS         =  20
END
```

Example 3: Two-dimensional Species Transport

```
------------------------------------------------------------
     Material ID Specifications
------------------------------------------------------------
Number of Materials              = 1

POROUS_MEDIUM                    = 0   "Material A"
         DENSITY           =  1.0
         U_INIT            =  0.5
         V_INIT            =  0.0
         MOIST_CONT        =  1.0
         DISP_LONG         =  0.01
         NUM_SPECIES       =  1
         SPECIES_NAME 1  C
         DIFF_COEFF      C =  1.0e-13
         WTSPECIES       C =  1.0
         ELEM_BLOCK_IDS  =  1
END Material ID Specifications
```

Description and Usage

{*POROUS_MEDIUM*|*ANISOTROPIC_POROUS_MEDIUM*} = *integer string*

This line identifies the beginning of a material subsection. For *porous_flow* problem types, the allowable material type is either *POROUS_MEDIUM* or *ANISOTROPIC_POROUS_MEDIUM*, depending on whether the permeability of the material is isotropic or orthotropic (anisotropic but with zero off-diagonals).

POROUS_MEDIUM	Material isotropic porous medium
ANISOTROPIC_POROUS_MEDIUM	Material is anisotropic porous medium
integer	Material identification number (*The first material should be 0 (zero) rather than 1)*
string	User-selected name of material, in quotes

G_VECTOR = *float1 float2 float3* [m^2/sec]

Gravitational vector with three components (g_x = *float1*, g_y = *float2*, g_z = *float3*). This defines the gravitational vector for the entire problem domain, and therefore this line may appear only once in the "Material ID Specifications" section. For two-dimensional problems, a third float value is ignored. The float values may be separated by white space (blanks or tabs), or commas. The default setting is ***0.0 0.0 0.0***.

[REF_PLANE = *float1 float2 float3 float4*]

Coefficients defining a reference plane for the calculation of hydraulic head values, see Appendix C. A reference plane is required for use of hydraulic head as an initial condition, constant head boundary conditions, or the auxiliary output of hydraulic head. The reference plane must be orthogonal to the gravitational vector. As with the gravitational vector, this reference plane may only be defined once. Further, the gravitational vector definition must precede that of reference plane. The plane is defined by the equation:

Reference Plane: $Ax + By + Cz + D = 0$

The float values specified by this parameter correspond to:

`float1`	Reference plane coefficient, *A*, Default = **`0.0`**	[-]
`float2`	Reference plane coefficient, *B*, Default = **`0.0`**	[-]
`float3`	Reference plane coefficient, *C*, Default = **`1.0`**	[-]
`float4`	Reference plane coefficient, *D*, Default = **`0.0`**	[m]

Reference Plane Default: ***z = 0***

Notes:

- The reference plane must be defined prior to specification of hydraulic head initial conditions, hydraulic head boundary conditions, or hydraulic head as an auxiliary output variable.

Material Properties

The following parameters specify various intrinsic and effective properties of the porous medium. Most values reflect the properties of the material alone, while others represent effective properties, *i.e.*, the material and fluids combined. This distinction is indicated in the descriptions.

`[CP = {float|VARIABLE_PROP}]` [J/kg-K]

Heat capacity (specific heat) of the porous medium (Eqn. 4, $e_s = C_p T$) which must be defined if the energy equation is to be solved. Supplying a `float` value results in a constant, homogeneous value for all element blocks assigned to this material. Specification of `VARIABLE_PROP` invokes the user function `user_cp` in the file `rf_user_Cp.c`. See the section ***Programming User Functions*** for function details.

`[THERMAL_CONDUCT = {float|VARIABLE_PROP}]` [W/m-K]

Effective thermal conductivity of the porous medium and fluid system, which must be specified if the energy equation is to be solved. Supplying a `float` value results in a constant, homogeneous value for all element blocks assigned to this material. Specification of `VARIABLE_PROP` invokes the user function `user_cond` in the file `rf_user_cond.c`. See the section ***Programming User Functions*** for function details.

`[DENSITY = {float|VARIABLE_PROP}]` [kg/m^3]

Grain density of the porous skeleton (not including the fluids), which must be specified if the energy equation is to be solved. Supplying a `float` value results in a constant, homogeneous value for all element blocks assigned to this material. Specification of `VARIABLE_PROP` invokes the user function `user_density` in the file `rf_user_density.c`. See the section ***Programming User Functions*** for function details.

`POROSITY = {`*`float|AUX_INP=string`*`}` [-]

Porosity of the porous medium. Supplying a *`float`* value results in a constant, homogeneous value for all element blocks assigned to this material. Specification of *`AUX_INP=string`*, where *`string`* is the name of an auxiliary variable, invokes the use of the auxiliary input capabilities. In this case the porosity is defined at each node according to the EXODUS II nodal variable in the specified `Auxiliary input file`. The nodal variable name specified by *`string`* must be included in the list of `Auxiliary nodal variable name(s):`. If no *`string`* value is given, the default value of ***`PORO`*** is assumed. See the ***Auxiliary Input Specifications*** section for more information on this capability.

`MATRIX_COMPRESS = `*`float`* [1/Pa]

Matrix compressibility of the porous formation. See Appendix B for a discussion of the formulation of matrix compressibility used in PorSalsa and how this value relates to standard formulations of aquifer storativity.

`PERM = {See Options Below}` [m^2]

Intrinsic permeability of the porous medium which may be homogeneous or heterogeneous, as well as isotropic or orthotropic (anisotropic but with zero off-diagonals). Heterogeneous intrinsic permeability is handled via auxiliary input; see the ***Auxiliary Input Specifications*** section for additional information. Relative permeability is described in the following section on ***Unsaturated Hydraulic Properties***.

Isotropic Permeability

`PERM = {`*`float|AUX_INP=string`*`}`

Supplying a *`float`* value results in a constant, homogeneous value for all element blocks assigned to this material. Specification of *`AUX_INP=string`*, where *`string`* is the name of the auxiliary variable representing permeability, invokes the use of the auxiliary input capabilities. In this case the permeability is defined at each node according to the EXODUS II nodal variable in the specified `Auxiliary input file`. The nodal variable name specified by *`string`* must be included in the list of `Auxiliary nodal variable name(s):`. If no *`string`* value is given, the default value of ***`PERM`*** is assumed.

Orthotropic Permeability

`PERM = {`*`float1 float2 float3|AUX_INP=string1 string2`*` [`*`string3`*`]}`

Supplying *`float1 float2 float3`* results in a constant, homogeneous value for all element blocks assigned to this material where k_{xx} = *`float1`*, k_{yy} = *`float2`* and k_{zz} = *`float3`*. Specification of *`AUX_INP=string1 string2`* `[`*`string3`*`]`, invokes the use of the auxiliary input capabilities. In this case the permeability is defined at each node according to the EXODUS II nodal variables in the specified `Auxiliary input file`. The nodal variable names specified by the *`string#`* values must be included in the list of `Auxiliary nodal variable name(s)` and have default values of ***`PERM_1`***, ***`PERM_2`***, and ***`PERM_3`***. The number of *`float`* or *`string`* values must match the dimensionality of the problem.

Unsaturated Hydraulic Properties

Models used to compute capillary pressure – saturation – relative permeability relationships must always be specified. Several pre-coded models are available, and the user may also provide custom functions via the user function capabilities. The parameters needed by the models may be specified either through function data specifications within the main input file (see the ***Function Data Specifications*** section), or via auxiliary input variables (see the ***Auxiliary Input Specifications*** section). Using the former method results in homogeneous properties throughout the material, while the latter option provides a means of defining heterogeneous unsaturated material properties.

Notes:

- There is no internal check that the same model is used for both the relative permeability – saturation calculation and the capillary pressure – saturation calculation.
- None of the currently available models allows for hysteretic behavior.

`[REL_PERM_FUNC =` {*`See Options Below`*}`]`

This parameter specifies which functional model will be used to compute ***relative permeabilities*** (liquid and gas***)*** for nodes where the state of the system is partially saturated.

`user_krel_lg`	User-supplied relative permeability function. The data passed to this user function includes temperature, liquid saturation, nodal coordinates, and the material property structure associated with the current node. *Notes*: This function is in the file *`rf_user_krel_lg.c`* and may be edited by the user; a relative permeability must be computed for both the liquid and gas phase. PorSalsa must be recompiled if any changes are made to the source code.
`VARIABLE_PROP`	Same as *`user_krel_lg`*.
`VG_krel`	Standard van Genuchten (van Genuchten, 1978) relative permeability functions with constant parameters supplied through the function data capability described in the ***Function Data Specifications*** section on page 59.
`VG_krel_auxinp`	Standard van Genuchten relative permeability functions with parameters supplied through the auxiliary nodal

	variables (*`SL_RES`*, *`SG_RES`*, *`ALPHA`*, *`BETA`*, *`KREL_G_OPT`*). These nodal variable names must be defined in the "`Auxiliary Input Specifications`" section of the input file.
`BC_krel`	Brooks-Corey (Brooks and Corey, 1966) relative permeability functions with constant parameters supplied through the function data capability described in the ***Function Data Specifications*** section.
`BC_krel_auxinp`	Brooks-Corey relative permeability functions with parameters supplied through the auxiliary nodal variables (*`SL_RES`*, *`SG_RES`*, *`P_ENTRY`*, *`LAMBDA`*). These nodal variable names must be defined in the "`Auxiliary Input Specifications`" section of the input file.
`Cubic_krel`	Cubic relative permeability functions (Udell and Fitch, 1985) with constant parameters supplied through the function data capability described in the ***Function Data Specifications*** section.
`Cubic_krel_auxinp`	Cubic relative permeability functions with parameters supplied through the auxiliary nodal variables (*`SL_RES`*, *`SG_RES`*, *`SIGMA`*, *`COEF1`*, *`COEF2`*, *`COEF3`*). These nodal variable names must be defined in the "`Auxiliary Input Specifications`" section of the input file.

`[CAP_PRESS_FUNC = {`*`See Options Below`*`}]`

This parameter specifies which model will be used to compute the capillary pressure – saturation function for nodes where the state of the system is partially saturated.

`user_cap_press`	User-supplied capillary pressure – saturation function. The data passed to this user function includes liquid saturation, nodal coordinates, and the material property structure. *Notes*: This function is in the file *`rf_user_cap_press.c`* and may be edited by the user. The code must be recompiled if any changes are made to the source code.
`VARIABLE_PROP`	Same as *`user_cap_press`*.
`VG_cap_press`	Standard van Genuchten (van Genuchten, 1978) capillary pressure – saturation function with constant parameters

	supplied through the function data capability described in the ***Function Data Specifications*** section.
VG_cap_press_auxinp	Standard van Genuchten capillary pressure – saturation function with parameters supplied through the auxiliary nodal variables (*SL_RES*, *SG_RES*, *ALPHA*, *BETA*). These nodal variable names must be defined in the "`Auxiliary Input Specifications`" section of the input file.
BC_cap_press	Brooks-Corey (Brooks and Corey, 1966) capillary pressure – saturation function with constant parameters supplied through the function data capability described in the ***Function Data Specifications*** section.
BC_cap_press_auxinp	Brooks-Corey capillary pressure – saturation function with parameters supplied through the auxiliary nodal variables (*SL_RES*, *SG_RES*, *P_ENTRY*, *LAMBDA*). These nodal variable names must be defined in the "`Auxiliary Input Specifications`" section of the input file.
Cubic_cap_press	Cubic capillary pressure – saturation function (Udell and Fitch, 1985) with constant parameters supplied through the function data capability described in the ***Function Data Specifications*** section.
Cubic_cap_press_auxinp	Cubic capillary pressure – saturation function with parameters supplied through the auxiliary nodal variables (*SL_RES*, *SG_RES*, *SIGMA*, *COEF1*, *COEF2*, *COEF3*). These nodal variable names must be defined in the "`Auxiliary Input Specifications`" section of the input file.

Governing Equation Specifications

Identifying which of the governing conservation equations to solve for a *porous_flow* problem type is also done within the "`Material ID Specifications`" section. Initial conditions for all three equations (water, air, and energy) are specified here, regardless of whether or not a particular governing equation is being solved; the additional information is needed for the computation of constitutive relationships and equations of state.

Governing conservation equations to be solved are identified with the following input lines:

[*WATER*]	Solve the transport equation for water
[*AIR*]	Solve the transport equation for air
[*ENERGY*]	Solve the transport equation for energy

Parameter specifications for initial conditions are as follows:

```
string = float
```

where `string` identifies the governing equation to which the initial value, specified by `float`, applies. The options for `string` are given in Table 3 and, for user convenience, allows specification of the primary solution variables, or secondary variables. When a secondary variable is supplied, the equivalent primary solution variable is computed.

Notes:

- Specification of `HEAD_INIT` computes an equivalent water density using the predefined subroutine `f_head_init_cond`. This function requires no function data. The equivalent water density is computed using the hydraulic head, which is relative to the reference plane defined by `REF_PLANE`, and the initial temperature specified by `T_INIT`, see Appendix C. Care must be taken to ensure that other settings do not violate the implicit assumptions that the entire domain is within the saturated zone. See the ***Initial Guess/Condition Specifications*** section of this report for further details.
- Specification of `PL_INIT` computes an equivalent water density through the use of the equations of state. The value computed is referenced to the initial temperature and air pressure. Care must be taken to ensure that other settings do not violate the implicit assumptions that the entire domain is within the saturated zone. A fatal error will occur if any node in the domain is found to be only partially saturated.
- Although the initial conditions are specified on a material basis in the "`Material ID Specifications`" section, these may be over-written by a function or constant value as described in the ***Initial Guess/Condition Specifications*** section of this report.

`string`	*Governing Equation*	*Variable Specified*	*Units*
Initial Conditions for Primary Variables			
`WD_INIT`	Water	Water Density, ρ_w	*kg/m³*
`PA_INIT`	Air	Partial Pressure of Air, P_{ag}	*Pascals*
`T_INIT`	Energy	Temperature, T	*ºCelsius*
Initial Conditions for Secondary Variables			
`SAT_INIT`	Water	Water Saturation, S_l	-
`HEAD_INIT`	Water	Hydraulic Head, h	*meters*
`PL_INIT`	Water	Liquid Pressure, P_l	*Pascals*
`PG_INIT`	Air	Total Gas Pressure, P_g	*Pascals*

Table 3: Initial condition specifiers and their corresponding governing equations and associated variable.

Volumetric Source Terms

[*string1* = *float*|*string2*]

Volumetric mass and energy sources (kg/m^3-sec or W/m^3) are applied on a material basis and may be supplied for any of the governing equations being solved. More information on user functions can be found in the section ***Programming User Functions***. Options available for *string1* include:

W_VOLUME	Constant volumetric source rate (kg/m^3-sec) of water = *float*.
W_VOLUME_VAR	Volumetric source rate of water computed from the user function specified by *string2*.
A_VOLUME	Constant volumetric source rate (kg/m^3-sec) of air = *float*.
A_VOLUME_VAR	Volumetric source of air computed from the user function specified by *string2*.
Q_VOLUME	Constant volumetric source rate (W/m^3) of heat = *float*.
Q_VOLUME_VAR	Volumetric source rate of heat computed from the user function specified by *string2*.

Properties Related to Species Transport

When solving problems involving the transport of dissolved species (*i.e.*, *mass_conv_diff* problem types), the following parameters are applicable.

U_INIT = {*float*|*AUX_INP=string*} [m/s]

This value specifies the Darcy velocity in the x-direction. Supplying a *float* value results in a constant, homogeneous value for the entire element block. Specification of *AUX_INP=string*, where *string* is the name of an auxiliary variable, invokes the use of the auxiliary input capabilities. In this case the parameter is defined on a node by node basis (possibly heterogeneous) and is taken from the specified Auxiliary input file. The parameter name specified by *string* must be included in the list of Auxiliary nodal variable name(s). If no value is specified, then a default name of ***VEL_X*** is assigned.

V_INIT = {*float*|*AUX_INP=string*} [m/s]

This value specifies the Darcy velocity in the y-direction. Supplying a *float* value results in a constant, homogeneous value for the entire element block. Specification of *AUX_INP=string*, where *string* is the name of an auxiliary variable, invokes the use of the auxiliary input capabilities. In this case the parameter is defined on a node by node basis (possibly heterogeneous) and is taken from the specified Auxiliary input file. The parameter name specified by *string* must be included in the list of Auxiliary nodal variable name(s). If no value is specified, then a default name of ***VEL_Y*** is assigned.

`[W_INIT =` {*`float`*|*`AUX_INP=string`*}`]` [m/s]

This value specifies the Darcy velocity in the z-direction, for three dimensional problems. Supplying a *`float`* value results in a constant, homogeneous value for the entire element block. Specification of *`AUX_INP=string`*, where *`string`* is the name of an auxiliary variable, invokes the use of the auxiliary input capabilities. In this case the parameter is defined on a node by node basis (possibly heterogeneous) and is taken from the specified `Auxiliary input file`. The parameter name specified by *`string`* must be included in the list of `Auxiliary nodal variable name(s)`. If no value is specified, then a default name of ***`VEL_Z`*** is assigned.

`[VELOCITY_VEC_FUNC =` {*`string`*|*`VARIABLE_PROP`*}`]` [-]

A function named by *`string`* will be used to compute the Darcy velocity vector. Specification of *`VARIABLE_PROP`* invokes the user function *`user_velocity`* in the file *`rf_user_velocity_fn.c`*. See the section on ***Programming User Functions*** for further details.

`MOIST_CONT =` {*`float`*|*`AUX_INP=string`*} [-]

This value specifies the moisture content of the porous medium. Supplying a *float* value results in a constant, homogeneous value for the entire element block. Specification of *AUX_INP=string*, where *string* is the name of an auxiliary variable, invokes the use of the auxiliary input capabilities. In this case the parameter is defined on a node by node basis (possibly heterogeneous) and is taken from the specified `Auxiliary input file`. The parameter name specified by *string* must be included in the list of `Auxiliary nodal variable name(s)`.

`DISP_LONG =` {*`float`*|*`AUX_INP=string`*} [m]

This value specifies the longitudinal dispersivity of the porous medium (Eqn. 7). Supplying a *`float`* value results in a constant, homogeneous value for the entire element block. Specification of *`AUX_INP=string`*, where *`string`* is the name of an auxiliary variable, invokes the use of the auxiliary input capabilities. In this case the parameter is defined on a node by node basis (possibly heterogeneous) and is taken from the specified `Auxiliary input file`. The parameter name specified by *`string`* must be included in the list of `Auxiliary nodal variable name(s)`.

`DISP_TRAN =` {*`float`*|*`AUX_INP=string`*} [m]

This value specifies the transverse dispersivity of the porous medium (Eqn. 7). Supplying a *`float`* value results in a constant, homogeneous value for the entire element block. Specification of *`AUX_INP=string`*, where *`string`* is the name of an auxiliary variable, invokes the use of the auxiliary input capabilities. In this case the parameter is defined on a node by node basis (possibly heterogeneous) and is taken from the specified `Auxiliary input file`. The parameter name specified by *`string`* must be included in the list of `Auxiliary nodal variable name(s)`.

`NUM_SPECIES =` *`integer`* [-]

This value specifies the number of dissolved species to be modeled. Up to 30 individual species may be specified, although this value may be further limited by memory constraints.

`SPECIES_NAME` *`integer string`* [-]

These parameters associate a name (as specified by *`string`*) with a species number (as specified by *`integer`*). The name may be up to 16 characters in length and cannot contain blanks. The species number must be less than or equal to the total number of species specified by `NUM_SPECIES`. Note: `NUM_SPECIES` must be specified prior to `SPECIES_NAME`.

`DIFF_COEFF` *`string float`* [m^2/sec]

This parameter specifies the effective diffusion coefficient (as specified by *`float`*) for the species named *`string`*. Note that the effective diffusion coefficient is equal to the free water diffusion coefficient multiplied by the tortuosity of the porous medium.

`WTSPECIES` *`string float`* [kg/mol]

This parameter specifies the molecular weight (as specified by *`float`*) for the species named *`string`*.

Volumetric Source Terms

[*`string1`* = *`float`*|*`string2`* {*`SINGLE`* | *`MULTIPLE`*}]

Molar production rates are applied on a material basis, and are defined as either a constant value or by a user function. More information on user functions can be found in the section ***Programming User Functions***. Options available for *`string1`* include:

`Y_VOLUME`	Constant molar production rate = *`float`*.
`Y_VOLUME_VAR`	Molar production rate computed from the user function specified by *`string2`*. When computing the production rate for a single species, specify *`SINGLE`* and use the function *`string2`* = `user_source`. When computing the production rate for multiple species, specify *`MULTIPLE`* and use the function *`string2`* = `user_source_multi`. See the MPSalsa User's Guide (Salinger, *et al*., 1996) for examples and more detail.

Close Material Definition

`ELEM_BLOCK_IDS =` *`integer1`* [*`integer2 integer3`* ...]

This parameter specifies which of the blocks of elements from the EXODUS II database named by `Input FEM file` correspond to the material defined here. Element blocks are identified by an integer and typically start at a value of 1.

Note that all element blocks in the domain must be assigned to a material. As well, no two materials may reference the same element block.

`END`

The word "`END`" terminates the parameter specifications for a given material.

Boundary Condition Specifications

The "`Boundary Condition Specifications`" section is where all non-default boundary conditions are specified. If no boundary condition is explicitly defined over a portion of the domain boundary, a Neumann boundary condition specifying a mass or energy flux of zero is applied.

Example:

```
------------------------------------------------------------
     Boundary Condition Specifications
------------------------------------------------------------
Number of Generalized Surfaces   = 0
Number of BC  = 4
BC = WD_BC DIRICHLET NS 10 INDEPENDENT  486.36    0
BC = PA_BC DIRICHLET NS 10 INDEPENDENT    7.021E4 0
BC =  T_BC DIRICHLET NS 10 INDEPENDENT   70.0     0
BC =  T_BC NEUMANN   SS 20 INDEPENDENT -100.0     0
```

Description and Usage

`Number of Generalized Surfaces   =` *`integer`*

A generalized surface is a side set in the EXODUS II file for which the outward normal and tangential vectors are defined in this section. This type of boundary condition is not currently applicable to problems of type *porous_flow*. Details are given in the MPSalsa User's Guide.

`Number of BC =` *`integer`*

This parameter specifies the number (as given by *`integer`*) of explicit boundary condition specifications (*i.e.*, lines of `BC =` ... specifications, see description below).

Boundary condition specifications are of the form:

`BC =` *`bc_var bc_type set_type set_id dependence_flag bc_values data_lines`*

where,

`bc_var`	= {*WD_BC*|*PA_BC*|*T_BC*|*SAT_BC*|*HEAD_BC*|*PW_BC*|*PG_BC*|*Y_BC*}
`bc_type`	= {*DIRICHLET*|*NEUMANN*|*MIXED*|*ATMOSPHERIC*}
`set_type`	= {*NS*|*SS*|*GS*}
`set_id`	= *integer*
`dependence_flag`	= {*INDEPENDENT*|*DEPENDENT*}
`bc_values`	= *See Description Below*
`data_lines`	= *integer*

Descriptions of the various boundary condition specification parameters are as follows:

`bc_var`

This parameter identifies which of the three governing equations the boundary condition is applied to. If the boundary condition specification refers to a secondary variable, this value is used to compute an equivalent boundary condition applied to the relevant primary variable. Table 4 lists the options for `bc_var` along with the equation to which the boundary condition applies, and the variable being supplied. Specialized boundary conditions including hydraulic head, liquid pressure, and energy slave boundaries are presented at the end of this section.

`bc_var`	*Governing Equation*	*Variable Specified*
Boundary Conditions for Primary Variables		
WD_BC	Water	Water Density, ρ_w
PA_BC	Air	Partial Pressure of Air, P_{ag}
T_BC	Energy	Temperature, T
Boundary Conditions for Secondary Variables		
SAT_BC	Water	Water Saturation, S_l
HEAD_BC	Water	Hydraulic Head, h
PW_BC	Water	Liquid Pressure, P_l
PG_BC	Air	Total Gas Pressure, P_g

Table 4: Boundary condition variable specifiers (`bc_var`) and their corresponding governing equations and associated variable.

bc_type

The type of boundary condition being applied is specified through `bc_type`. Options include

DIRICHLET	First-type boundary condition of the form: $$y = f\left(\underline{x}, t, primary\ variables\right)$$ Dirichlet boundary conditions may be applied to side sets or node sets.
NEUMANN	Second-type boundary condition of the form: $$n \cdot q_\alpha = f\left(\underline{x}, t, primary\ variables\right)$$ Neumann boundary conditions may be only applied to side sets.

`MIXED`	Third-type boundary condition of the form: $$n \cdot q_\alpha = f_1(\underline{x}, t, primary\ \ variables) \cdot (y - y_0) + a \cdot f_2(\underline{x}, t, primary\ \ variables)$$ Mixed boundary conditions may be only applied to side sets.
`ATMOSPHERIC`	This specifies a boundary exposed to a gaseous atmosphere at a specified temperature, pressure, and relative humidity. This type of boundary condition is useful for representing seepage faces, or other exposed surfaces. See Appendix D for further details.

`set_type`

This parameter identifies the type of boundary with respect to the EXODUS II file containing the finite element mesh. A value of `NS` indicates that the boundary condition applies to a node set, while `SS` specifies that the boundary condition applies to a side set.

`set_id`

The identification number of the node set or side set assigned in the EXODUS II file is specified by `set_id`.

`dependence_flag`

This flag is used to specify whether or not the boundary condition is a function of the primary variables (as specified by `DEPENDENT`, otherwise `INDEPENDENT`). It controls whether or not a Jacobian matrix entry should be computed for this BC.

`bc_values`

This parameter(s) is used to specify the numerical value(s) of the boundary condition. For Dirichlet or Neumann boundary conditions a single parameter is specified corresponding to $f(\underline{x}, t, primary\ variables)$. Options include:

{`float`|`string`}

`float`	Specified value. $f(\underline{x}, t, primary\ variables) = \texttt{float}$
`string`	Function name for the subroutine computing the boundary condition value.

For Mixed boundary conditions, four parameters must be specified corresponding to $f_1(\underline{x}, t, primary\ \ variables)$, y_0, a, and $f_2(\underline{x}, t, primary\ \ variables)$. Options include:

{`float1`|`string1`} {`float2`|`string2`} `float3` {`float4`|`string4`}

Table 5 lists the various boundary condition options and the associated value specified `bc_value`.

`float1`	Specified value. $f_1(\underline{x},\ t, primary\ \ variables) = \texttt{float1}$

string1	Function name for the subroutine computing the boundary condition value. $f_1(\underline{x}, t, \textit{primary variables}) = \texttt{string1}$
float2	Specified value. $y_0 = \texttt{float2}$
string2	Function name for the subroutine computing the boundary condition value. $y_0 = \texttt{string2}$
float3	Specified constant, $a = \texttt{float3}$
float4	Specified value. $f_2(\underline{x}, t, \textit{primary variables}) = \texttt{float4}$
string4	Function name for the subroutine computing the boundary condition value. $f_2(\underline{x}, t, \textit{primary variables}) = \texttt{string4}$

data_lines

Data may be passed to boundary condition functions by using BC_DATA input lines. This parameter tells how many lines of BC_DATA follow. The format of the BC_DATA input line is

[BC_DATA = {*See Options Below*}]

Parameter values passed to a function used to compute a boundary condition are supplied line by line according to the type of data (integer, floating point, or function name). A "#" in the first column may be used to indicate a comment line and blank lines are ignored. Descriptions of the parameter specification formats for these types of data are as follows.

FUNCTION|FUNC string

User function name specified by *string*.

INT|INTEGER integer [*integer integer* ...]

Integer parameter data. At least one value is expected.

FLOAT *float* [*float float* ...]

Floating point parameter data. At least one value is expected. As indicated, this is the default type.

`bc_var`	`bc_type`	*Value(s) Specified*	*Units*
`WD_BC`	`DIRICHLET`	Water Density, ρ_w	*kg / m*^{3}
`SAT_BC`	`DIRICHLET`	Liquid Saturation, S_l	*none*
`HEAD_BC`	`DIRICHLET`	Hydraulic Head, h	*meter*
`PW_BC`	`DIRICHLET`	Liquid Pressure, P_l	*Pascal*
`PA_BC`	`DIRICHLET`	Air Pressure, P_{ag}	*Pascal*
		(or Partial Pressure)	
`PG_BC`	`DIRICHLET`	Total Gas Pressure, P_g	*Pascal*
`T_BC`	`DIRICHLET`	Temperature, T	*ºCelsius*
`WD_BC`	`NEUMANN`	Water Flux, q_w	*kg / m*^{2}*-sec*
`PA_BC`	`NEUMANN`	Air Flux, q_a	*kg / m*^{2}*-sec*
`T_BC`	`NEUMANN`	Energy Flux, q_e	*Joule / m*^{2}*-sec*
`WD_BC`	`MIXED`	Multiplier	*m / sec*
		Water Density, ρ_w	*kg / m*^{3}
		Multiplier	*none*
		Water Flux, q_w	*kg / m*^{2}*-sec*
`PA_BC`	`MIXED`	Multiplier	*kg / Newton-sec*
		Air Pressure, P_{ag}	*Pascal*
		Multiplier	*none*
		Air Flux, q_a	*kg / m*^{2}*-sec*
`T_BC`	`MIXED`	Multiplier	*ºCelsius-Joule / m*^{2}*-sec*
		Temperature, T	*ºCelsius*
		Multiplier	*none*
		Energy Flux, q_e	*Joule / m*^{2}*-sec*

Table 5: Allowable boundary condition specifiers, and the physical units associated with the user-supplied value.

Further Illustrative Examples:

```
BC = WD_BC DIRICHLET NS 1 INDEPENDENT  486.36    0
```

Specifies a water density of 486.36 kg/m^3 on node set 1.

```
BC = WD_BC NEUMANN SS 2 INDEPENDENT    1.0E-04 0
```

Specifies a water mass flux of 1.0E-04 kg/m^2-sec normal to side set 2.

```
BC = WD_BC DIRICHLET NS 1 INDEPENDENT  user_bc_exact    2
BC_DATA INT 1 3 5
BC_DATA FLOAT 986.0
```

Specifies a water density is to be computed with the user function *user_bc_exact* with the three integer values “1”, “3”, and “5” and one float value of “986.0” passed to the function. The Dirichlet boundary condition will be applied to node set 1.

```
BC = WD_BC MIXED SS 2 INDEPENDENT -0.2  998.0  7.0  user_bc_exact  0
```

Specifies a water mass flux of the form

$$n \cdot q_w = (-0.2)(\rho_w - 998.0) + (7.0) f_2(\underline{x}, t, \textit{primary varbiales})$$

where the user function *user_bc_exact* is used to compute the value of f_2. No additional function data is specified for *user_bc_exact* in this example.

Hydraulic Head Boundaries

The formulation and implementation of hydraulic head boundaries are described in Appendix C. A hydraulic head boundary condition may be applied to any node set of the domain boundary as a secondary variable for water density. The format of the input data line is

```
BC = HEAD_BC DIRICHLET NS set_id INDEPENDENT f_constant_head_bc 1
BC_DATA = bc_value
```

where,

set_id	=	*integer*
		The identification number of the node set to which the constant head boundary is being applied.
bc_value	=	*float*
		The hydraulic head value to be applied to the node set.

This specification will compute an equivalent water density using the predefined function *f_constant_head_bc*, which requires a single function data value (*bc_value*). The equivalent water density is computed from the hydraulic head, which is based on the reference plane (REF_PLANE defined in the “Material Property Specifications” section of the input file), and the current temperature for the equation of state calculations.

Note:

- There are no error checks to ensure that the state of the nodes in the node set remain fully saturated.

Liquid Pressure Boundaries

A liquid pressure boundary condition may be applied to a node set of the domain boundary as a secondary variable for water density. The format of the input data line is

```
BC = PW_BC DIRICHLET NS set_id INDEPENDENT f_liquid_pressure_bc 1
BC_DATA = FLOAT bc_value
```

where,

set_id	=	*integer*
		The identification number of the node set to which the constant liquid pressure boundary is being applied.
bc_value	=	*float*
		The liquid pressure to be applied to the node set.

This specification will compute an equivalent water density using the predefined function *f_liquid_pressure_bc*, which requires one float value (*bc_value*) specifying the liquid water pressure. It uses the current solution vector in its computations.

Note:

- A fatal error condition will occur if the state of any node in the node set becomes partially saturated.

Energy Slave Boundary Conditions

For nonisothermal simulations involving the injection of fluid into a boundary it is useful to be able to couple the boundary conditions for mass and energy. This has been implemented by designating a master boundary condition associated with mass injection, and a slave boundary condition for the energy equation. Given a specified temperature and pressure of the injected fluid, the enthalpy equation of state is used to compute the energy flux that results from the given mass flux of either water, or air, over the master boundary. Therefore, the master boundary condition must defined as NEUMANN on a designated side set, and must be applied to either the air or water equation (PA_BC or WD_BC). The energy slave boundary must follow with the boundary condition name ESL_BC. Two BC_DATA lines are required for the energy slave boundary condition; the first identifies the boundary condition number of the master, and the second lines defines the temperature and pressure of the injected fluid. The following example describes the format, and required parameters. The format of the master boundary data input line is

```
BC = WD_BC|PA_BC NEUMANN SS set_id INDEPENDENT mass_inj_rate 0
```

followed by that of the energy slave

```
BC = ESL_BC NEUMANN SS set_id INDEPENDENT f_heat_flux_slave_bc 2
BC_DATA = INT master_bc_num
BC_DATA = FLOAT inj_temp inj_press
```

where,

set_id	= *integer*
	The identification number of the side set to which the mass and energy injection is being applied. Note that both the master and slave boundary conditions must be applied to the same side set.
mass_inj_rate	= *float*
	The mass injection flux of the master node. Being a Neumann boundary condition, this value represents mass per area per time. Typically, this value will be negative if the normal vector to the side set points outward from the domain.
master_bc_num	= *integer*
	The number of the master boundary condition in terms of its order (1 … `Number of BC`) in the list of boundary conditions.
inj_temp	= *float*
	The temperature (^{o}C) of the injected fluid.
inj_press	= *float*
	The pressure (Pascals) of the injected fluid.

Notes:

- The master boundary condition must precede the slave boundary condition in the list of boundary conditions.

Atmospheric Boundaries

The formulation and implementation of a boundary condition that emulates a boundary exposed to the atmosphere is described in detail in Appendix D. This boundary type is suitable for surfaces exposed to a normal range of atmospheric conditions with temperatures above the freezing point. It is also suitable for the simulation of seepage boundaries where liquid water may flow out of an exposed surface if the gradient driving advective flow of liquid is sufficient. The format of the data input line is

BC = *string1* ATMOSPHERIC NS|SS *set_id* DEPENDENT *type*_SEEPAGE *coef1 coef2 coef3 s* P_{ref} T_{ref} $R_{H,ref}$ *n_lines*

where,

string1	= *WD_BC* | *PA_BC* | *T_BC* Identifies to which equation, water, air, or energy, the atmospheric boundary condition is being applied.
set_id	= *integer* The identification number of the node set (NS) or side set (SS) to which the atmospheric boundary is being applied.
type	= *SATURATED* | *UNSATURATED* Identifies conditions under which outward liquid advection may occur.
coef1,coef2,coef3	= ***DEFAULT*** | *float* Coefficients used in the formulation equations. See Appendix D for details.
s	= *float* Distance between the atmospheric reference point and surface node. See Appendix D for details.
P_{ref}	= *float* Pressure of the atmospheric reference point.
T_{ref}	= *float* Temperature of the atmospheric reference point.
$R_{H,ref}$	= *float* Relative humidity of the atmospheric reference point.
n_lines	= *integer* Number of additional BC_DATA lines.

Initial Guess/Condition Specifications

The section "`Initial Guess/Condition Specifications`" defines initial conditions for primary variables. Inputs in this section can be used to override initial conditions specified in the "`Material ID Specifications`" section of the input file, the default method of specifying initial values. One option is to read data from an EXODUS II file which contains the solution from a prior simulation, and thus provides the functionality of restarting a simulation. The nodal variable names defined in this EXODUS II file must be those representing the primary solution variables of the current simulation; the required nodal variable names for water density, air pressure, and temperature are `Wden`, `Pair`, and `Temp`, respectively. The user may also specify the time index of the EXODUS II file to be read for restart simulations.

Example:

```
------------------------------------------------------------
     Initial Guess/Condition Specifications
------------------------------------------------------------
Set Initial Condition/Guess     = constant
Apply function                  = no
Restart from file               =
Time Index to Restart From      = 1
```

Description and Usage

[`Set Initial Condition/Guess =` {*`See Options Below`*}]

`constant` [*`float`*]	Applies a constant initial condition of *`float`* to the primary unknowns. If a *`float`* value is not specified, the initial values specified in the "`Material ID Specifications`" section are used.
`random`	Applies a random initial condition sampled from a uniform distribution between 0 and 1 to the primary unknowns.
`exoII_file`	The initial conditions are read from an EXODUS II file. The default file is that named on the input line `Output FEM file` in the "`General Problem Specifications`" section. This provides the functionality of restarting a simulation. Note that some or all of the default EXODUS II file may be overwritten with the results of the current simulation, see the parameter `Time Index to Restart From` described below.

The default value is ***`constant 0.0`***.

`[Apply function = {`*`string`*`|`***`no`***`|`*`f_head_init_cond`*`}]`

A user-written function can be supplied to compute the initial condition. The user can either create a new function, named by *string*, or can modify the existing function, *`user_init_cond`*. This function is executed after the `Set Initial Condition/Guess` input line so the function can be dependent on a solution read in from an EXODUS II file. See the MPSalsa User's Guide for details on how to write this function.

Specification of *`f_head_init_cond`* will compute hydraulic head initial conditions for the water equation. A constant hydraulic head (in space) implies a variable density due to the compressibility of water. The constant head value to be applied is specified in the "`Material ID Specifications`" section of the input file via the parameter *`HEAD_INIT`*. The reference plane is defined by *`REF_PLANE`*, and the temperature of the system as *`T_INIT`*. See Appendix C for details on the formulation of hydraulic head in PorSalsa. Note that if a spatially variable hydraulic head initial condition is desired, then a user-supplied function will be required.

`[Restart from file = `*`string`*`]`

If `Set Initial Condition/Guess = `*`exoII_file`*, then the initial conditions will be read from the EXODUS II file named by *`string`* instead of the default value, which is the EXODUS II file named by the `Output FEM file` line in the "`General Problem Specifications`" section.

`[Time Index to Restart From = `*`integer`*`]`

This line specifies the index of the time step from which to perform restarts or take the initial condition. This parameter is only pertinent if the `Set Initial Condition/Guess` value is *`exoII_file`*; the default value is ***`1`***. Restarts for steady or time-varying problems can use any solution data defined on the same finite element mesh.

Output Specifications

The "`Output Specifications`" section is used to control output to the EXODUS II output file, including the frequency of output (for transient problems), and which of the primary solution variables will be written to the output file. If this section is not found , results will not be written to the output EXODUS II file. This section also allows the user to specify an exact solution function, which is used to compare computed results to those provided by the named function. The user may also provide a function to compute, and subsequently output, other quantities of interest.

Example:

```
------------------------------------------------------------
     Output Specifications
------------------------------------------------------------
User Defined Output                = yes
Parallel Output                    = no
Scalar Output                      = yes
Time Index to Output To            = 1
Nodal variable output times:
      every 8.64e+05 seconds
Number of nodal output variables   = 1
Nodal variable names:
      Water_Density
Test Exact Solution Flag           = 1
Name of Exact Solution Function    = f_xx_yy
```

Description and Usage

[`User Defined Output = `{***no***|*yes*}]

This flag specifies whether the function `user_out` should be called to output additional information to *stdout*. This capability allows user-customized output. The version of `user_out` distributed with PorSalsa has a number of function calls for reporting min, max values and average solution variables. See Salinger *et al*. (1996) for more information.

Note: If integrated surface fluxes of mass and/or heat are to be computed over specified side sets then this must be set to *yes*.

[`Parallel Output = `{***no***|*yes*}]

This flag specifies whether or not parallel output is to be performed. It can be used simultaneously with scalar output. See the MPSalsa User's Guide for more information.

[`Scalar Output = `{***no***|*yes*}]

This parameter specifies whether or not output to a scalar EXODUS II file is to be performed. The name of the file is specified in the `General Problem Specifications` section as the `Output FEM file`.

[`Time Index to Output To = `*integer*]

This line is relevant when restarting a simulation, and the user wishes to designate a time index in the EXODUS II output file where the output from the current simulation will first be written. If this line is not found, and the simulation is a restart, the output from the current simulation is appended to the end of the EXODUS II output file. If this line is found, the output from the current simulation will be written at the time index specified, **overwriting** the existing data. Note that the initial guess is always output first. It is therefore suggested that the value of `Time Index to Output To` be set equal to the `Time Index to Restart From` so as to preclude having the same set of values stored twice in the file. The default is to append results to the EXODUS II output file.

`Nodal variable output times:`

`every` *`integer steps`*

-or-

`every` *`float`* {*`units`*|*`seconds`*|*`minutes`*}

This line specifies how often output is processed during transient simulations. This processing includes writing primary solution variables to the output EXODUS II file, computing and writing any auxiliary variables, and calling a user-defined output function, if requested. The designation of seconds or minutes is purely symbolic since units of the parameters supplied to PorSalsa are implicitly defined; it is the user's responsibility to supply parameters with a consistent set of units. Therefore, the input line

`every` *`float`* {*`units`*|*`seconds`*|*`minutes`*}

forces output to be processed when the simulation time exceeds *`n`***`float`* for any integer *`n`*. Similarly, the next time step output will be the first to have a time value greater than (*`n`*+1)**`float`*. The default is to process output at every time step.

`Number of nodal output variables  =` *`integer`*

This parameter specifies how many primary variables to output. Default = ***0***.

`[Nodal variable names:]`

`[Water_Density]`

`[Air_Pressure]`

`[Temperature]`

`[Mass_Fraction]`

These parameters specify which of the primary variables to output. When written to the output EXODUS II file, the nodal variable names for `porous_flow` problems are `Wden`, `Pair`, `Temp`, respectively. The number of variable names listed must match the *integer* number specified above. It should also be noted that all primary variables must be written to the output file. Premature program termination will result from specifying only a subset of the primary variables. However, the variable names can be listed in any order.

For multiple species `mass_conv_diff` problems, all species are output by simply specifying `Mass_Fraction.` However, each species name (specified in the section *Properties Related to Species Transport* on page 41) must appear in the `Output FEM file` specified in the `General Problem Specifications` section.

`[Number of global output variables =` *`integer`*`]`

This line is used to specify the number of global variables that are to be output to the EXODUS II file. Global variables are single-valued variables that only have the single dimension of time. They are used to store timing and global solution information. Default is ***0***.

```
[Global variable names:]
      string(1)
      string(2)
         :
      string(N)
```

The names of global variables available for output are listed below, and the number of names listed must match the number specified by `Number of global output variables`.

`Time_index`	Current time step
`Delta_time`	Time increment between time steps
`Matrix_Fill_Time`	Time required to assemble the matrix at each time step
`Matrix_Solve_time`	Time required to solve the problem at each time step

These global variable names are case-insensitive. When creating the output EXODUS II file, the pre-processor must account for global variables with these names, and the routine user_out will be used to output values for these variables during execution.

`[Test Exact Solution Flag = {`***`0`***`|`*`1`*`|`*`2`*`} [`*`SUMMARY`*`]]`

This line specifies whether or not the computed solution should be tested against a known analytic solution. The comparison includes L2-norm and max-norm error computations. Additional information on the location of the maximum error and an estimate of the largest characteristic length of an element in the finite element mesh is provided.

`0`	Do not compare against an exact solution.
`1`	Test the numerical answer against the exact solution given by the function to be specified below.
`2`	Same as *`1`*, but in addition, if the output of the solution is desired, then also output the difference between the exact solution and the numerical solution
`SUMMARY`	Also provide a separate error analysis for each variable in addition to the entire solution vector.

`[Name of Exact Solution Function = `*`string`*`]`

This line gives the name of the function that will be called to evaluate the accuracy of the computed solution. The generic function *`user_bc_exact`*, in the file *`rf_user_bc_exact_fn.c`*, may be modified to compute the exact solution.

Auxiliary Output Specifications

The "`Auxiliary Output Specifications`" section is used to identify which, if any, auxiliary variables will be output. Auxiliary variables are quantities computed from the primary solution variables, and written as nodal variables to the named auxiliary output EXODUS II file. The named function that generates the auxiliary variable may either be one of a number of precoded functions, or may be written by the user and added to the file *`rf_misc_fn.c`*. If auxiliary output is requested, then the frequency of output (for transient problems) will be identical to that specified in the "`Output Specifications"` section. This section is optional.

Example:

```
------------------------------------------------------------
     Auxiliary Output Specifications
------------------------------------------------------------
Auxiliary output file           = theis_3d_auxout.exoII
Number of auxiliary nodal variables = 2
Auxiliary nodal variable and function name(s):
  AUX_OUT = Head    FUNCTION = Hyd_head
  AUX_OUT = P_liq   FUNCTION = P_liq
```

Description

`[Auxiliary output file = {`*`string`*`|`***`none`***`}]`

This is the name of an EXODUS II file to which the auxiliary variables will be written. The path is optional if the file is located in the same directory as the PorSalsa executable. The auxiliary output EXODUS II file must exist at the time of program execution. Although the file will not contain nodal values initially, the finite element geometry, and names of nodal variables must be present; see the ***Auxiliary Output File*** section on page 75 for further details on creating an auxiliary output file. If this input line is not found, if *`string`* is blank, or if *`none`* is specified, then auxiliary output information is not generated.

`[Number of auxiliary nodal variables = `*`integer`*`]`

This parameter specifies how many auxiliary variables to output. Default = ***`0`***.

`[Auxiliary nodal variable and function name(s):]`

```
AUX_OUT = string1(1)    FUNCTION = string2(1)
AUX_OUT = string1(2)    FUNCTION = string2(2)
   :    =    :               :    =    :
AUX_OUT = string1(N)    FUNCTION = string2(N)
```

Each line specifies the name of the nodal variable in the auxiliary output file (*`string1`*), as well as the function (*`string2`*) used to generate the auxiliary values. The number of auxiliary variables listed (*`N`*) must match the number specified by `Number of auxiliary` nodal variables. Each nodal variable name (*string1*) must match a nodal variable name defined in the auxiliary output EXODUS II file.

Available options for precoded auxiliary output functions (*string2*) include:

P_total	Total (Gas Phase) Pressure, P_{tot}
P_cap	Capillary Pressure, Pc
P_liq	Liquid Phase Pressure, $P_l = P_{tot} - Pc$
Sat_liq	Liquid Phase Saturation, S_l
Mass_frac_water_in_liq	Mass Fraction of Water in the Liquid Phase, Y_{wl}
Mass_frac_water_in_gas	Mass Fraction of Water in the Gas Phase, Y_{wg}
Hyd_head	Hydraulic Head, h
Rho_liq	Density of Liquid Phase, ρ_l
Rho_gas	Density of Gas Phase, ρ_g
Liq_DarcyFlux_x	Darcy Flux of Liquid, x-component, $v_{l,x}$
Liq_DarcyFlux_y	Darcy Flux of Liquid, y-component, $v_{l,y}$
Liq_DarcyFlux_z	Darcy Flux of Liquid, z-component, $v_{l,z}$
Gas_DarcyFlux_x	Darcy Flux of Gas, x-component, $v_{g,x}$
Gas_DarcyFlux_y	Darcy Flux of Gas, y-component, $v_{g,y}$
Gas_DarcyFlux_z	Darcy Flux of Gas, z-component, $v_{g,z}$
Heat_Flux_x	Flux of Heat, x-component, q_x
Heat_Flux_y	Flux of Heat, y-component, q_y
Heat_Flux_z	Flux of Heat, z-component, q_z

The function Hyd_head is based on the REF_PLANE specified in the "Material Input Specifications" section of the input file. The heat fluxes are components of $\mathbf{q}$ in Eqn. (5). Surface integrals of mass or heat can be requested in the section ***Miscellaneous Functions*** on page 64.

Function Data Specifications

The section "Data Specification for User's Functions" is used to supply parameter values to user's functions, to precoded models for unsaturated hydraulic properties, to precoded initial and boundary condition functions, and to functions used to compute exact analytical solutions to compare against the numerical solution.

Example:

```
------------------------------------------------------------
     Data Specification for User's Functions
------------------------------------------------------------
Number of functions to pass data to = 5

Function = VG_krel 2
FN_DATA = INT 0
FN_DATA = 3.34 1.982 0.277 0.0 1

Function = VG_cap_press 2
FN_DATA = INT 0
FN_DATA = 3.34 1.982 0.277 0.0 1

Function = VG_krel 2
FN_DATA = INT 1
FN_DATA = 3.78 1.022 0.128 0.0 1

Function = VG_cap_press 2
FN_DATA = INT 1
FN_DATA = 3.78 1.022 0.128 0.0 1

Function = f_head_init_cond 1
FN_DATA = 200.00  0. -1. 0. -100.  10.0
```

Description and Usage

`Number of functions to pass data to =` *`integer`*

This number specifies how many sets of parameters will be supplied as function data, not how many unique functions will be named. This distinction can be seen in the example given above where only three unique functions are specified, but five sets of function specifications and parameters are listed.

`[Function =` *`string integer`*`]`

These parameters specifies the function that the parameters will be supplied to (*`string`*), and how many lines of function data (*i.e.*, `FN_DATA` lines) follow (*`integer`*). The default value of *`integer`* is ***`0`***. Pre-defined functions and their required function data are discussed below.

`[FN_DATA = {`*`See Options Below`*`}]`

The actual parameter values are supplied line by line according to the type of data (integer, floating point, character string, or tabulated values). A "`#`" in the first column may be used to indicate a comment line and blank lines are allowed. Descriptions of the parameter specification formats for these types of data are as follows.

INT|INTEGER integer [integer integer ...]

Integer parameter data. At least one value is expected.

FLOAT float [float float ...]

Floating point parameter data. At least one value is expected.

STRING string

Character string parameter data. At least one character is expected.

TABLE integer1 integer2

```
  [ integer|float  [integer|float   ...  integer|float] ]
  [       .                 .        ...         .       ]
  [       .                 .        ...         .       ]
  [       .                 .        ...         .       ]
  [ integer|float  [integer|float   ...  integer|float] ]
```

Tabulated parameter data which may be either integer or floating point. `integer1` and `integer2` specify the number of rows and columns in the data table, respectively. A "#" in the first column may be used to indicate a comment line and blank lines are allowed.

Characteristic Curves

Pre-defined functions specified by `string` are listed below, along with their required function data specifications, if any.

user_krel_lg	User-supplied relative permeability function.
user_cap_press	User-supplied capillary pressure – saturation function.
VG_krel	van Genuchten [*van Genuchten*, 1978] relative permeability functions.

$$k_{rl} = \sqrt{s}\left[1-\left(1-s^{1/\lambda}\right)^{\lambda}\right]^2 ,$$

where

$\lambda = 1 - 1/\beta$, and

$$s = \left(S_l - S_{liq\_res}\right) \Big/ \left(\left(1 - S_{gas\_res}\right) - S_{liq\_res}\right)$$

VG_cap_press	van Genuchten capillary pressure – saturation function. $p_c = \frac{\rho_l g}{\alpha}\left(s^{-1/\lambda} - 1\right)^{1/\beta}$

The van Genuchten models require the following function data specifications:

```
FN_DATA = INT integer
FN_DATA = float1 float2 float3 float4 float5
```

integer	Material Number	
float1	van Genuchten model parameter, *ALPHA*, α	[m^{-1}]
float2	van Genuchten model parameter, *BETA*, β	[-]
float3	Residual liquid saturation, *SL_RES*, s_{liq_res}	[-]
float4	Residual gas saturation, *SG_RES*, s_{gas_res}	[-]
float5	van Genuchten model, option for computing relative permeability to the gas phase, *KREL_G_OPT*, k_{rg}	

=0.0, $k_{rg} = 0.0$,

=1.0, $k_{rg} = 1 - k_{rl}$,

=2.0, $k_{rg} = \sqrt{1-s}\left[1 - s^{1/\lambda}\right]^{2\lambda}$

BC_krel Brooks-Corey (Brooks and Corey, 1966) relative permeability functions.

$$k_{rl} = s^{(2+3\lambda)/\lambda}$$

$$k_{rg} = \left(1 - s^2\right)\left[1 - s^{1+2/\lambda}\right],$$

where

$$s = \left(S_l - S_{liq\_res}\right) \Big/ \left(\left(1 - S_{gas\_res}\right) - S_{liq\_res}\right)$$

`BC_cap_press`	Brooks-Corey [*Brooks and Corey*, 1966] capillary pressure – saturation function. $p_c = \rho_l g \varphi_{entry} s^{-1/\lambda}$

The Brooks-Corey models require the following function data specifications:

```
FN_DATA = INT integer
FN_DATA = float1 float2 float3 float4
```

integer	Material Number	
float1	Brooks-Corey air entry head, *P_ENTRY*, ϕ_{entry}	[m]
float2	Brooks-Corey model parameter, *LAMBDA*, λ	[-]
float3	Residual liquid saturation, *SL_RES*, s_{liq_res}	[-]
float4	Residual gas saturation, *SG_RES*, s_{gas_res}	[-]

`cubic_krel` Cubic relative permeability functions (Udell and Fitch, 1985).

$$k_{rl} = s^3$$

$$k_{rg} = (1-s)^3 \text{ ,}$$

where

$$s = \left(S_l - S_{liq\_res}\right) \Big/ \left(\left(1 - S_{gas\_res}\right) - S_{liq\_res}\right)$$

`cubic_cap_press` Cubic capillary pressure – saturation function.

$$p_c = \sigma \sqrt{\frac{\phi}{k}} \left(c_1 (1-s) + c_2 (1-s)^2 + c_3 (1-s)^3\right)$$

The cubic models require the following function data specifications:

```
FN_DATA = INT integer
FN_DATA = float1 float2 float3 float4 float5 float6
```

integer	Material Number	
float1	Residual liquid saturation, *SL_RES*, s_{liq_res}	[-]
float2	Residual gas saturation, *SG_RES*, s_{gas_res}	[-]
float3	Surface tension, *SIGMA*, σ	[N/m]

`float4`	Cubic model coefficient 1, *COEF1*, c_1	[-]
`float5`	Cubic model coefficient 2, *COEF2*, c_2	[-]
`float6`	Cubic model coefficient 3, *COEF3*, c_3	[-]

Miscellaneous Functions

Miscellaneous pre-defined functions specified by *`string`* are listed below, along with their required function data specifications, if any.

`f_auxout_head` Auxiliary output of hydraulic head

The head auxiliary output function does not require function data specifications. However, it does reference the computed hydraulic head to the reference plane, which is specified in the "`Material ID Specifications`" section through the parameter specifications of *REF_PLANE*.

`f_ss_flux` Surface heat/mass flux integrated over a side set

Specification of surface flux integrals requires two sets of function data as follows:

```
FN_DATA = INT [side_set1] [side_set2] ... [side_setN]
FN_DATA = STRING [Liq] [Gas] [Heat]
```

The user may specify any number of side sets and any subset of the three primary flux vectors (Liquid = `Liq`, Gas = `Gas`, Energy = `Heat`). For each flux specified, a file named "`ss_<type>_flux.dat`" (where `<type>` is either `Gas`, `Liq`, or `Heat`), is generated with one line of data per output time as follows:

Time1 Flux Over `side_set1` [*Flux Over* `side_set2`] ... [*Flux Over* `side_setN`]

Time2 Flux Over `side_set1` [*Flux Over* `side_set2`] ... [*Flux Over* `side_setN`]

⋮

TimeN Flux Over `side_set1` [*Flux Over* `side_set2`] ... [*Flux Over* `side_setN`]

Notes:

- The units of `Liq` and `Gas` volumetric fluxes is m^3/sec. The units of the `Heat` flux is Watts, and it is the integral of the total energy flux, **q**, as defined in Eqn. (5).
- The user does not need to indicate how many parameter values are being supplied, only the number of lines of function data.

Programming User Functions

User functions provide a flexible way of customizing PorSalsa for specific applications. The available user functions are listed in Table 6, along with the associated parameter list. Table 7 through Table 10 show definitions of additional generic parameter lists. The general procedure for utilizing user functions is to customize these functions for the particular application and then recompile the code.

Function	***Value(s) Returned***	***Parameter List***
Material Properties		
user_cond	Thermal Conductivity	T, Mole Fractions, S_l, x, y, z, Pointer to Material Property Structure
user_hcoeff	Convective heat transfer coefficient	T, $Tref$, t, Side Set ID
user_density	Solid phase density	T, Mole Fractions, Thermodynamic Pressure, x, y, z, Pointer to Material Property Structure, Hydrodynamic Pressure
user_Cp	Specific heat	T, Mole Fractions, x, y, z, Thermodynamic Pressure, Pointer to Material Property Structure
user_bindiff	Binary diffusion coefficient	T, Mole Fractions, x, y, z, $Ptot$, S_l, Pointer to Material Property Structure
user_cap_press	Capillary pressure	CAP_PRESS_FUNCTION_ARGLIST
user_krel_lg	Relative permeability to both the liquid and gas phases	REL_PERM_FUNCTION_ARGLIST
Boundary Conditions		
user_bc_exact	Boundary condition value	SNGLVAR_FUNCTION_ARGLIST
user_atmbc_liq_adv	Coefficient for the liquid-advection term of an atmospheric boundary	SNGLVAR_FUNCTION_ARGLIST
user_atmbc_gas_adv	Coefficient for the gas-advection term of an atmospheric boundary	SNGLVAR_FUNCTION_ARGLIST
user_atmbc_wg_diff	Coefficient for the diffusion of water-in-gas term of an atmospheric	SNGLVAR_FUNCTION_ARGLIST

	boundary	
user_atmbc_ag_diff	Coefficient for the diffusion of air-in-gas term of an atmospheric boundary	SNGLVAR_FUNCTION_ARGLIST
user_atmbc_hcond	Coefficient for the heat conduction term for an atmospheric boundary	SNGLVAR_FUNCTION_ARGLIST
Miscellaneous		
user_continuation	User-defined continuation function. Changes in the boundary conditions or material properties can be made as a function of the continuation parameter	Continuation Parameter
user_out	Output quantities of interest.	*t*, Pointer to Mesh Structure, Status Flag, Solution Vector, Time Step Number
user_init_cond	Initial Conditions	SNGLVAR_FUNCTION_ARGLIST
user_source	Source term, single value	SNGLVAR_FUNCTION_ARGLIST
user_source_multi	Source term, vector of values	MULTIVAR_FUNCTION_ARGLIST
user_velocity	Darcy velocity vector	MULTIVAR_FUNCTION_ARGLIST

Table 6: Available user functions with description, and argument list.

Argument	***Description***
double soln[]	Current solution vector
double soln_dot[]	Current time derivative of solution vector
double x, y, z	*x*-, *y*-, and *z*- coordinates
double t	Time
MATRL_PROP_STRUCT *matID_ptr	Pointer to the material property structure for the material being processed (defined in `rf_matrl_const.h`).
int var_num	Equation index (WATER_DENSITY, AIR_PRESSURE, TEMPERATURE), see `rf_fem_const.h`.
int sub_var_num	Species equation index (applicable only when var_num=MASS_FRACTION)
short int eqn_offset[]	Offset in soln[] for each variable, *e.g.*, the temperature is soln[eqn_offset[TEMPERATURE]]
int num_dim	Number of spatial dimensions
BC_STRUCT *bc	Pointer to the boundary condition structure (defined in `rf_bc_const.h`)

Table 7: Arguments passed via SNGLVAR_FUNCTION_ARGLIST.

Argument	***Description***
double src_vec[]	Source vector computed within user function
double soln[]	Current solution vector
double soln_dot[]	Current time derivative of solution vector
double x, y, z	*x*-, *y*-, and *z*- coordinates
double t	Current value of time
MATRL_PROP_STRUCT *matID_ptr	Pointer to the material property structure for the material being processed (defined in *`rf_matrl_const.h`*).
short int eqn_offset[]	Offset in soln[] for each variable, e.g., the temperature is soln[eqn_offset[TEMPERATURE]]
int num_dim	Number of spatial dimensions

Table 8: Arguments passed via MULTIVAR_FUNCTION_ARGLIST.

Argument	***Description***
double *krel_l	Pointer to the relative permeability to the liquid phase (value to be computed).
double *krel_g	Pointer to the relative permeability to the gas phase (value to be computed).
double T	Current temperature
double X_k[]	Mole Fractions
double Sl	Liquid phase saturation
double x, y, z	*x*-, *y*-, and *z*- coordinates
NODE_STRUCT *current_node	Pointer to the node structure.
MATRL_PROP_STRUCT *matID_ptr	Pointer to the material property structure for the material being processed (defined in *`rf_matrl_const.h`*).

Table 9: Arguments passed via REL_PERM_FUNCTION_ARGLIST.

Argument	***Description***
double Sl	Liquid phase saturation
int matnum	Material number
NODE_STRUCT *current_node	Pointer to the node structure.
MATRL_PROP_STRUCT *matID_ptr	Pointer to the material property structure for the material being processed (defined in *`rf_matrl_const.h`*).

Table 10: Arguments passed via CAP_PRESS_FUNCTION_ARGLIST.

Other Miscellaneous Files

PorSalsa uses a number of files which have been referred to previously. These files are described below and include:

• Finite Element Mesh File (required)	**mesh_file.exoII**
• Mesh Load Balance File	**load_balance_file.nemI**
• Solution Output File	**mainout_file.exoII**
• Auxiliary Input File	**auxin_file.exoII**
• Auxiliary Output File	**auxout_file.exoII**

The filenames listed to the right of the tasks are for discussion purposes only.

Finite Element Mesh File

PorSalsa requires a finite element mesh formatted according to the EXODUS II file format (Schoof and Yarberry, 1992). Simulations may be run in 2- and 3-dimensions, but only the Cartesian coordinate system is supported at present. Future developments may include the ability to solve problems in radial (2D) or cylindrical (3D) coordinates.

Two widely available SNL software packages available for generation of finite element meshes are FASTQ (Blacker, 1988) and CUBIT (Blacker et al., 1994). FASTQ can be used for the generation of 2D finite element meshes in the EXODUS I format. These may be converted to 3D meshes using the Sandia Engineering Analysis Code Access System (SEACAS) utility programs GEN3D or GENSHELL. The SEACAS file format translation utility, **ex1ex2v2,** may then be used to convert the file from EXODUS I to EXODUS II. CUBIT is an interactive program for generating 3D finite element meshes in the EXODUS II format.

There are also a variety of other utility programs for examining, repositioning, and merging finite element meshes. Additional utility programs also exist for converting between a variety of finite element mesh database formats. The SEACAS home page gives an overview of many of these utility programs (http://www.cfd.sandia.gov/ESChome.html).

Load Balance File

To execute PorSalsa in a parallel mode, domain decomposition is needed to partition the mesh among the various processors. This is accomplished using the utility program **nem_slice**, which uses the CHACO software package (Hendrickson and Leland, 1995) to generate a NEMESIS I load balance file (Hennigan et al., 1998). This file is specified by the `LB file = `*`string`* data line in the "`General Problem Specifications`" section of the main input file.

The command for generating the load balance file is

```
>nem_slice –a nemslice_input_file
```

where *nemslice_input_file* is the name of the input file used by the **nem_slice** utility, which is not to be confused with input files for either PorSalsa or **add_var**. It contains five lines of parameter specifications, and an example is given below:

Example: Input File for nem_slice:

```
INPUT EXODUSII FILE        = pnc_simple_mesh.exoII
GRAPH TYPE                 = NODAL
DECOMPOSITION METHOD       = SPECTRAL, KL, NUM_SECTS=1
SOLVER SPECIFICATIONS      = TOLERANCE=2.0e-3,USE_RQI_VMAX=200
MACHINE DESECRIPTION       = MESH=3x4
```

In this example `pnc_simple_mesh.exoII` is the name of the finite element mesh. The mesh is being decomposed into 12 subdomains as specified by `MESH=3x4`. These two pieces of information are combined to form the name of the resultant load balance file, “`pnc_simple_mesh-m12-bKL.nemI`”.

The man page for the **nem_slice** utility is listed in Appendix E. It gives further information regarding the other parameters of the **nem_slice** input file. Online help is also available via the -h option, *i.e.*

```
>nem_slice –h
```

Output FEM File

The primary solution variables designated for output (“`Output Specifications`” section of the input file) are written to the primary output file (“`General Problem Specifications`” section of the input file). This file is in the EXODUS II format, and must be created prior to executing PorSalsa. One way to create this file is with the utility function **add_var**. Usage is as follows:

```
>add_var –n integer –l list -o mainout_file.exoII filename_mesh.exoII
```

where,

integer	Number of primary output variables (*1*,*2*, or *3*)
list	List of primary solution variable names separated by commas, and with no white space. The allowable names are Wden, Pair, and Temp) These names correspond to the primary solution variables water density, air pressure, and temperature, respectively. They are case sensitive and cannot be modified.
mainout_file.exoII	EXODUS II file for primary variable output

filename_mesh.exoII	EXODUS II file containing the finite element mesh

For example, if the "`Output Specifications`" section is as follows:

```
------------------------------------------------------------
     Output Specifications
------------------------------------------------------------
User Defined Output                 = yes
Time Index to Output To             = 1
Nodal variable output times:
      every 1 step
Number of nodal output variables  = 3
Nodal variable names:
      Water_Density
      Air_Pressure
      Temperature
```

then

```
>add_var –n 3 –l Wden,Pair,Temp -o mainout_file.exoII filename_mesh.exoII
```

would be the appropriate command to create the primary output file. It should be noted that all primary variables must be written to the output file. Premature program termination will result from specifying only a subset of the primary variables; this allows the output file to be used as a restart file.

Auxiliary Input File

An auxiliary EXODUS II file may be used for importing node-based material properties. A description of the functionality of this file can be found in the ***Auxilliary Input Specifications*** section of this report. The auxiliary input file itself is generated by mapping nodal data onto a copy of the finite element mesh file. The utility program **add_var** can be used to create this file. The command for executing **add_var** is:

```
>add_var –n integer –g input-create -o auxin_file.exoII mesh_file.exoII
```

where,

integer	Number of nodal variables to be added
input-create	Input file for **add_var**
auxin_file.exoII	Auxiliary input file to be created
mesh_file.exoII	Finite element mesh file

The file input-create specifies the name(s) of the variables being written, and directs how the numerical values of the auxiliary variables are generated. The format mirrors that of the "Material ID Specifications" section of the input file, where properties are assigned on a material number basis. The numbering and naming of materials, as well as the numbering of the element blocks, must match those specified in the PorSalsa input file.

Example: input-create

```
Material ID Specifications

Number of Materials             = 3

POROUS_MEDIUM                   = 0   "Zone_1"
 PORO RANDOM  0.3 0.5
 PERM LOGNORMAL 9.38e-12  0.9
 ELEM_BLOCK_IDS  =  1
END Material ID Specifications

POROUS_MEDIUM                   = 1   "Zone_2"
  PORO CONSTANT 0.351
  PERM RANDOM 1.e-13  5.0e-11
  ELEM_BLOCK_IDS    =  2
END Material ID Specifications

POROUS_MEDIUM                   = 3   "Zone_3"
 PORO GEO_STAT 1 0
 PERM GEO_STAT 2 0
 ELEM_BLOCK_IDS    =  3
END Material ID Specifications
```

Description:

`Number of Materials =` *`integer`*

This parameter specifies the number of materials requiring auxiliary variables.

`POROUS_MEDIUM =` *`integer`* `"`*`string`*`"`

This line indicates the start of the parameter specifications for a particular material.

`integer`	Material Number (*The first material should be 0 (zero) rather than 1)*
`string`	Name of Material

`string1` `= {`*`See Options Below`*`}`

This line specifies the name of a nodal variable (*`string1`*) as well as options defining how the values are assigned.. The variable(s) named must match those specified in the "`Material ID Specifications`" section. For example, in the input specification:

```
POROSITY     = AUX_INP=PORO
```

the variable name "`PORO`" is used to identify porosity. This variable name must be listed as *`string1`* within ***input-create***.

The options defining how the nodal variables are assigned are as follows:

`constant` *`float`*	Assigns a constant value of *`float`* to the nodal variables.
`zero`	Same as "`constant 0.0`".
`one`	Same as "`constant 1.0`".
`random`	Assigns a random value.
`NORMAL` *`float1 float2`*	Values sampled from a Normal distribution mean = *`float1`* standard deviation = *`float2`*
`LOGNORMAL` *`float1 float2`*	Values sampled from a Lognormal distribution *ln* mean = *`float1`* standard deviation = *`float2`*
`GEO_STAT` *`integer1 integer2`*	Values are taken from a column of data listed in a file named block#.dat, where # is specified by *`integer1`*. The column number is indicated by *`integer2`*. An example of the file **block#.dat** is given here:

Example: block#.dat File

```
NX = 41
NY = 21
NZ = 21
XMIN = 0.0
YMIN = 0.0
ZMIN = 0.0
XMAX = 82.0
YMAX = 4.20e+01
ZMAX = 4.2e+01
2
Porosity
Permeability
DATA
0.32  1.0e-12
0.30  1.7e-13
0.24  2.6e-12
0.21  8.9e-12
  :      :
```

The values `NX`, `NY`, and `NZ` specify how many data values there are in the *x*- *y*-, and *z*-directions. `XMIN` and `XMAX` specify the minimum and maximum coordinate values along the x-coordinate axis. "`2`" specifies that there are two columns of data, and the following two lines specify the names of the nodal variables. The line "`DATA`" must be included, and this is followed by columnar data of length `NX` x `NY` x `NZ`.

`SNGLVAR_FUNCTION` *`string`*	Values are assigned according to user-defined function named *string*. This capability is not currently functional.

`ELEM_BLOCK_IDS =` *`integer1`* `[`*`integer2 integer3`* `...]`

This parameter specifies which of the blocks of elements from the EXODUS II database defining the mesh correspond to the material number. Element blocks are identified by an integer and typically start at a value of 1.

`END`

The word "`END`" terminates the parameter specifications for a given material number. Any characters following this word are ignored, but in this case "`END`" must be followed by white space.

Notes on the `GEO_STAT` option:

- Interpolation is used to map the auxiliary input values to the grid. The auxiliary input data, as specified in a block#.dat file, is assumed to be in order from "across" in increasing *x*-direction, then "back" into the *y*-direction, and finally "up" the *z*-direction. This means that the auxiliary data must be in a prismatic block. The spatial limits of the prismatic block are specified by the parameters `XMIN`, `XMAX`, `YMIN`, `YMAX`, `ZMIN`, and `ZMAX` in the header information of block#.dat, while the discretization of the prism is specified by `NX`, `NY`, and `NZ`.
- If the mesh contains a hole or other irregularities then the auxiliary data that overlaps the hole is simply ignored.

Auxiliary Output File

Auxiliary variables to be calculated and the auxiliary output file to which they are to be written, are defined in the ***Auxiliary Output Specifications*** section of this report. This additional EXODUS II file must be created prior to execution of PorSalsa in much the same way as the primary output file. The utility **add_var** may be used to create this file. The command for adding variables is:

>add_var –n *integer* –l *list* -o *filename_aux.exoII filename_mesh.exoII*

where,

integer	Number of auxiliary output variables
list	List of auxiliary output variable names separated by commas, and with no white space. These names correspond to those named in the "Auxiliary Output Specifications" section of the input file.

filename_aux.exoII	EXODUS II file for auxiliary output
filename_mesh.exoII	EXODUS II file containing finite element mesh

For example, if the "`Auxiliary Output Specifications`" section is as follows:

```
------------------------------------------------------------
     Auxiliary Output Specifications
------------------------------------------------------------
Auxiliary output file           = theis_3d_auxout.exoII
Number of auxiliary nodal variables = 2
Auxiliary nodal variable and function name(s):
  AUX_OUT = Head    FUNCTION = Hyd_head
  AUX_OUT = P_liq   FUNCTION = P_liq
```

then

```
>add_var –n 2 –l Head,P_liq -o auxout_file.exoII mesh_file.exoII
```

would be the appropriate command to create the auxiliary output file. Quantities available for output as auxiliary variables are given in the section titled ***Auxiliary Ouput Specifications***. The nodal variable names are case sensitive.

Post-Processing

EXODUS II finite element databases may be examined using the software visualization packages MUSTAFA (Glass, 1998) or BLOT (Gilkey and Glick, 1988). There are also several post-processing utilities that allow the databases to be manipulated or edited prior to visualization or plotting. These include ALGEBRA (Gilkey, 1988) and APREPO (Sjaardema, 1997). Further editing can be accomplished after converting the EXODUS II finite element database to ASCI format using the Sandia Engineering Analysis Code Access System (SEACAS) utility routine **exotxt**. (See http://endo.sandia.gov/SEACAS/Documentation/SEACAS.html for further details).

Example Problems

Example 1: Saturated Flow to a Well in a Confined Aquifer

Synopsis

This validation problem tests the water equation implementation. It consists of a simulation involving a fully-penetrating water extraction well in a confined, saturated, isothermal reservoir of infinite horizontal extent. The problem is transient and a comparison is made to the analytical solution of Theis (1935). The problem demonstrates several PorSalsa capabilities and features including saturated, transient water flow, the use of hydraulic head initial and boundary conditions, and the auxiliary output of hydraulic head and the Theis solution at the same grid points for direct comparison to the PorSalsa solution. This validation problem was proposed by Ross *et al*., (1982) and a variation of the problem served as a validation and verification problem for the computer code FEHMN (See "Test of Pressure Transient Analysis", [Dash *et al*., 1997, pp. 83-85]).

Discussion

The numerical solution is compared to the analytical solution for hydraulic head drawdown presented by Theis (1935). The analytical solution is for radial coordinates and represents the well as a line sink. A cylindrical coordinate system has yet to be implemented in PorSalsa, so 2- and 3-dimensional pie-shaped meshes were used with a finite diameter well.[1] The domains encompass a 30° wedge stretching from the well (r_w = 0.05 m) out to a distance of 5000 m. The reservoir thickness is 10 m and the well is screened throughout the entire thickness of the aquifer.

The boundary conditions model the well screen by a constant water flux, while the hydraulic head is specified (at 200m) on the outer boundary. The initial hydraulic head is specified at 200m.

The parameters for this problem are listed in Table 11. The parameters listed as Miscellaneous were used to convert the analytical solution problem specifications to PorSalsa specifications. The PorSalsa input file for the 2D mesh is included in Appendix F.

[1] Note: The FEHMN problem used a two-dimensional vertical domain with cylindrical (*i.e*., radial) coordinates.

Parameter	***Value***
Storage Coefficient, *S*	$1x10^{-3}$
Transmissivity, $T_{xx} = T_{yy}$	$1x10^{-3}$ m^2/sec
Pumping Rate, *Q*	$3x10^{-3}$ m^3/sec
Miscellaneous	
Water Density, ρ_w	995.4 kg/m^3
Water Viscosity, μ_w	$1.13x10^{-3}$ kg/m-s
Water Compressibility, κ_T	$4.4x10^{-10}$ Pa^{-1}
Aquifer Compressibility, C_r	$3.2x10^{-8}$ Pa^{-1}
Porosity, ϕ	0.30
Aquifer Thickness, *b*	10 m
Temperature, *T*	10° C
Well Radius, r_w	0.05 m

Table 11: Parameters for Example Problem 1.

Figure 1 shows the head profile along a radial transect after 10 days of pumping. The figure compares the analytical and numerical solutions. Two meshes were used in calculating the solution with PorSalsa, a 2D mesh and a 3D mesh. The Theis problem itself depends only on time and the radial distance from the well. The 2D mesh is over a 30^{o} "pie" slice, in a plane perpendicular to the well bore, extending from the 5 cm well bore radius out to 5 km. In order to model the problem in a manner similar to a real wellbore, the 3D mesh includes the 10 meters of confined aquifer. This also provides a test of the hydraulic head boundary and initial condition formulations; there is a hydrostatic variation in pressure along the direction of the wellbore in the 3D version. From Figure 1, PorSalsa is seen to slightly under-predict the drawdown throughout most of the domain on both meshes, nevertheless the comparison is good. The greatest error occurs near the well and diminishes with radial distance. The symbols on the numerical curves denote the actual grid data, hence it is clear that the mesh is somewhat coarse near the well in the 3D problem. Figure 2 shows the corresponding relative errors in hydraulic head defined as

$$\frac{\left(h_a - h_{num}\right)}{\Delta h_a}$$

where the subscript *num* refers to the numerical solution and *a* corresponds to the analytical solution. The denominator is the drawdown at the current time, about 5 meters at 10 days. After 10 days of pumping the relative error is less than 5% in the 2D and 3D solutions, except for the grid point at the well bore in the 3D solution, which has about 14% error. This large error is due to the coarse meshing at the well bore where the steepest gradient occur. The errors stem from two sources: 1) the analytical solution specifies a line source whereas PorSalsa requires a finite diameter well, and 2) element distortion (extremely long thin elements in the 3D grid) produce numerical errors.

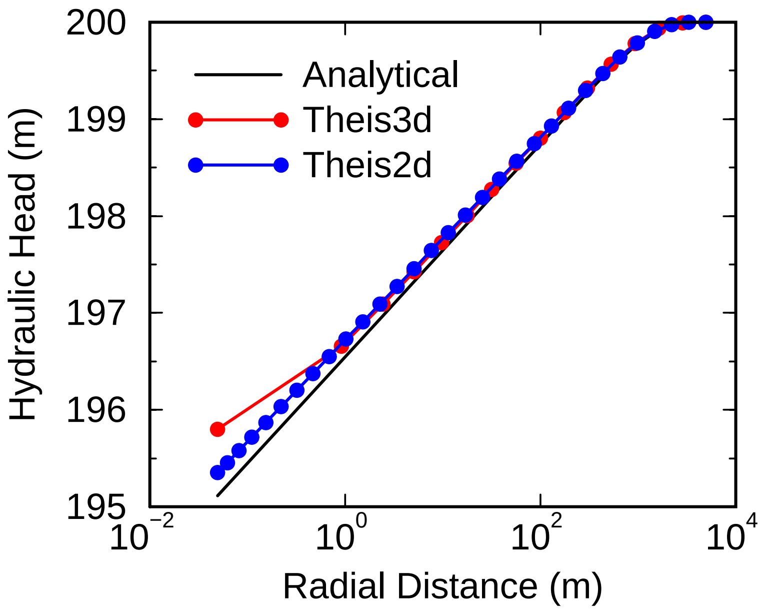


Figure 1: Head profiles along a radial transect after 10 days of pumping.

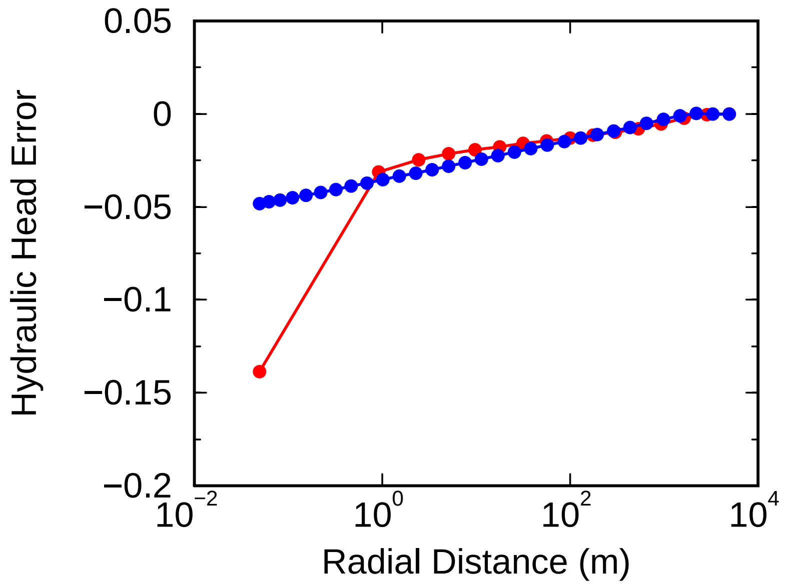


Figure 2: Hydraulic head error, (h-h_{num})/ Δh.

Example 2: Vapor Extraction Well

Synposis

This problem involves the flow of gas to an extraction well. The system is isothermal and the water phase is assumed to be immobile. This problem tests the implementation of the air balance equation, as well as implementation of anisotropic permeability. This example problem served as a validation and verification problem for the computer code FEHMN (See "Test of Vapor Extraction from an Unsaturated Reservoir", Dash *et al*., 1997, pp. 91-94). The example presented here differs from the FEHMN simulation in the spatial discretization and dimensionality of the problem.

Discussion

The numerical solution is compared to the analytical solution for air pressure drawdown as presented by Shan *et al*. (1992). The analytical solution is developed for a cylindrical coordinate system, which has yet to be implemented in PorSalsa. For this reason, a three-dimensional mesh was used for the simulation.[2] The domain encompasses a pie-shaped domain of 30° stretching from the well (r_w = 0.05 m) out to 115.68 m. The reservoir thickness is 10 m and the well is screened between the interval from 3 m to 7 m. The parameters for this problem are listed in Table 12, and include both an isotropic and anisotropic case. Along the well screen, an air mass flux is applied. The well extraction rates are 0.0825 kg/sec and 0.05 kg/sec in the isotropic and anisotropic problems, respectively. Mass fluxes for use in PorSalsa are obtained by dividing by the well screen area. The upper surface, which represents the ground surface, is modeled as a constant pressure boundary, with a gas pressure of 1.01325e+05 Pa. The remaining surfaces are no-flow boundaries. The input file for the anisotropic case is listed in Appendix F. The input file for the isotropic case would only differ in specifying the material as `POROUS_MEDIUM` rather than `ANISOTROPIC_POROUS_MEDIUM`, and in setting one value for the isotropic permeability. Figure 3 shows the mesh and isobars for isotropic permeability. This rather coarse mesh contains 780 hexahedral elements.

Case 1: Isotropic Permeability

For case 1 the permeability is isotropic ($k_{xx} = k_{yy} = k_{zz} = 1.e^{-11}$ m^2) and the input file for PorSalsa is similar to the one for anisotropic permeability shown in Appendix F. Figure 4a presents the analytical solution along a vertical plane extending from the well screen to a distance of 30 meters. The contours depict the gas-phase pressure relative to the initial value (P_{an}/P_{init}), and the region of drawdown can be seen to extend only about 10 meters from the well. Figure 5 gives the relative error for the numerical solution (defined as $(P_{num}-P_{an})/P_{an}$). The maximum relative

[2] Note: The FEHMN problem used a two-dimensional vertical domain with cylindrical (*i.e*., radial) coordinates.

error occurs adjacent to the borehole, but never exceeds 1.5%. Beyond 1 meter from the borehole the error drops below 0.5% and is negligible for distances greater than 10 meters.

Case 2: Anisotropic Permeability

For case 2 the permeability is anisotropic with the permeability in the vertical direction (along the *y*-coordinate in this case) decreased by one order of magnitude with respect to the horizontal ($k_{xx} = k_{zz} = 1.e^{-11}$ m^2, $k_{yy} = 1.e^{-12}$ m^2). The input file is listed in Appendix F. Figure 4b presents the analytical solution along a vertical plane extending from the well screen to a distance of 30 meters. The contours depict the gas-phase pressure relative to the initial value (P/P_{init}), and the region of drawdown now extends much further out with respect to the isotropic case. The maximum relative pressure reduction is about 0.82 near the well as compared with 0.90 for case 1 . Figure 6 gives the relative error for the numerical solution. The maximum relative error occurs adjacent to the borehole, as in the isotropic case, but now reaches almost 2.5%. Beyond 1 meter from the borehole the error still drops below 0.5% and is negligible for distances greater than 10 meter.

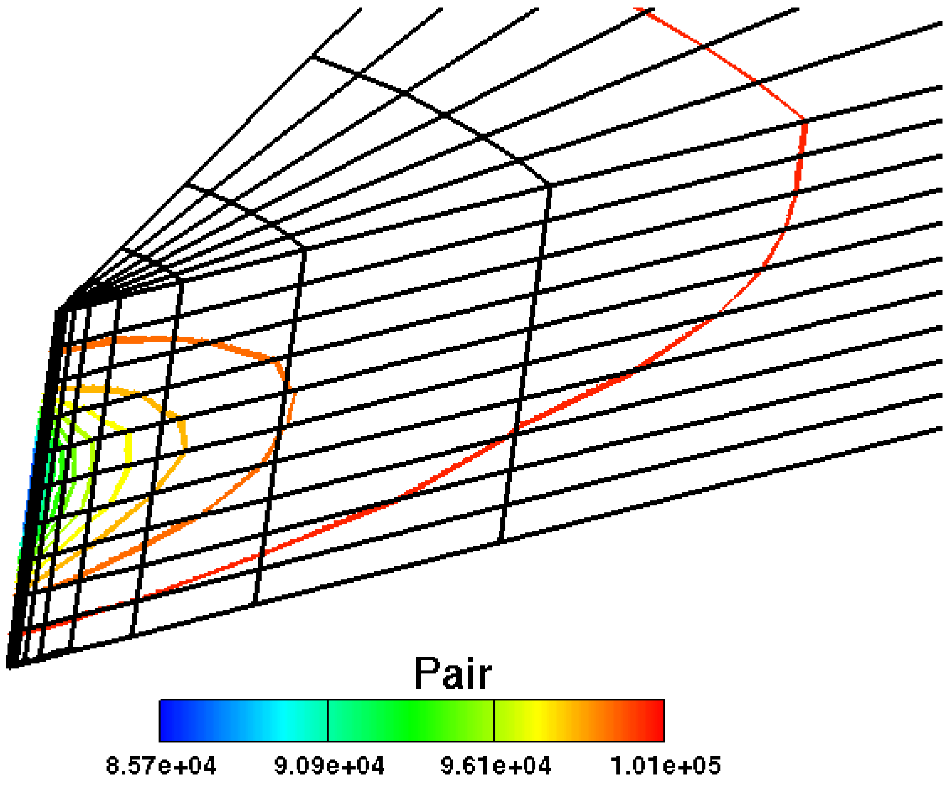
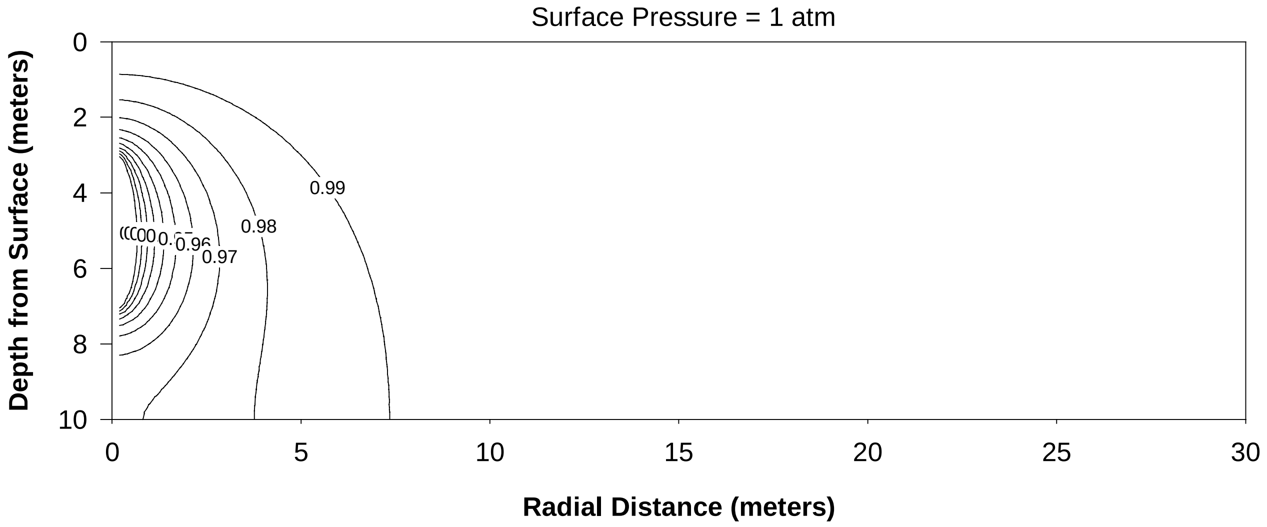


Figure 3: Mesh and air pressure isobars (Pascals) near the vapor extraction well.

Parameter	***Value***
Aquifer Compressibility, C_r	$1.0\text{x}10^{-8}$ Pa^{-1}
Porosity, ϕ	0.40
Aquifer Thickness, h	10 m
Top of Well Screen, a	3 m
Bottom of Well Screen, b	7 m
Temperature, T	10° C
Well Radius, r_w	0.05 m
Case 1: Isotropic	
Permeability, horizontal, k_{xx}, k_{zz}	$1\text{x}10^{-11}$ m^2
Permeability, vertical, k_{yy}	$1\text{x}10^{-11}$ m^2
Case 2: Anisotropic	
Permeability, horizontal, k_{xx}, k_{zz}	$1\text{x}10^{-11}$ m^2
Permeability, vertical, k_{yy}	$1\text{x}10^{-12}$ m^2

Table 12: Parameters for Example Problem 2.

a) Case 1: Isotropic Permeability ($k_r = 1x10^{-11}$, $k_z = 1x10^{-11}$)



b) Case 2: Anisotropic Permeability ($k_r = 1x10^{-11}$, $k_z = 1x10^{-12}$)

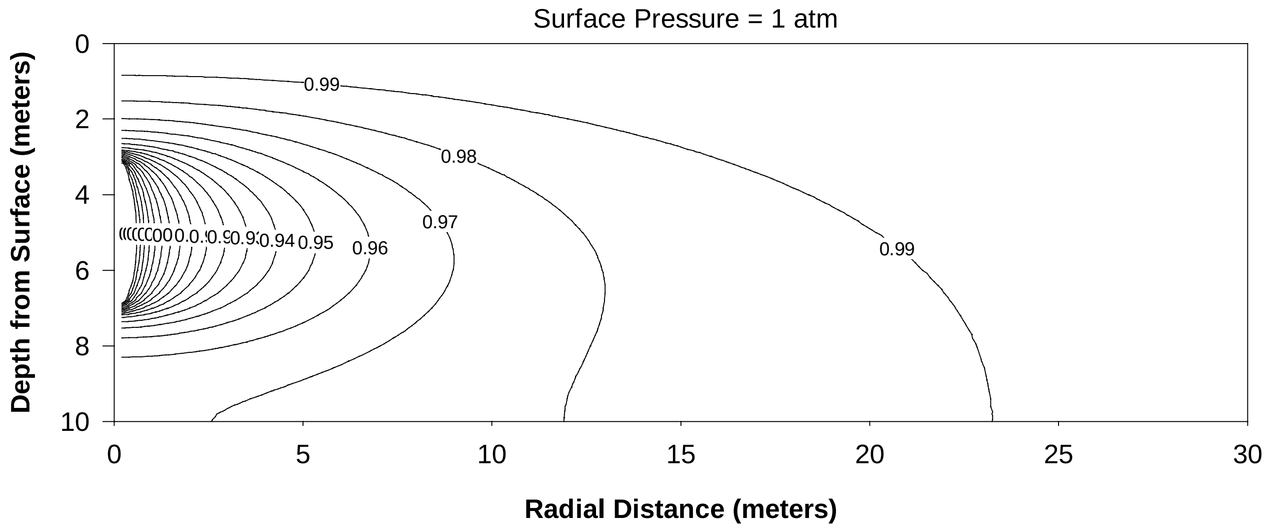


Figure 4: Analytical solution for relative pressure (P_{an}/P_{init}) for steady-state flow to a vapor extraction well for a) isotropic permeability, b) anisotropic permeability (*Shan et al.*, 1992).

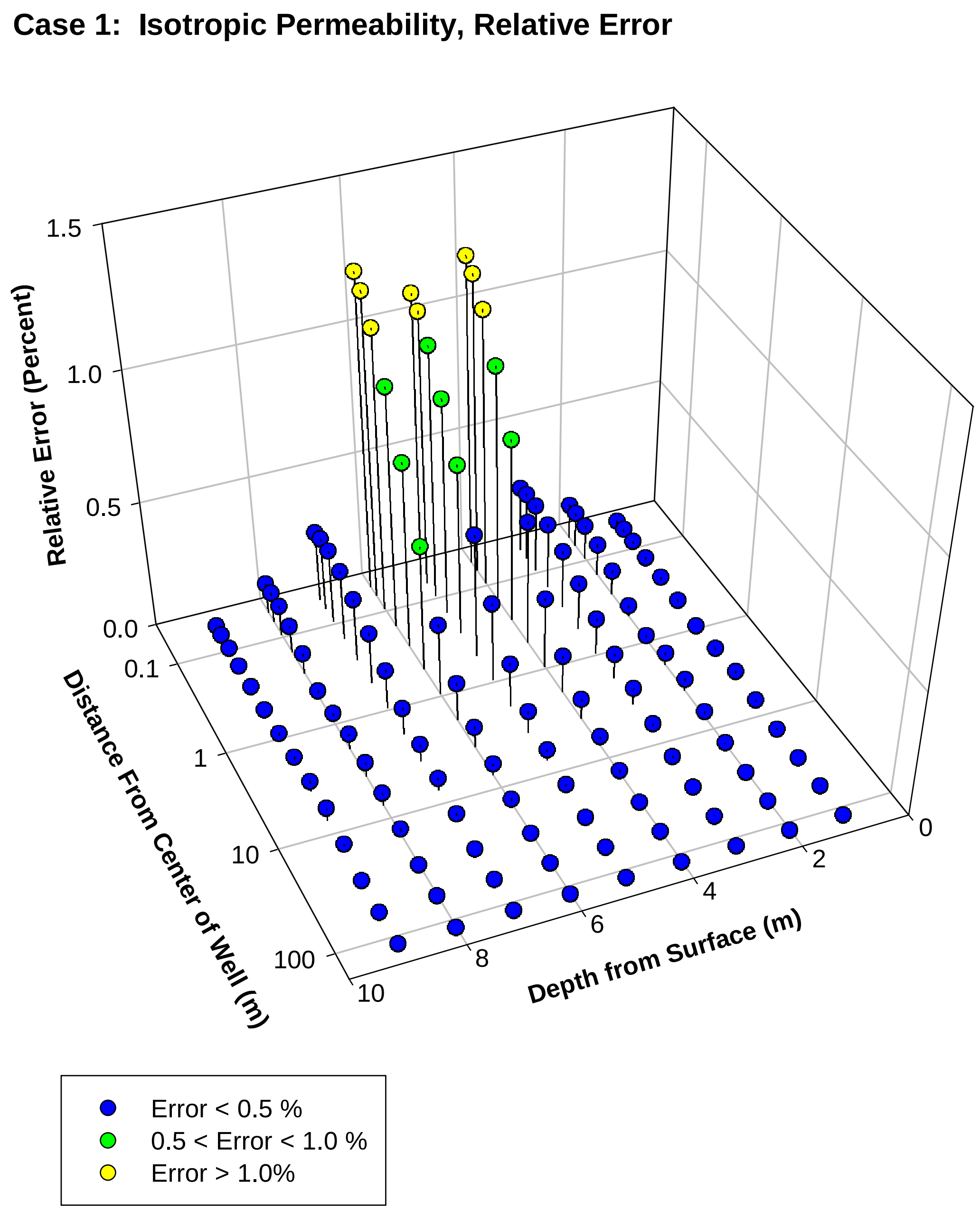


Figure 5: Relative error for the case of isotropic permeability. Note that the radial axis is plotted on a logarithmic scale.

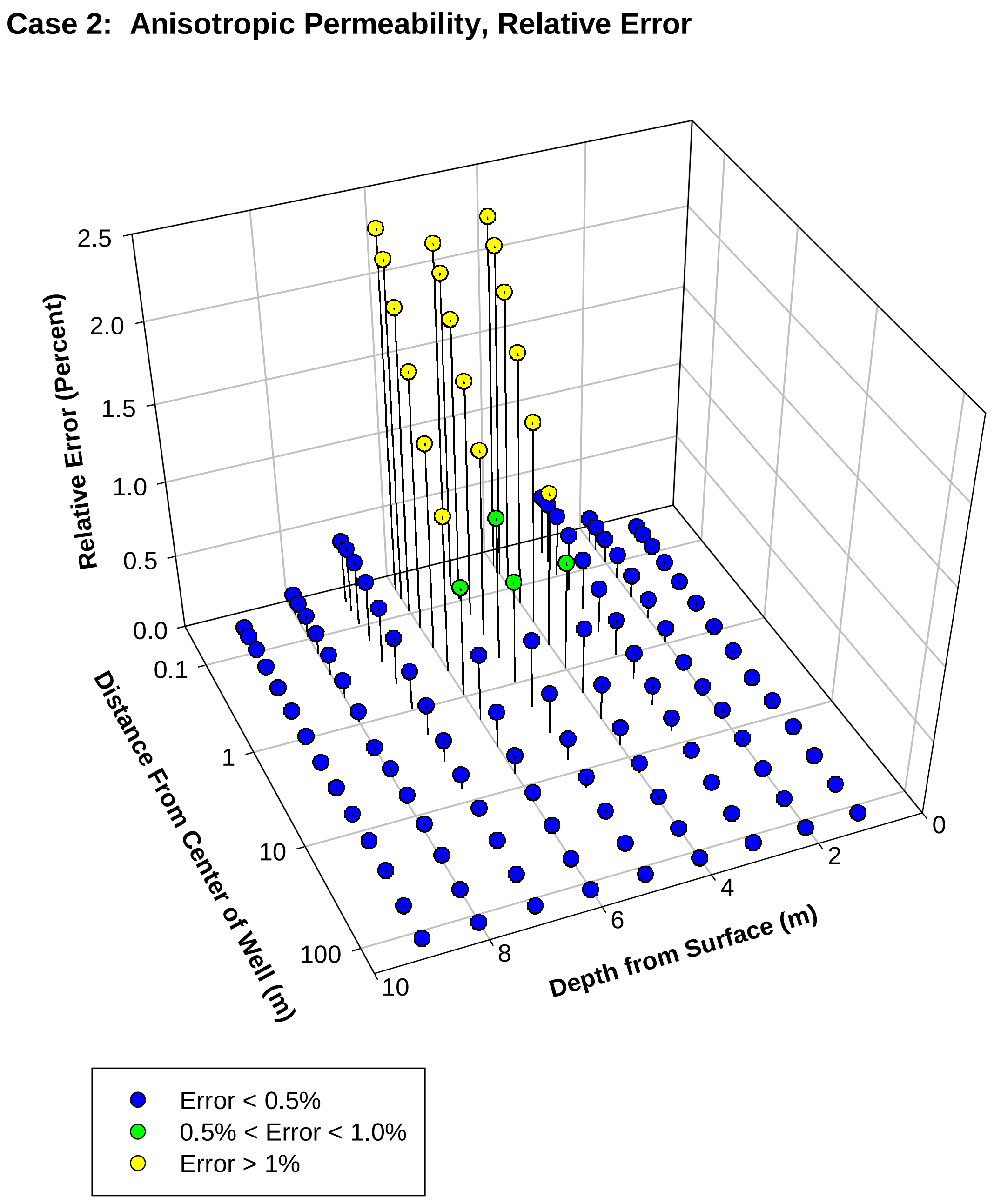


Figure 6: Relative error for the case of anisotropic permeability. Note that the radial axis is plotted on a logarithmic scale.

Example 3: Heat Induced Dryout in a Heatpipe

Synopsis

The steady heat pipe problem discussed by Udell and Fitch (1985) is the basis for this steady, thermal two-phase flow example problem. The problem involves the injection of heat into a one-dimensional horizontal column of porous material in which the void volume is filled with air and water (liquid and vapor). This problem has been used as a benchmark for the TOUGH code (Pruess, 1987), which compared well with the solution of Udell and Fitch, considering the simplifications made. The problem was also solved previously (Martinez, 1995) using a one-dimensional, method-of-lines code (MOL). This problem exercises features of evaporation/condensation and vapor and liquid flows in the code. Binary diffusion in the gas phase is included, and a saturation-dependent effective thermal conductivity is specified.

Discussion

The material properties specified for this problem are the same as in the steady heat pipe problems posed by Udell and Fitch (1985). The grid used was the same as in the TOUGH calculation; a 2.25 m column is discretized into 90, 2.5 cm finite elements. The material has 40% porosity and 1 Darcy permeability (=10^{-12} m^2). The capillary pressure-saturation relation and relative permeabilities are given by the cubic functions, `cubic_cap_press` and `cubic_krel` described in the ***Function Data Specifications*** section. In addition, the effective, saturation-dependent thermal conductivity was specified as

$$\lambda = \lambda_0 + \sqrt{\boldsymbol{S_l}}\left(\lambda_1 - \lambda_0\right),$$

with $\lambda_0 = 0.582$ W/m-K and $\lambda_1 = 1.13$ W/m-K. This functional form was specified by programming the `user_cond` function discussed in the **Programming User Functions** chapter on page 65. The problem definition input file is given in Appendix F.

The steady solution is determined by computing a transient solution, starting from an arbitrary initial condition, until a steady-state solution is obtained. The initial conditions are T=70 ^{o}C, P_g=1 atm, and S_l=0.5. To initiate the transport, the left end (x=0) is abruptly saturated with liquid, while the temperature and pressure are maintained at 70 C and 0.10133 MPa, respectively. At the same time, a 100 W/m^2 heat flux is applied at x=L, which is also closed to the flow of air and water.

Figure 7 compares the steady profiles of liquid saturation (S), air mole fraction (X) and temperature (T) as given by TOUGH2[3] and PorSalsa. The mole fractions are determined from the mass fractions ($Y_{\alpha\beta}$) according to

$$X_{ag} = \frac{Y_{ag}W_w}{Y_{ag}W_w + Y_{wg}W_a}$$

where W_a and W_w are the molecular weights of air and water, respectively. The PorSalsa solutions shown are at 80 days, at which the steady solution is obtained. It is noted that complete dry-out of the porous material occurs at about 2.1 m. The solutions agree well overall except for a small discrepancy in the calculation of the dryout point.

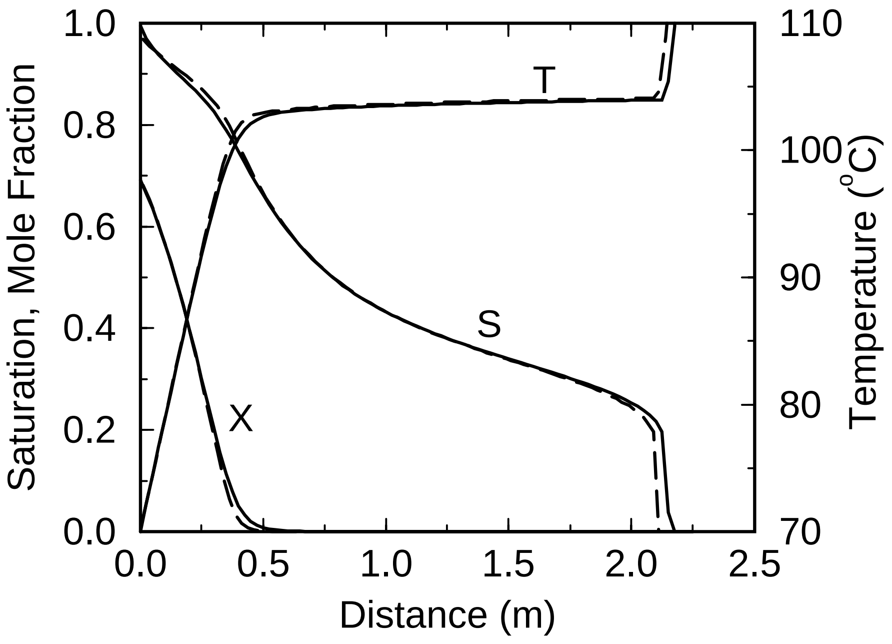


Figure 7: Comparison of steady state heatpipe solutions given by PorSalsa (solid line) and TOUGH2 (dashed line) for temperature (T), air mole fraction (X) and liquid saturation (S).

[3] We thank Teklu Hadgu of SNL for providing the TOUGH2 results presented here.

Example 4: Flow Past a Circular Inclusion

Synposis

This problem involves steady saturated flow past a cylindrical inclusion driven by a fixed pressure gradient. The cylindrical inclusion has permeability ten times greater than the surrounding material. A pressure drop of $2.1721x10^4$ Pa/m is applied over a 10 m length. There is an analytical solution for pressure and velocity, offering an opportunity to test the projection method used in PorSalsa to compute a continuous velocity field from the pressure solution. The input file for this problem is shown in Appendix F.

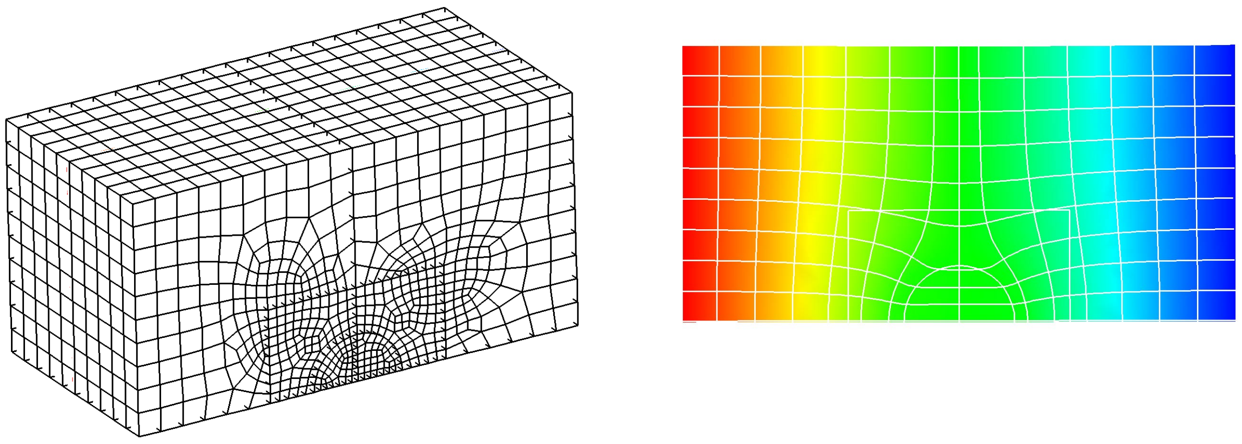


Figure 8: Mesh and pressure solution for flow past a cylindrical inclusion. The color indicates pressure level, with red being higher pressure.

Discussion

The mesh is shown in Figure 8, and has a 5x5 m^2 cross-section and is 10 meters long in the direction of the applied pressure gradient The problem is clearly two-dimensional; the 3D unstructured grid was used strictly for testing the algorithms. The grid consists of 5380 8-node hexahedral finite elements, for a total of 6347 node points. This figure also shows the pressure distribution and streamlines. The higher permeability inclusion focuses the streamlines. The pressure field shown was obtained with the GFEM scheme and is seen to display fore-aft symmetry, even on this irregular unstructured grid.

As it turns out, this problem is not ideal for testing the flux routines and surface flux integration capabilities, as it was intended. The reason is that the velocities are discontinuous across the interface with the inclusion. The analytical solution shows a uniform velocity field in the inclusion, i.e. V_x=constant, V_y=0. The numerical solutions show a similar behavior, although they can only approximate the discontinuity. Figure 9 shows velocities obtained with a mass-lumped projection scheme; the legend refers to the method used to solve for the pressures, which were obtained with the GFEM scheme and the CVFEM/Lobatto methods. Specifically, the pressure solution obtained with the CVFEM/Lobatto method (on this unstructured grid) was processed with the mass-lumped projection method to produce the velocity. The figure on the left gives the velocities along a horizontal line (in the direction of the pressure gradient) passing through the center of the inclusion. The figure on the right is along a vertical line through the center of the inclusion.

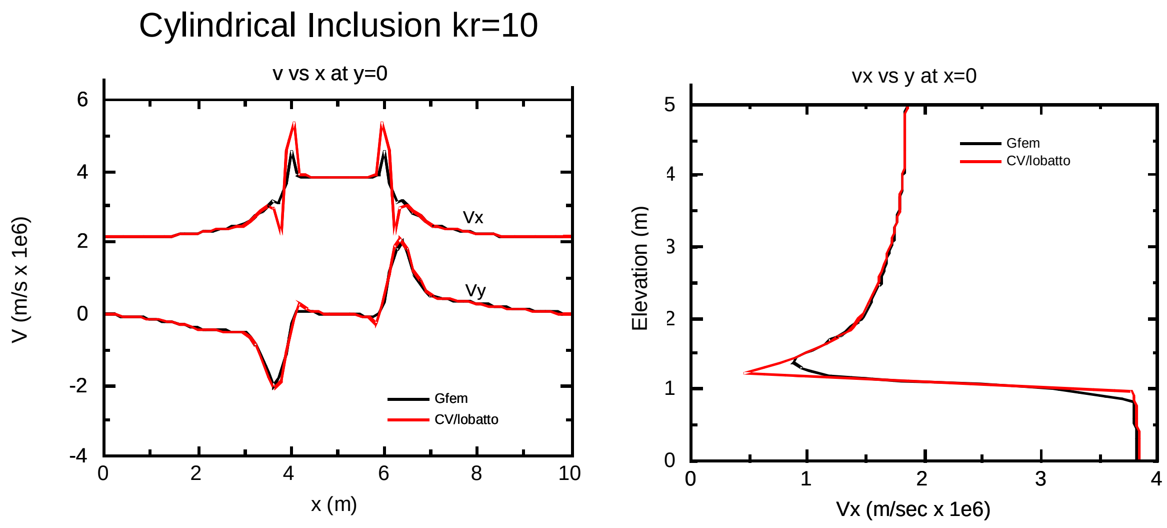


Figure 9: Darcy velocity profiles. Notice that the velocities are discontinuous across the interface with the inclusion .

Away from the interface, both schemes produce similar velocity fields. The velocity field computed from CVFEM/Lobatto pressures displays oscillations of higher amplitude when compared to the GFEM velocities. The velocities plotted on the line y=0 indicate approximations to a Heaviside jump for V_x and a dipole for V_y.

The results of surface flux integrations are shown in **Table 13**. In this table, side sets 1 and 2 are the high and low pressure faces, respectively. The algebraic sign indicates the orientation of the flux integral with respect to the outward pointing normal on the surface. Side sets 3, 4 and 5 form the part of the vertical plane bisecting the cylinder contained in the mesh. Side set 5 lies in the cylindrical inclusion, but due to our point-wise assignment of property data, is a half-element height short of the full cylinder. Side set 6 forms the upstream-facing cylinder interface and joins side set 5 near the top of the cylinder. The total volume flux through the inflow and outflow boundaries match very well (side sets 1 and 2). Further, the sums of fluxes on side sets 3, 4, and 5 also match the inflow and outflow boundaries well; the values for the GFEM, CVFEM/Lobatto and CVFEM/Gauss are, respectively, 5.14339, 5.13697, and 5.14339. The worst comparison is for the fluxes over the upstream side of the cylinder. The surface flux on side set 6 should match with those on side set 5, since they form a closed circuit. In particular, the values obtained from using the CVFEM/Lobatto scheme for the pressure are significantly in error. Referring to Figure 9, this must be due to the discontinuous nature of the velocity field on the interface of the inclusion. It should be noted that, on a rectangular grid, the CVFEM/Lobatto scheme reverts to a 7-point difference stencil. On the present grid, the sparseness in connectivity is also imparted by this scheme, but appears to result in a less accurate surface flux integral. The analytical value for the net flux through the mesh is 4.964x10^{-5} m^3/sec (taking a cylinder of unit radius, i.e., not accounting for an "effective" radius); the errors in the numerical values are about 3.6%.

Pressure Scheme	*Surface Flux(x10^5) on Side sets*					
	1	2	3	4	5	6
GFEM	-5.1443	5.1440	2.6073	0.6616	1.8745	-2.1742
CVFEM /Lobatto	-5.1443	5.1435	2.5968	0.6327	1.9075	-2.9001
CVFEM /Lobatto	-5.1444	5.1440	2.6073	0.6616	1.8745	-2.1742

Table 13: Surface Fluxes on Side Sets; the first row of integers denote the side set numbers referred in the text.

Example 5: Advective-Dispersive Transport

Synopsis

This sample problem involves the transport of an instantaneous injection of a contaminant at a point in a two-dimensional (2D) confined aquifer with a constant horizontal flow field. The problem serves as a verification of the transport equations in PorSalsa, including the hydrodynamic dispersion implementation.

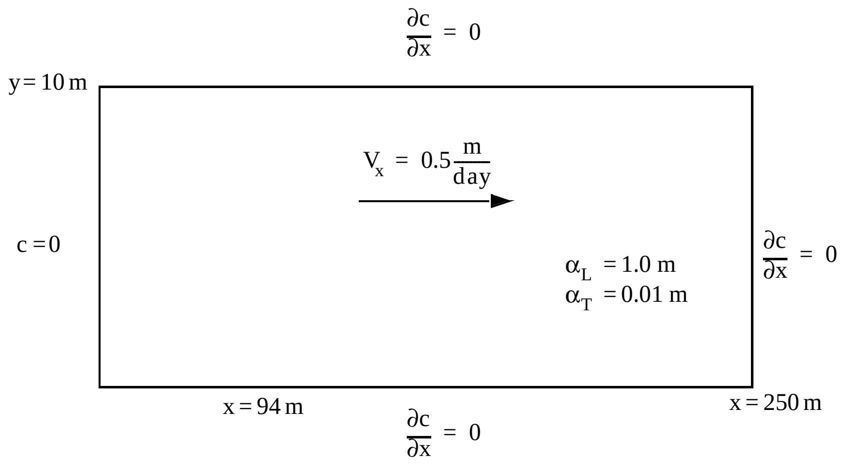
Discussion

In terms of the concentration form of the species transport equation (where we use the concentration, *C*, in place of $\rho \boldsymbol{Y}$),

$$\phi \frac{\partial C}{\partial t} + U \frac{\partial C}{\partial x} = D_{xx} \frac{\partial^2 C}{\partial x^2} + D_{yy} \frac{\partial^2 C}{\partial y^2} \qquad \begin{aligned} D_{xx} &= \alpha_L U \\ D_{yy} &= \alpha_T U \end{aligned}$$

the analytical solution for a Dirac delta point source of strength *M*, is (Bear, 1979, Eqn. 7-152, with a sign correction),

$$C = \frac{M}{4\pi t \sqrt{D_{xx} D_{yy}}} \exp\left(-\frac{\phi}{4t} \left(\frac{(x - Ut/\phi)^2}{D_{xx}} + \frac{y^2}{D_{yy}} \right) \right)$$



$\frac{\partial c}{\partial x} = 0$
y = 10 m
$V_x = 0.5 \frac{m}{day}$
c = 0
$\frac{\partial c}{\partial x} = 0$
$\alpha_L = 1.0$ m
$\alpha_T = 0.01$ m
x = 94 m
x = 250 m
$\frac{\partial c}{\partial x} = 0$

Figure 10: Two-dimensional domain for transport problem.

The rectangular computational domain for the numerical simulation measures 250 m x 10 m with the point source 94 m inside the left boundary (Figure 10). A uniform horizontal velocity of 0.5 m/day is defined. Dirichlet conditions of *C*=0 are defined on the left vertical boundary; all other boundaries are Neumann with zero concentration gradient. Besides those shown in Figure 10, other parameter values used in the evaluation of the analytical solution include M=1/6, $\phi=1$, $D_{xx}=0.5$ m^2/day, and $D_{yy}=0.005$ m^2/day. Note the significant anisotropy in the dispersivities. The finite element mesh is composed of 6426 nodes with a spacing of 2.0 and 0.2 m in the *x* and *y* directions, respectively.

Because of the error introduced in representing an instantaneous source, initial conditions for the transport calculations were defined by the analytical solution at a time of *t*=20 days. Subsequent transport was then computed and compared to the analytical solution along the *x*-axis (*y*=0). A comparison between analytical and numerical solutions at times of *t*=20 (initial conditions), 50, and 80 days is shown in Figure 11. Note that the *x*-coordinate of the computed solution has been shifted so that the source occurs at *x*=0, as is assumed by the analytical solution.

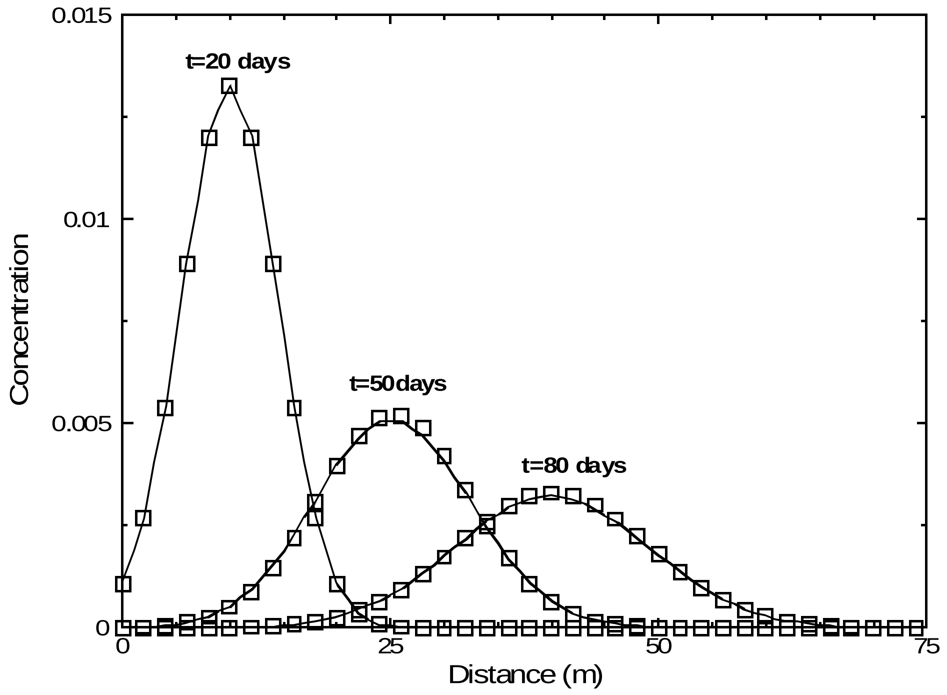


Figure 11: Comparison of analytical and numerical solutions for 2D anisotropic dispersive transport.

As mentioned earlier, the analytical solution was used to generate an initial condition for the transport calculations via the user-supplied function, *`user_init_cond`*. The computer code also allows specification of an exact solution function, *`f_pt_src`* in this application, to which the computed results are compared and various error measures reported. These error measures report a maximum difference between the analytical and computed solution of 3.3% and 2.8% at t=50 and 80 days, respectively. It should be noted that these errors are a function of the user-specified solution error tolerances in the calculation and could be reduced by defining a smaller tolerance.

For completeness and illustrative purposes, the input file used in the example problem is provided in Appendix F. Note that the combination of specifying *mass_conv_diff* as the *`Problem type`* under the "General Problem Specifications" section and *POROUS_MEDIUM* as the material type under the "Material ID Specifications" is what invokes the current implementation. Also, note that the material moisture content (*MOIST_CONT*), and transverse and lateral dispersivities (*DISP_TRAN* and *DISP_LONG*) must be defined in the "Material ID Specifications" section.

Example 6: Flow and Transport Through Fractured Rock

Synopsis

As a final example we consider flow and transport in a large scale 3D heterogeneous and fractured domain. This problem involves saturated, isothermal, steady-state groundwater flow through a fractured aquifer in which a backfilled drift is embedded. The drift represents a waste repository. The steady flow field through this domain will be calculated directly, (i.e., not via a false transient) and the resulting flow field will be used in a subsequent simulation of the advective-dispersive transport of a trace contaminant through the same region. The flow solution and heterogeneous data simulation represent one realization out of 50 which were performed in a modeling study reported by Reeves *et al.* (1999). In that work, the contaminant transport was modeled by a particle tracking method. Here we compare this method to an advective-dispersive model. This problem demonstrates the use of auxiliary input to define pointwise heterogeneous data generated external to PorSalsa, and the use of parallel processing.

Discussion

Flow Simulations

The so-called H-12 reference domain considered in Reeves *et al.* (1999) contains intensely fractured granodiorite and the flow conditions are steady-state, fully-saturated and isothermal. The computational domain is a cubical region measuring 200m on a side. In order to justify a continuum representation of the porous system, a very fine discretization is required. The finite-element mesh for these simulations used 2 x 2 x 2 meter regular hexahedral elements with eight nodes each. The final mesh contained 1,020,000 uniform hexahedral elements and 1,050,804 nodes. More details of the spatial discretization are given in Reeves *et al*. (1999). The heterogeneity of material properties is handled by using an auxiliary input file containing an array of values of intrinsic permeability, porosity, and fracture ratio (this third quantity is not used in the problem). The material properties from the auxiliary input file are mapped onto the mesh nodes via simple spatial interpolation.

In the study reported in Reeves *et al*. (1999), the flow and transport realizations were performed sequentially by first solving for the flow field with PorSalsa and then running a particle-tracking algorithm on the computed flux fields. The flow solution was computed on 20 of 72 440-MHz alpha processors on a DEC8400 parallel machine. Time per realization for the flow solution varied between 15 and 25 minutes due to the shared memory constraints of the DEC8400. This rapid turnaround time allowed the entire suite of 50 simulations to be run overnight.

Figure 12 shows the heterogeneous, spatially correlated permeability field that was read into PorSalsa via an auxiliary input file. The field was generated by Sean A. McKenna (SNL, Department 6115) using a "fractured continuum model" geostatistical simulation, which is useful for representing highly fractured domains where fracture statistics are available. It clearly shows a series of high-permeability fracture zones, embedded in substantially less permeable granite. The contrast in permeability between the "conductive" fracture-like features and the background "matrix" is about 8 orders of magnitude, and several large-scale conductive features are evident. The steady flow is driven by a 0.008 m/m head gradient oriented perpendicular to the repository (see Figure 12). Figure 12 also shows particle traces against a background of the hydraulic head distribution. This figure clearly shows a high degree of mechanical dispersion in the streamline patterns. Particle traces emanating from the high-head plane depict a highly tortuous path across the domain, resulting in large spatial variation with respect to their starting location. They give an indication of the variations in flow speed and mechanical dispersion along different paths. The particle paths are correlated with the high permeability streaks seen in the permeability field. See Reeves *et al.* for more discussion of the flow solution and various statistical properties of the flow.

Transport Simulations

The steady flow fields discussed above were used as input to advective-dispersive contaminant transport simulations performed on the same grid. Contaminant is introduced at the repository by specifying a unit value of concentration on the periphery of the backfill about the repository. The simulation is integrated in time using the 2nd order trapezoid rule. Isotropic dispersivities were specified as 0.2 m, such that the mesh Peclet (Δ/α_L, where Δ=2m is the mesh size) number (*Pe*) is about 10.

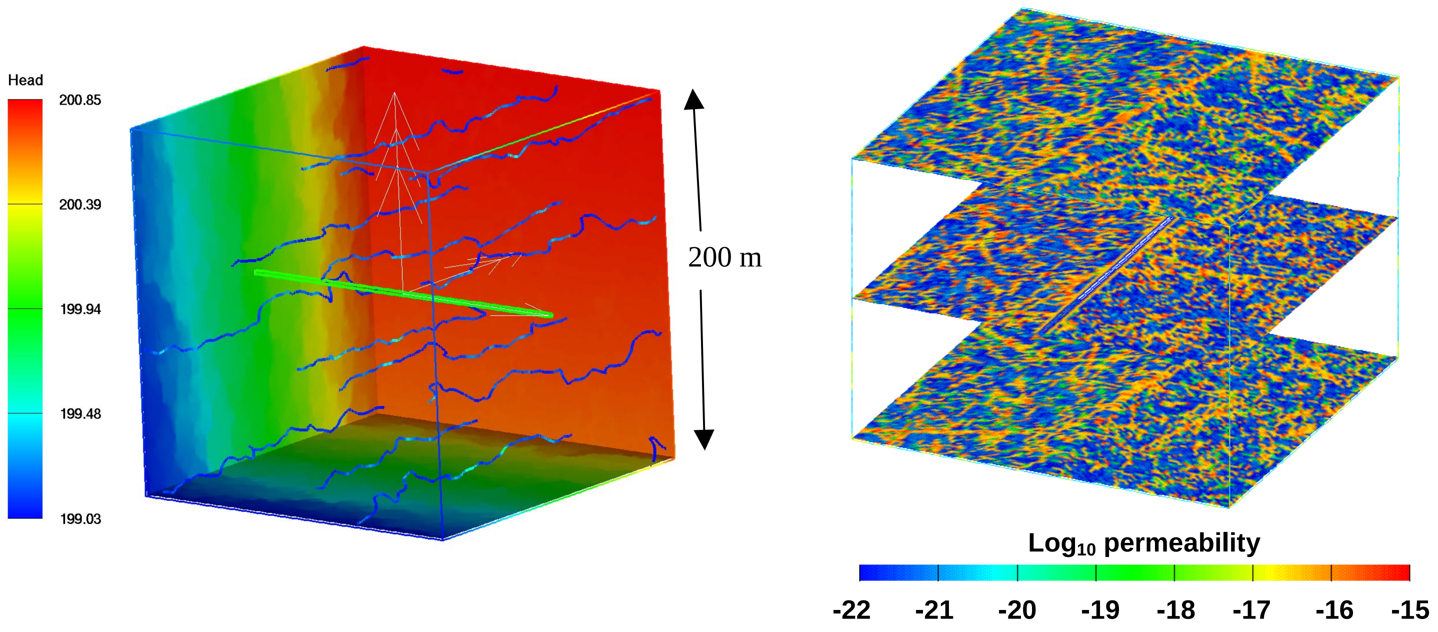


Figure 12: High resolution flow simulation in fractured granite.

Figure 13 shows an isosurface of the 1% relative concentration level at 100 years. The isosurface is highly irregular reflecting the tortuous transporting velocity field. The figure indicates the 1% concentration level has reached the downstream boundary at 100 yrs. Figure 14 is a view from above the repository of the contaminant concentration plume in a horizontal plane passing through the contaminant storage repository sited in the fractured granite. A high degree of flow channeling through the fractured granite is evident in the resulting concentration field. First arrival of contaminant to the outer boundary occurs at about 100 yrs. This is near the average breakthrough time computed via particle tracking (for pure advection) as reported in Reeves *et al.*, (1999). By 300 yrs, a significant amount of contamination has reached the outer boundary.

Dispersion errors upstream to the repository are evident in Figure 14, reflecting the large value of mesh Pe number used in the simulation. Dispersion occurs roughly for $Pe > 2$. In this simulation, we wanted to get a solution similar to pure advection to compare with the particle tracking simulations reported in Reeves *et al.*, (1999). MPSalsa includes a stream-line upwind Petrov-Galerkin (SUPG) formulation (Hughes *et al.*, 1986; Tezduyar *et al.*, 1992), which can be readily implemented in PorSalsa. The SUPG is a method of dissipating the wiggles without introducing cross-stream diffusion, as in a simple upwind scheme. The SUPG adds artificial dissipation along the local streamwise direction only.

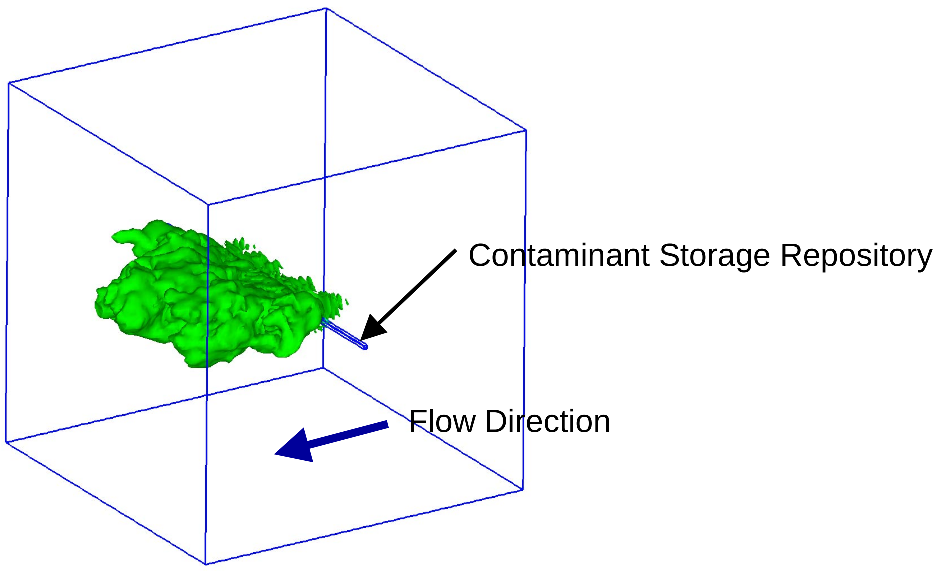


Figure 13: Isosurface of 1% concentration at 100 years.

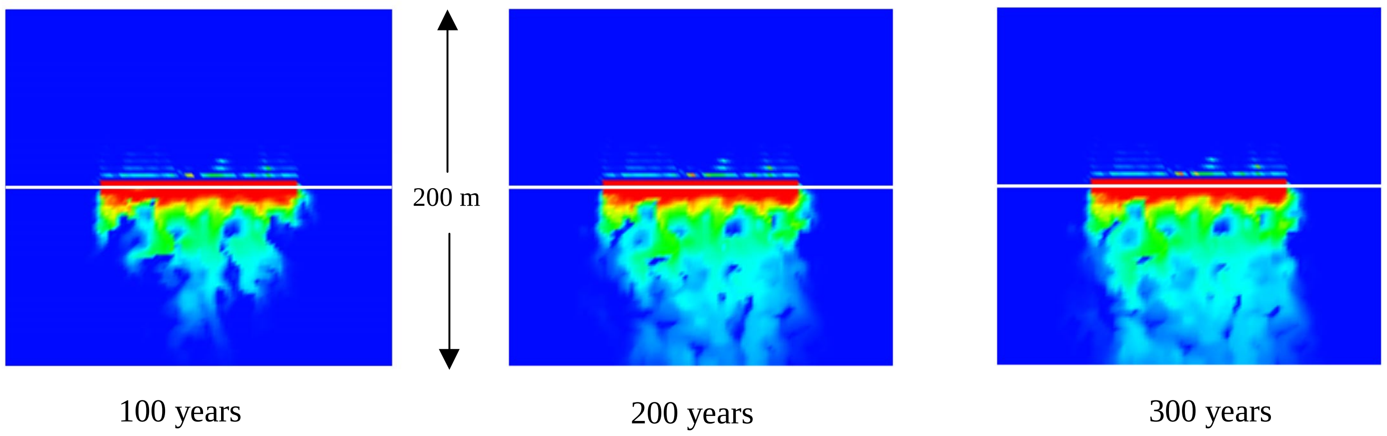


Figure 14: Contaminant concentration plume in a plane passing through the contaminant repository sited in fractured granite.

References

Bear, J., 1972, *Dynamics of Fluids in Porous Media*, American Elsevier, New York, p. 764.

Bear, J. 1979, *Hydraulics of Groundwater*, McGraw-Hill, New York, p. 569.

Blacker, T.D., *FASTQ Users Manual Version 1.2*, SAND88-1326, Sandia National Laboratories, Albuquerque, New Mexico, 119 pp..

Blacker, T.D., W.J. Bohnhoff, T.L. Edwards, J.R. Hipp, R.R. Lober, S.A. Mitchell, G.D. Sjaardema, T.J. Tautges, and T.J. Wilson, 1994, *CUBIT Mesh Generation Environment; Volume 1: Users Manual*, SAND94-1100, Sandia National Laboratories, Albuquerque, New Mexico, 226 pp.

Brooks, R. H., and A. T. Corey, 1966, Properties of porous media affecting fluid flow, *Proc. Amer. Soc. Civil Eng.*, No IR2, *92*, 61-87.

Dash, Z.V., B.A. Robinson, and G.A. Zyvoloski, 1997, *Software Requirements, Design, and Verification and Validation for the FEHM Application – A Finite-Element Heat- and Mass-Transfer Code*, LA-13305-MS, Los Alamos National Laboratory, Los Alamos, New Mexico, 206 pp..

Deutsch, C.V. and A.G. Journel. 1998, GSLIB: Geostatistical Software Library and User's Guide, 2nd ed., Oxford University Press, New York, 369 pp.

Freeze, R. A., and Cherry, J. A., 1979, *Groundwater*, Prentice Hall, Englewood Cliffs, New Jersey, 604 pp.

Eisenstat, S. C., and H. F. Walker, 1996, Choosing the forcing terms in an inexact Newton method, *SIAM J. Sci. Comput*., **17**, 16-32.

Gilkey, A.P., and J. Glick, 1988, *BLOT - A Mesh and Curve Plot Program for the Output of a Finite Element Analysis*, SAND88-1432, Sandia National Laboratories, Albuquerque, New Mexico, 82 pp..

Glass, M.W., 1998, *MUSTAFA User's Guide, Version 1.0*, SAND Report In Review, Sandia National Laboratories, Albuquerque, New Mexico, 155 pp.

Gresho, P. M., R. L., Lee, and R. L. Sani, 1980, On the time-dependent solution of the incompressible Navier-Stokes equations in two and three-dimensions, *Recent Advances in Numerical Methods in Fluids*, Volume **1**, Pineridge Press Ltd., Swansea, U. K., 27-81.

Gropp, W., E. Lusk, and A. Skjellum, 1995, *Using MPI*, MIT Press, Cambridge.

Hendrickson, B.A., and R.W. Leland, 1995, *The Chaco User's Guide Version 2.0*, SAND95-2344, Sandia National Laboratories, Albuquerque, New Mexico, 44 pp..

Hennigan, G.L., M. St. John, and J.N. Shadid, 1998, *NEMESIS I: A Set of Functions for Describing Unstructured Finite-Element Data on Parallel Computers*, Sandia National Laboratories unpublished report, Albuquerque, New Mexico, 75 pp

Hughes, J. R., L. P. Franca, and M. Balestra, 1986, A New Finite Element Formulation for Computational Fluid Dynamics: V. Circumventing the Babuska-Brezzi Condition: A Stable Petrov-Galerkin Formulation of the Stokes Problem Accommodating Equal-order Interpolations, *Comp. Meth. App. Mech. and Eng.*, **59**, 85-99.

Martinez, M.J., P.L. Hopkins, and J.N. Shadid, 1997, *LDRD Final Report: Physical Simulation of Nonisothermal Multiphase Multicomponent Flow in Porous Media*, SAND97-1766, Sandia National Laboratories, Albuquerque, New Mexico, 65 pp.

Pruess, K., 1987, TOUGH User's Guide, *LBL-20700* (NUREG/CR-4645), Lawrence Berkeley Laboratory, Berkeley, CA, 78pp.

Reeves, P. C., S. A. McKenna, E. K. Webb, A. H. Treadway, M. J. Martinez and P. L. Hopkins, Intercomparison of Flow Calculation for H-12 Reference Fractured Rock: Progress Report on Stage 1 Calculations, Sandia National Laboratories Report to Japanese Nuclear Fuel Cycle (JNC) Report, Submitted to JNC, April, 1999, 40 pp (un-refereed report).

Saad, Y., 1996, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston.

Salinger, A.G., K.D. Devine, G.L. Hennigan, H.K. Moffat, S.A. Hutchinson, and J.N. Shadid, 1996, *MPSalsa: A Finite Element Computer Program for Reacting Flow Problems, Part 2 - User's Guide*, SAND96-2331, Sandia National Laboratories, Albuquerque, New Mexico, 152 pp.

Scheidegger, A. E., 1974, *The physics of Flow Through Porous Media*, 3rd Ed., University of Toronto Press, Buffalo, NY, 353 pp.

Schoof, L.A., and V.R. Yarberry, 1992, *EXODUSII: A Finite Element Data Model*, SAND92-2137, Sandia National Laboratories, Albuquerque, New Mexico, 170 pp. + Appendices.

Shadid, J.N., H.K. Moffat, S.A. Hutchinson; G.L. Hennigan, K.D. Devine, A.G. Salinger, 1996, *MPSalsa : A Finite Computer Program for Reacting Flow Problems, Part 1 - Theoretical development*, SAND95-2752, Sandia National Laboratories, Albuquerque, New Mexico, 81 pp..

Shadid, J.N., A.G. Salinger, R.C. Schmidt, T.M. Smith, S.A. Hutchinson, G.L. Hennigan, K.D. Devine, and H.K. Moffat, 1999, *MPSalsa Version 1.5 : A Finite Element Computer Program for Reacting Flow Problems*, SAND98-2864, Sandia National Laboratories, Albuquerque, New Mexico, 62 pp..

Shan, C., R.W. Falta, and I. Javandel, 1992, Analytical solutions for steady state gas flow to a soil vapor extraction well, *Water Resour. Res*., **28**(4), 1105-1120.

Sjaardema, G. D., 1997, APREPO: A algebraic preprocessor for parameterizing finite element analyses, *SAND92-2291*, Sandia National Laboratories, Albuquerque, New Mexico, 58 pp.

Tezduyar, T. E., 1992, Stabilized Finite Element Formulations for Incompressible Flow Computations, *Advances in App. Mech.*, **28**, 1-44.

Tuminaro, R. S., M. Heroux, S. A. Hutchinson, and J. N. Shadid, 1999, Official Aztec User's Guide, Version 2.1, *SAND99-8801J*, Sandia National Laboratories, Albuquerque, New Mexico, 63 pp..

Udell, K.S., and J.S. Fitch, 1985, Heat and mass transfer in capillary porous media considering evaporation, condensation, and non-condensible gas effects, preesented at the 23rd ASME/AICHE National Heat Transfer Conference, Denver, CO.

van Genuchten, R., 1978, Calculating the Unsaturated Hydraulic Conductivity with a New Closed Form Analytical Model, *Water Resources Bulletin*, Princeton University Press, Princeton University, Princeton, NJ.

Van Wylen, G. J., and Sonntag, R. E., 1976, *Fundamentals of Classical Thermodynamics*, 2nd Ed., John Wiley and Sons, New York, 718 pp.

Zyvoloski, G. A., B. A. Robinson, Z. V. Dash, and L. L. Trease, 1995, Models and Methods Summary for the FEHMN Application, Report No. LA-UR-94-3787, Rev. 1., Los Alamos National Laboratory, Los Alamos, NM.

Appendix A: Installing and Executing PorSalsa

A very comprehensive makefile for PorSalsa is included with a source code distribution. This makefile, which contains directives to the **make** utility, allows executables to be constructed for a variety of machines and operating systems. Because available computing hardware and operating systems change more frequently than documentation, this section describes the overall philosophy and elements of the procedure, but will refrain from details specific to machines. The user will need to become familiar with the current makefile configuration.

The source code is maintained in a number of subdirectories. The ../Salsa subdirectory contains the main makefile(s). Within this subdirectory are those that hold source code (../Salsa/ds, /el, /md, /pe, /ps, and /rf). Other subdirectories (e.g., ../Salsa/Obj_sol, /Obj_dec) will contain a makefile and, during a "make", are targets for object code, and the resulting executable named "salsa" for the specified platform. Both ANSI C and Fortran compilers are invoked; compiler options can be controlled either as parameters to the "make", or by directly editing the appropriate makefile.

PorSalsa makes use of a number of libraries, including MPI and its associated include files, AZTEC, EXODUS, NEMESIS, and NETCDF. Others may also be needed for particular applications. The paths to these libraries must be defined for the platform of interest. Again, the user will need to become familiar with the current make operation.

Once an executable has been created, the user may invoke it in serial mode by simply entering the executable name. Unless otherwise specified, the default input file name of "input-salsa" will be assumed. That input file will be read to define the other files needed for the simulation. If the code is to be executed in a parallel mode, the user will need to invoke MPI according to the requirements of the specific platform. Typically, the command is of the following form:

>mpirun –np *integer* [–machinefile *mach-file-name*] *executable-name* > *output-file*

where

integer	Number of processors to use (default of 1). This number must match that specified by the `Number of processors = integer` data line of the `"General Problem Specifications"` section of the PorSalsa input file.
mach-file-name	File specifying machine names, if execution is to occur on machines other than just the host machine (default is host machine).
executable-name	PorSalsa executable name; include pathname if not in the current subdirectory.
output-file	Directs screen writes to this file (default is the screen).

Appendix B: Matrix Compressibility and Storativity

PorSalsa does not have a direct means of specifying a storage coefficient. Instead, this can be accomplished by properly specifying a formation porosity and compressibility. The parameter `MATRIX_COMPRESS` in the "`Material ID Specifications`" section of the main input file specifies the compressibility of the porous medium. It is defined as

$$C_r = \frac{1}{\phi}\frac{d\phi}{dP} \tag{B.1}$$

where ϕ is the porosity and P is the system (i.e., total) pressure.

The mass storage term for a water-saturated flow equation, when written in terms of pressure, is given by (c.f. *Bear*, 1979, Eqns. 5-19, 5-20),

$$\frac{\partial(\rho\phi)}{\partial t} = \rho\left(\alpha(1-\phi)+\phi\kappa_T\right)\frac{\partial P}{\partial t} \tag{B.2}$$

where the matrix (α) and fluid compressibilities are defined by (Eqn. 5-7 of *Bear*, 1979),

$$\alpha = \frac{1}{(1-\phi)}\frac{\partial\phi}{\partial P} \quad \kappa_T = \frac{1}{\rho}\frac{\partial\rho}{\partial t},$$

respectively. The mass storage term in PorSalsa is formulated in terms of the effective water density (nearly equal the liquid water density under water-saturated conditions)

$$\frac{\partial(\rho_w\phi)}{\partial t} = \phi\left(1+\frac{C_r}{\kappa_T}\right)\frac{\partial\rho_w}{\partial t},$$

See for example Sec. 5.3.2 of *Scheidegger*, 1974. Now, in order to convert this in terms of pressure, we introduce the equation of state used to relate density and pressure,

$$P = P_{sat} + \kappa_P^{-1}(\rho-\rho_{sat}),$$

where the subscript refers to thermodynamically saturated properties and the constant $\kappa_P^{-1} = 2.17\times10^6$ Pa-m^3/kg is used in the code. Converting in terms of pressure, using the equation of state, one gets,

$$\frac{\partial(\rho_w\phi)}{\partial t} = \rho\phi\left(\frac{C_r}{\rho}\frac{\kappa_P}{\kappa_T}+\frac{\kappa_P}{\rho}\right)\frac{\partial P}{\partial t}. \tag{B.3}$$

The saturated liquid density of water at 25 ^{o}C is 997.009 (*Van Wylen & Sonntag*, 1976), and the code presently uses $\kappa_T = 4.4\times10^{-10}$ Pa^{-1} (*Freeze & Cherry*, 1979, p. 55), so at 25 ^{o}C, the ratio $\kappa_P/\rho\kappa_T = 1.05$. Thus, comparing Eqn. (B.1) with (B.3) to a close approximation,

$$\phi C_r = (1-\phi)\alpha \text{.} \qquad \text{(B.4)}$$

The specific volumetric storativity is defined by (*Bear,* 1979, Eqn. 5-23)

$$\rho g \left(\alpha (1-\phi) + \phi \kappa_T \right),$$

when the flow equation is written in terms of head. Hence using the value of formation compressibility from Eqn (B.4) will give a storativity in PorSalsa of approximately the same value, see Eqn (B.3). The variable density in the PorSalsa formulation prevents specifying the exact same storativity at every location in the simulation. However, since the density changes are very small, a very close approximation is obtained by calculating the effective formation compressibility from the formula,

$$\frac{S}{\rho g} = \phi \left(\frac{C_r}{\rho} \frac{\kappa_P}{\kappa_T} + \frac{\kappa_P}{\rho} \right) \qquad \text{(B.5)}$$

where S is the specific storage (for a head based flow equation) and we have divided by ρg to put it into a pressure-based flow equation formulation. ρ is an average water density for the problem. This formula can be used to compute a formation compressibility (C_r) for PorSalsa given a specific storage value for the formation.

Appendix C: Hydraulic Head Specifications

Hydraulic head may be specified as a boundary condition, initial condition, or auxiliary output variable. If conditions are fully saturated then a unique relationship exists between hydraulic head and water density. The formulation is as follows:

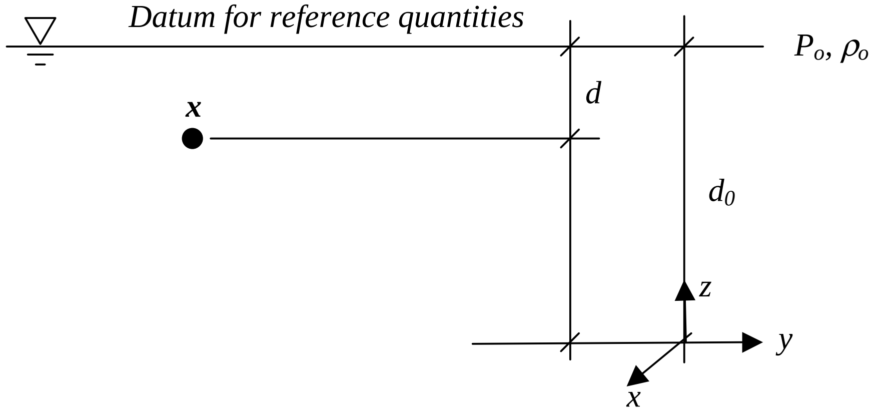


Figure C.1: Formulation of hydraulic head at a point below the water table.

The hydraulic head for a compressible fluid (Hubbert potential, see *Bear* 1972) at a point below the water table is defined as

$$h(\mathbf{x}) - h_0 = d - d_0 + \int_{P_o}^{P} \frac{dP}{\rho(P)\,g} \tag{C.1}$$

where d denotes the depth from the datum to the point $\boldsymbol{x}$. The datum is specified by the REF_PLANE parameter discussed in the ***Material ID Specifications*** section on page 32. This parameter specifies the reference plane in the form

$$Ax + By + Cz + D = 0,$$

with respect to the mesh coordinate system. We recall that a normal vector to the plane is given by (A,B,C) and so the distance to this plane from a point (x,y,z) is given by the inner product of the position vector $\boldsymbol{x}$ with the *unit* normal vector to the reference plane, resulting in the formula,

$$d = \frac{Ax + By + Cz + D}{\sqrt{A^2 + B^2 + C^2}}. \tag{C.2}$$

In PorSalsa the equation of state relating fluid pressure and density in compressed liquid states is given as

$$P = P_{sat} + (\rho - \rho_{sat})\,\kappa_P^{-1} \quad or \quad \rho = \rho_{sat} + (P - P_{sat})\,\kappa_P \tag{C.3}$$

where $\kappa_P^{-1} = 2.17\times10^6$ Pa-m^3/kg in PorSalsa, and where the subscript, *sat*, indicates thermodynamically-saturated conditions. Using (C.3) in (C.1) and performing the integration gives the following relation between head, depth from the datum and pressure,

$$h(P) - h_0 = d - d_0 + \frac{1}{g\kappa_P}\ln\left[\frac{\rho_{sat} + (P - P_{sat})\kappa_P}{\rho_{sat} + (P_o - P_{sat})\kappa_P}\right] \tag{C.4}$$

or, in terms of density (substituting the EOS),

$$h(\rho) = h_0 + d - d_0 + \frac{1}{g\kappa_P}\ln\left(\frac{\rho}{\rho_o}\right) \tag{C.5}$$

where,

$$\rho_o = \rho_{sat} + (P_o - P_{sat})\kappa_P$$

Eqn. (C.5) may be rearranged to express the fluid density as a function of head:

$$\rho(h) = \rho_o \exp\left[g\kappa_P\left((h - h_0) - (d - d_0)\right)\right] \tag{C.6}$$

Equations (C.5) and (C.6) may then be used to relate hydraulic head to water density and *vice versa*. PorSalsa models water in saturated regions as a compressible fluid. For simplicity, when using hydraulic head for initial conditions or boundary conditions, PorSalsa uses $P_o = P_{sat}$ (and therefore $\rho_o = \rho_{sat}$), which corresponds to setting the reference plane at the water table (even if the water table is outside the boundaries of the computational domain). Note also that this means the reference plane will be *above* the saturated region. The head value at the reference plane (h_0) can be freely chosen. When specifying an initial head, which is useful for specifying a hydrostatic initial condition, the head is everywhere equal the reference value, hence the density distribution is computed from (c.f. Eqn. C.6),

$$\rho(h) = \rho_{sat} \exp\left[g\kappa_P(d_0 - d)\right]$$

which illustrates that $d_0 \geq d$ in order for $\rho \geq \rho_{sat}$ as required for a saturated region.

To summarize**, when using hydraulic head for initial conditions or boundary conditions** the following should be noted:

- The implementation in PorSalsa uses $P_o = P_{sat}$ ($\rho_o = \rho_{sat}$).
- The reference plane defined by the `REF_PLANE` parameter in the "`Material ID Specifications`" section of the main input file must be specified at the water table location (which could be outside the mesh boundaries).

Appendix D: Atmospheric Boundary Conditions

The formulation and implementation of a boundary condition that emulates a boundary exposed to the atmosphere is described here. This boundary type is suitable for surfaces exposed to a normal range of atmospheric conditions with temperatures above the freezing point. It is also suitable for the simulation of seepage boundaries where liquid water may flow out of an exposed surface if the gradient driving advective flow of liquid is sufficient.

Formulation

The formulation for the atmospheric boundary conditions is based on comparing the state at a surface node with that of a reference point in the adjacent atmosphere. Gradients in phase pressures, mass fractions, and temperature between the points are used to compute the fluxes of water, air, and energy normal to the surface. These fluxes are then applied as a Neumann boundary condition to each of the respective equations. Since the state of the surface nodes varies with time, this results in a transient specified-flux boundary condition. The primary advantage of this approach is that a specific pressure, saturation, or temperature is not artificially applied to the surface node. Rather, the conditions at the surface are allowed to equilibrate with the adjacent atmospheric conditions. These influences propagate naturally into the interior of the porous domain.

The surface fluxes are computed from the governing equations in PorSalsa for the flow of water, air, and energy. In this manner, the reference point can be thought of as being located at the external boundary of an additional layer of porous media whose material properties match those of the surface node (See Figure D.1). Given that Δs is small, this does not represent a significant change to the physical shape of the original domain. Since variations in Δs only serve to modify the magnitude of the driving gradients rather than their sign, the value chosen for Δs only influences the speed with which the boundary node responds to the applied atmospheric boundary condition.

Reference Point

The reference point is located at a user-specified distance, Δs, from the surface and its thermodynamic state is determined by specifying a temperature (T_{ref}), pressure (P_{ref}), and relative humidity (R_H). Given these quantities, the remaining set of reference parameters required for the atmospheric boundaries can be computed. These are:

$$\rho_{g,ref},\ Y_{wg,ref},\ Y_{ag,ref},\ h_{ag,ref},\ h_{wg,ref},\ h_{g,ref}\ .$$

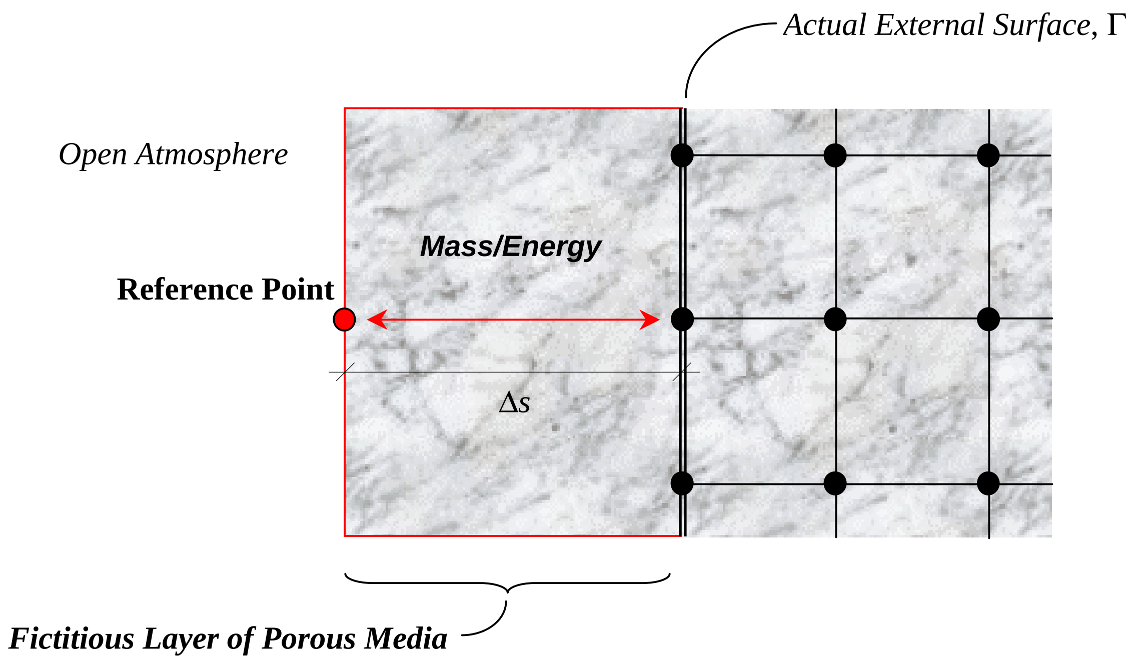


Figure D.1: Schematic of surface node with respect to the reference point.

The partial pressure of water vapor can be calculated from the definition of the relative humidity:

$$P_{wg} = R_H P_{wg,sat} \tag{D.1}$$

where

$$P_{wg,sat} = P_{wg,sat}\left(T_{ref}\right)_{EOS}. \tag{D.2}$$

The density of water vapor can then be computed as

$$\rho_{wg} = \frac{P_{wg}}{R_v T_{ref}} \tag{D.3}$$

where R_v is the gas constant for water vapor (287 J/kg·K / 0.622). Likewise, the density of air in the gas phase is given as

$$\rho_{ag} = \frac{P_{ref} - P_{wg}}{R_d T_{ref}} \tag{D.4}$$

where R_d is the gas constant for dry gas (287 J/kg·K). The density of the gas phase is the sum of the densities of each component in the gas phase:

$$\rho_{g,ref} = \rho_{wg} + \rho_{ag} \tag{D.5}$$

This allows the mass fractions of the components in the gas phase to be computed:

$$Y_{wg,ref} = \frac{\rho_{wg}}{\rho_{g,ref}}, \tag{D.6}$$

$$Y_{ag,ref} = \frac{\rho_{ag}}{\rho_{g,ref}} \tag{D.7}$$

If equations of state are used to compute the enthalpies of each component in the gas phase, *i.e.*,

$$h_{\alpha g,ref} = h_{\alpha g}\left(T_{ref}, P_{ref}\right)_{EOS}, \tag{D.8}$$

then the total enthalpy of the gas phase can be computed as

$$h_{g,ref} = h_{ag,ref}Y_{ag,ref} + h_{wg,ref}Y_{wg,ref}. \tag{D.9}$$

Water-Component Flux

The normal component of the mass flux of water through a boundary exposed to the atmosphere can be written as the sum of the normal components of the advective and diffusive fluxes in each phase (*gas* and *liquid*).

$$\underline{q}_w \cdot \underline{n}\Big|_\Gamma = \left(Y_{wl}\,\underline{f}_l + \underline{J}_{wl} + Y_{wg}\,\underline{f}_g + \underline{J}_{wg}\right)\cdot \underline{n} \tag{D.10}$$

If the atmosphere is comprised of gas only, then it is reasonable to assume that the diffusive flux in the liquid phase is negligible so that $\underline{J}_{wl} \approx 0$. Then, substitution of the generalized extension of Darcy's Law for the advective fluxes and the standard expression for the diffusive flux in the gas phase gives

$$\begin{aligned}\underline{q}_w \cdot \underline{n}\Big|_\Gamma = \Bigg[& -Y_{wl}\frac{\rho_l k_{rl}}{\mu_l}\underline{\underline{k}}\cdot\left(\nabla P_l + \rho_l\,\underline{g}\right) \\ & -Y_{wg}\frac{\rho_g k_{rg}}{\mu_g}\underline{\underline{k}}\cdot\left(\nabla P_g + \rho_g\,\underline{g}\right) - \rho_g D_{wg}\nabla Y_{wg}\Bigg]\cdot\underline{n}\ .\end{aligned} \tag{D.11}$$

The reference pressure (P_{ref}) and mass fraction of water in the gas phase($Y_{wg,ref}$) may be used to approximate the gradients of these variables normal to the boundary:

$$\nabla P_\beta \cdot \underline{n} = \frac{\partial P_\beta}{\partial n} \approx -\frac{P_\beta - P_{ref}}{\Delta s} \qquad (\beta = l, g) \tag{D.12}$$

and,

$$\frac{\partial Y_{wg}}{\partial n} \approx -\frac{Y_{wg} - Y_{wg,ref}}{\Delta s}. \tag{D.13}$$

For an orthotropic material it is also necessary to approximate a directional permeability normal to the boundary. Bear (1972, Section 5.6.2) gives a general formulation. Here we make an approximation and define the permeability normal to the surface as $k_n = k_{xx} n_x + k_{yy} n_y + k_{zz} n_z$. Substitution into (D.12) yields:

$$\begin{aligned} \underline{q}_w \cdot \underline{n}\Big|_\Gamma = &-Y_{wl} \frac{\rho_l k_{rl}}{\mu_l} k_n \left(-\frac{P_l - P_{ref}}{\Delta s} + \rho_l \underline{g} \cdot \underline{n} \right) \\ &-Y_{wg} \frac{\rho_g k_{rg}}{\mu_g} k_n \left(-\frac{P_g - P_{ref}}{\Delta s} + \rho_g \underline{g} \cdot \underline{n} \right) \\ &-\rho_g D_{wg} \left(-\frac{Y_{wg} - Y_{wg,ref}}{\Delta s} \right) \end{aligned} \tag{D.14}$$

Assumption that the permeability tensor is orthotropic gives the general form of the normal component of the total mass-flux of water through the boundary:

$$\underline{q}_w \cdot \underline{n}\Big|_\Gamma = -\tilde{Y}_{wl} \frac{\tilde{\rho}_l k_{rl}}{\mu_l} A - \tilde{Y}_{wg} \frac{\tilde{\rho}_g k_{rg}}{\mu_g} B - \tilde{\rho}_g D_{wg} C \tag{D.15}$$

where *A*, *B* and *C* are the driving gradients defined in Eqn. (D.14).

The densities multiplying the volumetric advective and diffusive fluxes have been denoted $\tilde{Y}_{wl} \tilde{\rho}_l$, $\tilde{Y}_{wg} \tilde{\rho}_g$, and $\tilde{\rho}_g$ respectively. The overbar reflects the need to discriminate between outward and inward fluxes. The sign of the driving force determines the direction of the flux, as follows:

$A > 0$	Outward Advective Liquid Flux
$A < 0$	Inward Advective Liquid Flux
$B > 0$	Outward Advective Gas Flux
$B < 0$	Inward Advective Gas Flux
$C > 0$	Outward Diffusive Flux of Water Vapor
$C < 0$	Inward Diffusive Flux of Water Vapor

If the flux is outward then the density associated with the porous medium is used; otherwise, the reference quantity is appropriate. Table D.1 gives a summary of the appropriate expressions for each case.

Flux	***Phase***	***Direction***	***Appropriate Multiplier***
Advective	Liquid	Outward	$Y_{wl}\rho_l$
"	"	Inward	0
Advective	Gas	Outward	$Y_{wg}\rho_g$
"	"	Inward	$Y_{wg,ref}\rho_{g,ref}$
Diffusive	Gas	Outward	ρ_g
"	"	Inward	$\rho_{g,ref}$

Table D.1: Appropriate mass fraction - density expressions for the inward and outward components of the normal flux of water.

Air-Component Flux

The normal component of the mass flux of air through a boundary exposed to the atmosphere can be written as the sum of the normal components of the advective and diffusive fluxes in each phase (*gas* and *liquid*).

$$\underline{q}_a \cdot \underline{n}\Big|_\Gamma = \left(Y_{al}\,\underline{f}_l + Y_{ag}\,\underline{f}_g + \underline{J}_{ag}\right)\cdot \underline{n} \tag{D.16}$$

As in the governing equation, the diffusive flux in the liquid phase is ignored. Substitution of the generalized extension of Darcy's Law for the advective fluxes and the standard expression for the diffusive flux in the gas phase gives

$$\begin{aligned}\underline{q}_a \cdot \underline{n}\Big|_\Gamma = \Bigg[&-Y_{al}\frac{\rho_l k_{rl}}{\mu_l}\underline{\underline{k}}\cdot\left(\nabla P_l + \rho_l\,\underline{g}\right)\\ &-Y_{ag}\frac{\rho_g k_{rg}}{\mu_g}\underline{\underline{k}}\cdot\left(\nabla P_g + \rho_g\,\underline{g}\right) - \rho_g D_{ag}\nabla Y_{ag}\Bigg]\cdot\underline{n}\,,\end{aligned} \tag{D.17}$$

As with the water flux, let a reference mass fraction ($Y_{ag,ref}$) be assigned beyond the limits of the domain at a *normal* distance of Δs from the boundary. This value may be used to approximate the gradient of the mass fraction of air at the boundary:

$$\frac{\partial Y_{ag}}{\partial n} \approx -\frac{Y_{ag} - Y_{ag,ref}}{\Delta s}. \tag{D.18}$$

This leads to the following expression for the normal component of the total mass-flux of air through the boundary:

$$\underline{q}_a \cdot \underline{n}\Big|_\Gamma = -\tilde{Y}_{al}\frac{\tilde{\rho}_l k_{rl}}{\mu_l}A - \tilde{Y}_{ag}\frac{\tilde{\rho}_g k_{rg}}{\mu_g}B - \tilde{\rho}_g D_{ag} D \tag{D.19}$$

where the coefficients depend on the type of problem being solved. Coefficients A and B are identical to those given for the water flux while

$$D = \frac{\partial Y_{ag}}{\partial n}. \tag{D.20}$$

The densities multiplying the volumetric advective and diffusive fluxes have been denoted $\tilde{Y}_{al}\tilde{\rho}_l$, $\tilde{Y}_{ag}\tilde{\rho}_g$, and $\tilde{\rho}_g$ respectively. Again, overbar reflects the need to discriminate between outward and inward fluxes. The sign of the driving force determines the direction of the flux:

$A > 0$	Outward Advective Liquid Flux
$A < 0$	Inward Advective Liquid Flux
$B > 0$	Outward Advective Gas Flux
$B < 0$	Inward Advective Gas Flux
$D > 0$	Outward Diffusive Flux of Air
$D < 0$	Inward Diffusive Flux of Air

If the flux is outward then the density associated with the porous medium is used; otherwise the reference quantity is appropriate. Table D.2 gives a summary of the appropriate expressions for each case.

Flux	***Phase***	***Direction***	***Appropriate Multiplier***
Advective	Liquid	Outward	$Y_{al}\rho_l$
"	"	Inward	0
Advective	Gas	Outward	$Y_{ag}\rho_g$
"	"	Inward	$Y_{ag,ref}\rho_{g,ref}$
Diffusive	Gas	Outward	ρ_g
"	"	Inward	$\rho_{g,ref}$

Table D.2: Appropriate mass fraction - density expressions for the inward and outward components of the normal flux of air.

Energy Flux

The normal component of the energy flux through a boundary exposed to the atmosphere can be written as the sum of the normal components of the advective and diffusive fluxes in each phase (*gas* and *liquid*) together with the conductive flux of energy.

$$\underline{q}_e \cdot \underline{n}\Big|_\Gamma = \left(h_l\, \underline{f}_l + h_g\, \underline{f}_g + h_{wg}\, \underline{J}_{wg} + h_{ag}\, \underline{J}_{ag} - \lambda_T \nabla T\right)\cdot \underline{n} \tag{D.21}$$

As in the previous expressions, the diffusive fluxes in the liquid phase are ignored for each component. Substitution of the generalized extension of Darcy's Law for the advective fluxes and standard expressions for the diffusive fluxes in the gas phase gives

$$\begin{aligned}\underline{q}_e \cdot \underline{n}\Big|_\Gamma = \Bigg[&- h_g \frac{\rho_g k_{rg}}{\mu_g} \underline{\underline{k}} \cdot \left(\nabla P_g + \rho_g\, \underline{g}\right) - h_l \frac{\rho_l k_{rl}}{\mu_l} \underline{\underline{k}} \cdot \left(\nabla P_l + \rho_l\, \underline{g}\right) \\ &- h_{wg} \rho_g D_{wg} \nabla Y_{wg} - h_{ag} \rho_g D_{ag} \nabla Y_{ag} - \lambda_T \nabla T \Big] \cdot \underline{n}\ ,\end{aligned} \tag{D.22}$$

Let a reference temperature (T_{ref}) be assigned beyond the limits of the domain at a *normal* distance of Δs from the boundary. This value may be used to approximate the gradient of the temperature the boundary:

$$\frac{\partial T}{\partial n} \approx -\frac{T - T_{ref}}{\Delta s}\,. \tag{D.23}$$

This leads to the following expression for the normal component of the total mass-flux of energy through the boundary:

$$\underline{q}_e \cdot \underline{n}\Big|_\Gamma = -\tilde{h}_l \frac{\tilde{\rho}_l k_{rl}}{\mu_l} A - \tilde{h}_g \frac{\tilde{\rho}_g k_{rg}}{\mu_g} B - \tilde{h}_{lg} \tilde{\rho}_g D_{lg} C - \tilde{h}_{ag} \tilde{\rho}_g D_{ag} D - \lambda_T E \tag{D.24}$$

where the coefficients depend on the type of problem being solved. Coefficients *A*, *B*, *C*, and *D* are identical to those given for the water and air fluxes, while

$$E = \frac{\partial T}{\partial n} \tag{D.25}$$

The enthalpies and densities multiplying the volumetric advective and diffusive fluxes have been denoted $\tilde{h}_g \tilde{\rho}_g$, $\tilde{h}_l \tilde{\rho}_l$, $\tilde{h}_{ag} \tilde{\rho}_g$, and $\tilde{h}_{wg} \tilde{\rho}_{wg}$, respectively. Again, the overbar reflects the need to discriminate between outward and inward fluxes. The sign of the driving force determines the direction of the flux:

$A > 0$	Outward Advective Liquid Flux
$A < 0$	Inward Advective Liquid Flux
$B > 0$	Outward Advective Gas Flux
$B < 0$	Inward Advective Gas Flux

$C > 0$	Outward Diffusive Flux of Water Vapor
$C < 0$	Inward Diffusive Flux of Water Vapor
$D > 0$	Outward Diffusive Flux of Air
$D < 0$	Inward Diffusive Flux of Air
$E > 0$	Outward Conductive Flux
$E < 0$	Inward Conductive Flux

If the flux is outward then the enthalpy-density product associated with the porous medium is used; otherwise the reference value is appropriate. Table D.3 gives a summary of the appropriate enthalpy - density expressions for each case.

Flux	***Phase / Component***	***Direction***	***Appropriate Multiplier***
Advective	Liquid	Outward	$h_l \rho_l$
"	"	Inward	0
Advective	Gas	Outward	$h_g \rho_g$
"	"	Inward	$h_{g,ref} \rho_{g,ref}$
Diffusive	Gas /Air	Outward	$h_{ag} \rho_g$
"	"	Inward	$h_{ag,ref} \rho_{g,ref}$
Diffusive	Gas / Water	Outward	$h_{wg} \rho_g$
"	"	Inward	$h_{wg,ref} \rho_{g,ref}$

Table D.3: Appropriate enthalpy – density expression for the inward and outward components of the normal fluxes of energy.

Comments

A. This implementation is only valid for "typical" ranges of atmospheric conditions since the equations of state for the atmospheric reference point are not valid for critical regions (freezing, steam, *etc*.)

B. It is also necessary to specify the reference parameters such that they uniquely specify the thermodynamic state in the atmosphere. In this sense, pressures, temperatures, mass fractions, and densities must be consistent with a single state.

In considering the normal advective flux of water through a boundary exposed to the atmosphere (*i.e.*, a "Seepage Boundary"):

C. If the porous medium is partially saturated and if the atmospheric pressure is zero, then the water pressure will be negative. This will yield a negative pressure gradient. As well, if the normal direction has an upward component then the contribution to the driving gradient from gravitational forces will be negative. The sum of these terms will result in a *net* negative driving gradient and will result in an *inward flux*. In this case the density term multiplying the volumetric advective flux is zero (Table D.1), and will lead to a shutoff of flow into the system from the atmosphere.

D. Conversely, if the gravitational contribution to the driving force is sufficient to counteract the negative pressure gradient (say, in the ceiling in an excavated shaft where the normal direction is aligned with gravity), the outward flux will be mitigated by the low (presumably) relative permeability. In this sense, the advective flux of water out of the system *can* occur under unsaturated conditions, but only under rare geometrical circumstances and at low flow rates (*i.e.*, less than saturated).

E. Under saturated conditions the water pressure should be positive. Given a zero (gage) reference pressure this will result in a positive outward pressure gradient. If this is sufficient to counterbalance the gravitational contribution to the driving gradient then outward flow will occur. This will allow outward flow to occur along a seepage face, which is defined as the portion of an atmospheric boundary between the water table (defined by $P_w = 0$) and a free water body.

Implementation

Atmospheric boundaries have been implemented to simulate a boundary in contact with the atmosphere composed of air and water vapor at a given temperature and pressure. Full implementation mimics flow through porous media with the driving gradients computed as a function of the contrast of the atmospheric conditions with the conditions at a surface quadrature point.

These boundary conditions can also be used to assign "penalty" boundaries for each of the three equations with coefficients specified by the user or generated from primary and/or secondary variables in user-defined functions.

Atmospheric Reference Conditions

The user specifies the thermodynamic state of a reference point located at a normal distance Δs from the quadrature points along the boundary surface. The reference state is specified by the following quantities:

$$P_{ref},\ T_{ref},\ R_H$$

where R_H is the relative humidity. Given these quantities, the remaining set of reference parameters required for the atmospheric boundaries can be computed. These are:

$$\rho_{g,ref},\ Y_{wg,ref},\ Y_{ag,ref},\ h_{ag,ref},\ h_{wg,ref},\ h_{g,ref}\ .$$

The calculation of these values was discussed in the preceeding.

Water-Component Flux Implementation

$$\underline{q}_w \cdot \underline{n}\Big|_\Gamma = -Term\ 1 - Term\ 2 - Term\ 3$$

where:

Term 1: Advection of Water Component in Liquid Phase

$$Term\ 1 = flag \cdot coefficient_1 \cdot gradient_1$$

$$flag = \begin{cases} Option\ 1:\ Outward\ Seepage\ can\ occur\ when\ S < 1 \\ \qquad\qquad = 1 \\ Option\ 2:\ Outward\ Seepage\ can\ occur\ only\ when\ S = 1 \\ \qquad\qquad = 1\ \ if\ \ S = 1 \\ \qquad\qquad = 0\ \ Otherwise \end{cases}$$

$$coefficient_1 = \begin{cases} Option\,1:\; = \tilde{Y}_{wl}\,\dfrac{\tilde{\rho}_l k_{rl}}{\mu_l} \\ Option\,2:\; = User\ Supplied\ Constant \\ Option\,3:\; = User\ Supplied\ Function \end{cases}$$

$$gradient = \begin{cases} if\quad coefficient = Option\ \ 1: \\ = k_n\left[\left(-\dfrac{P_l - P_{ref}}{\Delta s}\right) + \rho_l\,\underline{g}\cdot\underline{n}\right] \\ if\quad coefficient = Option\ \ 2\ \ or\ \ Option\ \ 3: \\ = \left(P_l - P_{ref}\right) \end{cases}$$

$$\tilde{Y}_{wl}\,\tilde{\rho}_l = \begin{cases} Y_{wl}\,\rho_l & if\ \ gradient_1 > 0 \\ 0 & if\ \ gradient_1 < 0 \end{cases}$$

Note: This term is only nonzero for simulations involving liquid-phase transport.

Term 2: Advection of Water Component in Gas Phase

$$Term\ 2 = coefficient \cdot gradient$$

$$coefficient_2 = \begin{cases} Option\,1:\; = \tilde{Y}_{wg}\,\dfrac{\tilde{\rho}_g k_{rg}}{\mu_g} \\ Option\,2:\; = User\ Supplied\ Constant \\ Option\,3:\; = User\ Supplied\ Function \end{cases}$$

$$gradient_2 = \begin{cases} if\quad coefficient_2 = Option\ \ 1: \\ = k_n\left[\left(-\dfrac{P_g - P_{ref}}{\Delta s}\right) + \rho_g\,\underline{g}\cdot\underline{n}\right] \\ if\quad coefficient_2 = Option\ \ 2\ \ or\ \ Option\ \ 3: \\ = \left(P_g - P_{ref}\right) \end{cases}$$

$$\tilde{Y}_{wg}\,\tilde{\rho}_g = \begin{cases} Y_{wg}\,\rho_g & if\ \ gradient_2 > 0 \\ Y_{wg,ref}\,\rho_{g,ref} & if\ \ gradient_2 < 0 \end{cases}$$

Note: This term is only nonzero for simulations involving gas phase-transport.

Term 3: Diffusion of Water Component in Gas Phase

$$Term\ 3 = coefficient_3 \cdot gradient_3$$

$$coefficient_3 = \begin{cases} Option\ 1:\ = \tilde{\rho}_g D_{wg} \\ Option\ 2:\ = User\ Supplied\ Constant \\ Option\ 3:\ = User\ Supplied\ Function \end{cases}$$

$$gradient_3 = \begin{cases} if\ \ coefficient_3 = Option\ \ 1: \\ = -\dfrac{\left(Y_{wg} - Y_{wg,ref}\right)}{\Delta s} \\ if\ \ coefficient_3 = Option\ \ 2\ \ or\ \ Option\ \ 3: \\ = \left(Y_{wg} - Y_{wg,ref}\right) \end{cases}$$

$$\tilde{\rho}_g = \begin{cases} \rho_g & if\ gradient_3 > 0 \\ \rho_{g,ref} & if\ gradient_3 < 0 \end{cases}$$

Note: This term is only nonzero for simulations involving gas-phase transport *and* binary diffusion.

Usage

```
BC                        = WD_BC ATMOSPHERIC SS 10 DEPENDENT
```

```
UNSATURATED_SEEPAGE  DEFAULT  DEFAULT  DEFAULT  1.0  0.96E6  20.  0.8  0
```

Atmospheric boundary applied to the water equation on side set 10, outward liquid advection may occur under unsaturated conditions and Option 1 is used for the coefficient and gradient. Δs = 1.0, P_{ref} = 0.96E6, T_{ref} = 20., $R_{H,ref}$ = 0.8, no extra lines of BC_DATA given.

```
BC                        = WD_BC ATMOSPHERIC SS 10 DEPENDENT
```

```
SATURATED_SEEPAGE  1.0  2.0  3.0  1.0  0.96e6  20.  0.8  0
```

Atmospheric boundary applied to the water equation on side set 10, outward liquid advection may occur only under saturated conditions , Option 2 is used for all the coefficients (*coefficient*$_1$=1.0, *coefficient*$_2$ = 2.0, *coefficient*$_3$ = 3.0), Δs = 1.0, P_{ref} = 0.96E6, T_{ref} = 20., $R_{H,ref}$ = 0.8, no extra lines of BC_DATA given.

Air-Component Flux Implementation

$$\underline{q}_a \cdot \underline{n}\Big|_\Gamma = -Term\ 1 - Term\ 2 - Term\ 3$$

where:

Term 1: Advection of Air Component in Liquid Phase

$$Term\ 1 = flag \cdot coefficient_1 \cdot gradient_1$$

$$flag = \begin{cases} Option\ 1:\ Outward\ Seepage\ can\ occur\ when\ S < 1 \\ \qquad = 1 \\ Option\ 2:\ Outward\ Seepage\ can\ occur\ only\ when\ S = 1 \\ \qquad = 1\ \ if\ \ S = 1 \\ \qquad = 0\ \ Otherwise \end{cases}$$

$$coefficient_1 = \begin{cases} Option\ 1:\ = \tilde{Y}_{al}\, \dfrac{\tilde{\rho}_l k_{rl}}{\mu_l} \\ Option\ 2:\ = User\ Supplied\ Constant \\ Option\ 3:\ = User\ Supplied\ Function \end{cases}$$

$$gradient_1 = \begin{cases} if\ \ coefficient_1 = Option\ \ 1: \\ = k_n \left[\left(-\dfrac{P_l - P_{ref}}{\Delta s} \right) + \rho_l\, \underline{g} \cdot \underline{n} \right] \\ if\ \ coefficient_1 = Option\ \ 2\ \ or\ \ Option\ \ 3: \\ = \left(P_l - P_{ref} \right) \end{cases}$$

$$\tilde{Y}_{al}\, \tilde{\rho}_g = \begin{cases} Y_{al}\, \rho_l & if\ \ gradient_2 > 0 \\ Y_{al,ref}\, \rho_{l,ref} & if\ \ gradient_2 < 0 \end{cases}$$

Note: This term is only nonzero for simulations involving liquid-phase transport.

Term 2: Advection of Air Component in Gas Phase

$$Term\ 2 = coefficient_2 \cdot gradient_2$$

$$coefficient_2 = \begin{cases} Option\ 1:\ = \tilde{Y}_{ag}\, \dfrac{\tilde{\rho}_g k_{rg}}{\mu_g} \\ Option\ 2:\ = User\ Supplied\ Constant \\ Option\ 3:\ = User\ Supplied\ Function \end{cases}$$

$$gradient_2 = \begin{cases} if \quad coefficient_2 = Option \ 1: \\ = k_n \left[\left(-\dfrac{P_g - P_{ref}}{\Delta s} \right) + \rho_g \underline{g} \cdot \underline{n} \right] \\ if \quad coefficient_2 = Option \ 2 \ or \ Option \ 3: \\ = \left(P_g - P_{ref} \right) \end{cases}$$

$$\tilde{Y}_{ag} \tilde{\rho}_g = \begin{cases} Y_{ag} \rho_g & if \ gradient_2 > 0 \\ Y_{ag,ref} \rho_{g,ref} & if \ gradient_2 < 0 \end{cases}$$

Term 3: Diffusion of Air Component in Gas Phase

$$Term\ 3 = coefficient_3 \cdot gradient_3$$

$$coefficient_3 = \begin{cases} Option\,1: \ = \tilde{\rho}_g D_{ag} \\ Option\ 2: \ = User\ Supplied\ Constant \\ Option\ 3: \ = User\ Supplied\ Function \end{cases}$$

$$gradient_3 = \begin{cases} if \quad coefficient_3 = Option \ 1: \\ = -\dfrac{\left(Y_{ag} - Y_{ag,ref} \right)}{\Delta s} \\ if \quad coefficient_3 = Option \ 2 \ or \ Option \ 3: \\ = \left(Y_{ag} - Y_{ag,ref} \right) \end{cases}$$

$$\tilde{\rho}_g = \begin{cases} \rho_g & if \ gradient_2 > 0 \\ \rho_{g,ref} & if \ gradient_2 < 0 \end{cases}$$

Note: This term is only nonzero for simulations involving binary diffusion..

Usage

```
BC                          = PA_BC ATMOSPHERIC SS 10 DEPENDENT

UNSATURATED_SEEPAGE  DEFAULT  DEFAULT  DEFAULT  1.0  0.96E6  20.  0.8  0
```

Atmospheric boundary applied to the air equation on side set 10, outward liquid advection may occur under unsaturated conditions, option 1 is used for the coefficients, Δs = 1.0, P_{ref} = 0.96E6, T_{ref} = 20., $R_{H,ref}$ = 0.8, and no extra lines of BC_DATA given.

```
BC                          = PA_BC ATMOSPHERIC SS 10 DEPENDENT

SATURATED_SEEPAGE  1.0  2.0  3.0  1.0  0.96e6 20.  0.8  0
```

Atmospheric boundary applied to the air equation on side set 10, outward liquid advection may occur only under saturated conditions ,option 2 is used for all the coefficients (*coefficient*$_1$=1.0, *coefficient*$_2$ = 2.0, *coefficient*$_3$ = 3.0), Δs = 1.0, P_{ref} = 0.96E6, T_{ref} = 20., $R_{H,ref}$ = 0.8, and no extra lines of BC_DATA given.

Energy Flux Implementation

$$\underline{q}_e \cdot \underline{n}\Big|_\Gamma = -Term\ 1 - Term\ 2 - Term\ 3 - Term\ 4 - Term\ 5$$

where:

Term 1: Advection of Heat in Liquid Phase

$$Term\ 1 = flag \cdot coefficient_1 \cdot gradient_1$$

$$flag = \begin{cases} Option\ 1:\ Outward\ Seepage\ can\ occur\ when\ S < 1 \\ \qquad\qquad = 1 \\ Option\ 2:\ Outward\ Seepage\ can\ occur\ only\ when\ S = 1 \\ \qquad\qquad = 1\ \ if\ \ S = 1 \\ \qquad\qquad = 0\ \ Otherwise \end{cases}$$

$$coefficient_1 = \begin{cases} Option\ 1:\ = \tilde{h}_l \dfrac{\tilde{\rho}_l k_{rl}}{\mu_l} \\ Option\ 2:\ = User\ Supplied\ Constant \\ Option\ 3:\ = User\ Supplied\ Function \end{cases}$$

$$gradient_1 = \begin{cases} if\ \ coefficient_1 = Option\ \ 1: \\ = k_n \left[\left(-\dfrac{P_l - P_{ref}}{\Delta s} \right) + \rho_l \underline{g} \cdot \underline{n} \right] \\ if\ \ coefficient_1 = Option\ \ 2\ \ or\ \ Option\ \ 3: \\ = \left(P_l - P_{ref} \right) \end{cases}$$

$$\tilde{h}_l \tilde{\rho}_l = \begin{cases} h_l \rho_l & if\ \ gradient_2 > 0 \\ h_{l,ref}\, \rho_{l,ref} & if\ \ gradient_2 < 0 \end{cases}$$

Note: This term is only nonzero for simulations involving liquid-phase transport.

Term 2: Advection of Heat in Gas Phase

$$Term\ 2 = coefficient_2 \cdot gradient_2$$

$$coefficient_2 = \begin{cases} Option\,1: \; = \tilde{h}_g \dfrac{\tilde{\rho}_g k_{rg}}{\mu_g} \\ Option\,2: \; = User\;Supplied\;Constant \\ Option\,3: \; = User\;Supplied\;Function \end{cases}$$

$$gradient_2 = \begin{cases} if \quad coefficient_2 = Option \;\; 1: \\ = k_n \left[\left(-\dfrac{P_g - P_{ref}}{\Delta s} \right) + \rho_g \underline{g} \cdot \underline{n} \right] \\ if \quad coefficient_2 = Option \;\; 2 \;\; or \;\; Option \;\; 3: \\ = \left(P_g - P_{ref} \right) \end{cases}$$

$$\tilde{h}_g \tilde{\rho}_g = \begin{cases} h_g \rho_g & if \;\; gradient_2 > 0 \\ h_{g,ref} \, \rho_{g,ref} & if \;\; gradient_2 < 0 \end{cases}$$

Note: This term is only nonzero for simulations involving gas phase-transport.

Term 3: Diffusion of Heat in Water Component in Gas Phase

$$Term\ 3 = coefficient_3 \cdot gradient_3$$

$$coefficient_3 = \begin{cases} Option\,1: \; = \tilde{h}_{wg} \tilde{\rho}_g D_{ag} \\ Option\,2: \; = User\;Supplied\;Constant \\ Option\,3: \; = User\;Supplied\;Function \end{cases}$$

$$gradient_3 = \begin{cases} if \quad coefficient_3 = Option \;\; 1: \\ = -\dfrac{\left(Y_{wg} - Y_{wg,ref} \right)}{\Delta s} \\ if \quad coefficient_3 = Option \;\; 2 \;\; or \;\; Option \;\; 3: \\ = \left(Y_{wg} - Y_{wg,ref} \right) \end{cases}$$

$$\tilde{h}_{wg} \tilde{\rho}_g = \begin{cases} h_{wg} \rho_g & if \;\; gradient_2 > 0 \\ h_{wg,ref} \, \rho_{g,ref} & if \;\; gradient_2 < 0 \end{cases}$$

Note: This term is only nonzero for simulations involving gas-phase transport *and* binary diffusion..

Term 4: Diffusion of Heat in Air Component in Gas Phase

$$Term\ 4 = coefficient_4 \cdot gradient_4$$

$$coefficient_4 = \begin{cases} Option\,1: \ = \tilde{h}_{ag}\tilde{\rho}_g D_{ag} \\ Option\,2: \ = User\ Supplied\ Constant \\ Option\,3: \ = User\ Supplied\ Function \end{cases}$$

$$gradient_4 = \begin{cases} if \quad coefficient_4 = Option \ \ 1: \\ = -\dfrac{\left(Y_{ag} - Y_{ag,ref}\right)}{\Delta s} \\ if \quad coefficient_4 = Option \ \ 2 \ \ or \ \ Option \ \ 3: \\ = \left(Y_{ag} - Y_{ag,ref}\right) \end{cases}$$

$$\tilde{h}_{ag}\tilde{\rho}_g = \begin{cases} h_{ag}\rho_g & if \ \ gradient_2 > 0 \\ h_{ag,ref}\,\rho_{g,ref} & if \ \ gradient_2 < 0 \end{cases}$$

Note: This term is only nonzero for simulations involving gas-phase transport *and* binary diffusion..

Term 5: Heat Conduction

$$Term\ 5 = coefficient_5 \cdot gradient_5$$

$$coefficient_5 = \begin{cases} Option\,1: \ = \lambda_T \\ Option\,2: \ = User\ Supplied\ Constant \\ Option\,3: \ = User\ Supplied\ Function \end{cases}$$

$$gradient_5 = \begin{cases} if \quad coefficient_5 = Option \ \ 1: \\ = -\dfrac{\left(T - T_{ref}\right)}{\Delta s} \\ if \quad coefficient_5 = Option \ \ 2 \ \ or \ \ Option \ \ 3: \\ = \left(T - T_{ref}\right) \end{cases}$$

Usage

```
BC                          = T_BC ATMOSPHERIC SS 10 DEPENDENT

UNSATURATED_SEEPAGE  DEFAULT  DEFAULT DEFAULT  DEFAULT DEFAULT  1.0  0.96E6  20.  .8  0
```

Atmospheric boundary applied to the energy equation on side set 10, outward liquid advection may occur under unsaturated conditions and Option 1 is used for the coefficient and gradient. Δs = 1.0, P_{ref} = 0.96E6, T_{ref} = 20., $R_{H,ref}$ = 0.8, no extra lines of BC_DATA given.

```
BC                         = T_BC ATMOSPHERIC SS 10 DEPENDENT

SATURATED_SEEPAGE  1.0  2.0  3.0  4.0  5.0  1.0 0.96E6 20. .8 0
```

Atmospheric boundary applied to the energy equation on side set 10, outward liquid advection may occur only under saturated conditions ,option 2 is used for all the coefficients (*coefficient*$_1$=1.0, *coefficient*$_2$ = 2.0, *coefficient*$_3$ = 3.0, *coefficient*$_4$ = 4.0, *coefficient*$_5$ = 5.0), Δs = 1.0, P_{ref} = 0.96E6, T_{ref} = 20., $R_{H,ref}$ = 0.8, and no extra lines of BC_DATA given.

Notation

Component Variables: α = Air (a)/ Water (w)

d_α	Bulk Density of Component α (Mass of Component α / Total Volume)	[M/L^3]
$\underline{F}_\alpha$	Net Mass Flux Vector of Component α (Mass of Component α / Area – Time)	[M/L^{2}T]

Phase Variables: β = Gas (g) / Liquid (l) / Energy (e) / Solid (s)

e_β	Bulk Energy of Phase β (Solid / Gas / Liquid)	[M/LT^2]
$\underline{f}_\beta$	Advective Mass Flux Vector of Phase β	[M/L^{2}T]
h_β	Enthalpy of Phase β (Energy of Phase β / Mass of Phase β)	[M^2/T^2]
$k_{r\beta}$	Relative Permeability of Phase β	[-]
μ_β	Viscosity of Phase β	[M/L-T]
P_β	Pressure of Phase β	[M/LT^2]
$\underline{q}_\beta$	Mass Flux Vector of Phase β	[M/L^{2}T]
ρ_β	Density of Phase β (Solid / Gas / Liquid) (Mass of Phase β / Volume of Phase β)	[M/L^3]
S_β	Saturation of Phase β (Volume of Phase β / Volume of Voids)	[-]
$\underline{v}_\beta$	Volumetric Flux Vector of Phase β	[L^3/L^{2}T]

Mixed Variables

$D_{\alpha\beta}$	Diffusion Coefficient for Component α in Phase β	[L^2/T]
$\underline{F}_{\alpha\beta}$	Net Mass Flux Vector of Component α in Phase β (Mass of α in Phase β / Area – Time)	[M/L^2T]
$h_{\alpha\beta}$	Enthalpy of Component α in the Phase β (Energy of Component α / Mass of Component α)	[M^2/T^2]
$\underline{J}_{\alpha\beta}$	Diffusive Mass Flux Vector of Component α in Phase β (Mass of α in Phase β / Area – Time)	[M/L^2T]
$\rho_{\alpha\beta}$	Concentration (Density) of Component α in Phase β (Mass of Component α / Volume of Phase β)	[M/L^3]
$Y_{\alpha\beta}$	Mass Fraction of Component α in the Phase β (Mass of Component α / Mass of Phase β)	[-]
	Miscellaneous	
ϕ	Porosity (Volume of Voids / Total Volume)	[-]
Γ	Domain Surface	[-]
$\underline{\underline{k}}$	Permeability Tensor	[L^2]
t	Time Coordinate	[T]
T	Temperature	[° Celsius]
e	Bulk Internal Energy	[$M\text{-}L^2/T^2$]
$\underline{n}$	Unit Normal Vector	[-]
λ_T	Heat Capacity	[]

Appendix E: nem_slice Utility Program

Man Page Description

NAME

nem_slice – generate a load-balance file from an EXODUS II geometry file

SYNOPSIS

nem_slice [-h][-V][-a *inpfile*][< -n | -e > -v -m *machine_desc* -l *loadbal_desc* -s *eigensolv_desc*][

-w *weighting_desc*][-o *outfile*][*exoIIfile*]

DESCRIPTION

nem_slice reads in a FEM description of the geometry of a problem from an EXODUS II file, *exoIIfile* , generates either a nodal or elemental graph of the problem, calls *Chaco* to load balance the graph, and outputs a *NemesisI* load-balance file.

OPTIONS

The -h option causes *nem_slice* to print out usage information. If the -h option is specified all other options are ignored.

The -V option causes *nem_slice* to print out the version number. If the -V option is specified all other options are ignored.

If the -a option is specified all parameters will be set from an ASCII input file. Other command line options may be used with -a and such options will override anything respecified in the ASCII input file.

The -n option specifies that a nodal decomposition should be performed on the input FEM mesh, while the -e option specifies that an elemental decomposition should be performed. One of either -e or -n is required.

The -v option is used to indicate that you wish to output an EXODUS II file which will allow you to visualize the load balance results. If the mesh consists of one type of element then the visualization is done by element blocks. If the mesh consists of mixed element types then the visualization is done by assigning nodal results variables to each FEM node in the mesh, the value of which corresponds to the processor ID that node is assigned to.

The -m option is used to describe the parallel architecture for which the load balance will be performed. There are currently two choices for this option, either *mesh* for an n-dimensional mesh architecture, or *hcube* for a hypercube architecture. Both machine type specifications require a sub-option giving the dimension(s) of the machine, and thus the number of processors, for which the load balance is to be performed.

The -l option is used to specify which method to utilize in generating a load balance for the given problem. In addition, a sub-option of kl can be specified if Kernighan-Lin refinement is desired and the main method does not use KL by default. The other sub-option is the number of sections to use (see the Chaco User's Guide for further information). This can be specified with the sub-option num_sects=value. Supported methods for load balancing are:

multikl
spectral
inertial
linear
random
scattered

The -s option is used to specify parameters for the eigensolver, if the method utilizes an eigensolver. Parameters which can be specified are:

tolerance=value
use_rqi
vmax=value

The -o option gives the name of the *NemesisI* output file. By default the output file name is generated from the input file name with different additions depending on the method being used for load balancing and the number of processors for which the load balance was generated.

The -w option is used to specify how to weight the graph of the problem once it is generated. Currently *read*, *eb*, and *edges* sub-options are supported. The *read* sub-option allows the reading in of weights from an EXODUS II file. Nodal values are read for nodal decompositions and elemental values for elemental decompositions. Along with the *read* sub-option other sub-options which can be specified are the name of the variable, either nodal or elemental depending on the method requested, as it exists in the EXODUS II input file. This variable can either be specified via it's name with the *var_name* sub-option. Or it can be specified with it's index via the *var_index* sub-option. In addition the time index from which to read the given variable can also be specified with the *time_index* sub-option. The *eb* sub-option is used to give a specific weight to a given element block. The format to give block *n* a weight of *m* is

eb=n:m

If the sub-option *eb* comes after the *read* sub-option in the weighting specifications, then the value for the element block overwrites the values read in from the EXODUS II file. If the *eb* sub-option comes before the *read* sub-option, then it is ignored. The *edges* sub-option is used to turn on edge weights for an elemental decomposition. The default is that edge weights are not calculated. This sub-option is ignored for a nodal decomposition.

INPUT FILE FORMAT

The optional ASCII input file closely mimics the command line options. The file consists of a sequence of keys, each with a tab or "=" separated value. The order of the keys is not significant and a line beginning with a "#" is considered to be a comment. The following keys are recognized by *nem_slice*:

Input ExodusII File
Output NemesisI File
Graph Type
Decomposition Method
Solver Specifications
Machine Description
Weighting Specifications

The case of the words forming a key is not significant.

COMMAND LINE EXAMPLES

To obtain a nodal load balance for a 2D mesh parallel computer with processors arrayed in a 10x20 grid using multi-level decomposition:

```
>nem_slice -n -m mesh=10x20 -l multikl geom.exoII
```

For the same problem to be run on a 5 dimensional hypercube use:

```
>nem_slice -n -m hcube=5 -l multikl geom.exoII
```

To generate an elemental based load balance using a spectral method, with KL refinement, for a 3D mesh architecture, with the RQI eigensolver and quadrisection:

```
>nem_slice -e -m mesh=10x5x2 -l spectral,kl,num_sects=1 -s use_rqi
```

To generate a nodal based load balance with Inertial+KL with weights read from nodal results contained in an EXODUS II file use:

```
>nem_slice -n -m mesh=10x5x2 -l inertial,kl -w read=weights.exoII,vindx=1,time=2
```

ASCII INPUT FILE EXAMPLE

(See the file "input-ldbl" distributed with the executable)

SEE ALSO

The Chaco User's Guide Version 1.0; B. Hendrickson, R. Leland; Sandia Report SAND93-2339, Nov. 1993

NemesisI: A Set of Functions for Describing Unstructured Finite Element Data on Parallel Computers; G. Hennigan, J. Shadid

NOTES

nem_slice attempts to be smart about parsing command line, or ASCII input file, options. However, some errors may sneak through, in which case Chaco is relied upon for error detection.

AUTHORS

Gary L. Hennigan, Sandia National Labs, Dept. 9221

Matthew M. St. John, Sandia National Labs, Dept. 9221

John N. Shadid, Sandia National Labs, Dept. 9221

Input File Description

```
#=====================================================================
# ------------------------
# | CVS File Information |
# ------------------------
#
# $RCSfile: input-ldbl,v $
#
# $Author: rwstotz $
#
# $Date: 1999/09/30 13:05:20 $
#
# $Revision: 1.14 $
#
# $Name:  $
#=====================================================================
##############################################################################
# Lines beginning with a "#" are considered a comment.
#
# Blank lines are ignored.
#
# This file is set up as a number of lines, each of which consists of
# a key phrase, followed by a value.
#
# The order of the key phrases is not significant
#
# Case is not significant, unless it is significant in the value of a
# variable (such as file names).
#
# Order of the suboptions is not significant.
##############################################################################

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# INPUT EXODUSII FILE = <filename>
#
# This line contains the name of the input ExodusII mesh file which is
# to be load balanced
#-----------------------------------------------------------------------------
INPUT EXODUSII FILE       = testa.exoII

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# OUTPUT NEMESISI FILE = <filename>
#
# This line is OPTIONAL.
#
# This line contains the name of the output NemsisI. This filename must
# be different than the input filename.
#
# Default: nem_slice generates a name
#-----------------------------------------------------------------------------
OUTPUT NEMESISI FILE      = testa.nemI

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# OUTPUT VISUALIZATION FILE = <option>
#
# This line is OPTIONAL.
#
# Specify if a visualization file should be generated for this decomposition.
# The options for this line are "yes", "true", "false", and "no".
#
# Default: no visualization file
#----------------------------------------------------------------------------
OUTPUT VISUALIZATION FILE = true

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# GRAPH TYPE FILE = [ELEMENTAL | NODAL]
#
# This line determines what kind of a decomposition is going to be
# generated. The options are ELEMENTAL and NODAL.
#----------------------------------------------------------------------------
GRAPH TYPE                 = ELEMENTAL

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# DECOMPOSITION METHOD = <options>
#
# This line contains all of the information about the decomposition
# method that will be used. The options are:
#     multikl                 - use multilevel Kernighan-Lin partitioning
#                               method
#     spectral                - use spectral partitioning method
#     scattered               - use scattered partitioning method
#     linear                  - use linear partitioning method
#     inertial                - use inertial partitioning method
#     random                  - use random partitioning method
#     infile=<filename>       - read assignment vector from file
# NOTE: one and only one of the above options must be specified
#
#     kl                      - OPTIONAL, use Kernighan-Lin refinement for
#                                 multilevel methods; Default = none
#     none                    - OPTIONAL, do not use any refinement;
#                               Default = none
#     num_sects=<integer>     - OPTIONAL, number of eigenvectors; Default = 1
#     cnctd_dom               - OPTIONAL, set CONNECTED_DOMAINS parameter
#                               in Chaco. This option forces Chaco to make
#                               sure that all domains are connected after
#                               the partitioning. Default = off
#     outfile=<filename>      - OPTIONAL, One the partition is complete,
#                               write out the assignment vector to an ascii
#                               file. The nemesisI file is still created if
#                               this option is used. NOTE: This option and
#                               the "infile" option cannot be used at the
#                               same time. Default = off
#----------------------------------------------------------------------------
DECOMPOSITION METHOD      = INERTIAL,KL, NUM_SECTS=1, CNCTD_DOM

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# SOLVER SPECIFICATIONS = <options>
```

```
#
# This line is OPTIONAL.
#
# This line contains all of the options for the eigensolver. The options are:
#     tolerance=<number>      - OPTIONAL, set the eigensolver tolerance;
#                               Default = 1.0e-3
#     use_rqi                 - OPTIONAL, use the RQI/Symmlq eigensolver;
#                               Default = OFF
#     vmax=<integer>          - OPTIONAL, set the number of vertices to
#                               coarsen down to; Default = 200
#----------------------------------------------------------------------------
SOLVER SPECIFICATIONS     = TOLERANCE=2.0e-3,USE_RQI,VMAX=200

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# MACHINE DESCRIPTION = <options>
#
# This line is used to describe the machine. The options are:
#       hcube=<integer>         - Set machine type to hypercube where the
#                                 order of the cube is given.
#     hypercube=<integer>     - same as "hcube"
#     mesh=<mesh description> - Set the machine type to a mesh. The
#                               description can be given as a 1, 2, or
#                               three dimensional mesh by giving the
#                               number of processors in each dimension
#                               seperated by an "x". For example, a 2d
#                               mesh with 10 processors on each side is
#                               designated be "10x10"
#     cluster=<cluster desc>  - Set the machine type to a Cluster machine.
#                               This causes nem_slice to slice first on
#                               the number of boxes, and then call Chaco
#                               for each box. The hope is that this will
#                               minimize the communication accross the boxes.
#                               The machine description is as follows:
#                                     <# boxes><m|h><description>
#                                     # boxes     - number of boxes, integer
#                                     m | h       - designates each boxe as
#                                                   either mesh or hypercube
#                                     description - the description of the
#                                                   mesh or hypercube same
#                                                   as above.
#                                     example:
#                                     cluster=16m8x8
#                                     16 boxes, and each box has a 2d mesh
#                                     that is 8 by 8 processors.
#----------------------------------------------------------------------------
MACHINE DESCRIPTION             = MESH=2x2

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# WEIGHTING SPECIFICATIONS = <options>
#
# This line is OPTIONAL.
#
# This line contains any weighting information. If this line is absent,
# then all weighting is turned off. The options are:
```

```
#     read=<filename>         - OPTIOINAL, This is the name of an ExodusII
#                               file from which to read the vertex weights.
#                               If this is specified, then either var_name
#                               or var_index must be specified.
#     var_name=<var name>     - This is the name of the variable in the
#                               file, read, which will be used as vertex
#                               weights.
#     var_index=<integer>     - This is the index of the variable in the
#                               file, read, which will be used as vertex
#                               weights.
#     time_index=<integer>    - OPTIONAL, This is the time index of the
#                               variable to be read. If it is not given
#                               then the default is time index 1.
#     eb=<block id>:<weight>  - OPTIONAL, This is a way to weight a specific
#                               element block. Weights should be specified
#                               with integers. So, in order to assign element
#                               block id 10 with a weight of 6 the following
#                               would be used: eb=10:6. If an element block
#                               weight is place after a read on the weighting
#                               specification line, then it will overwrite
#                               the vertex weights from the read.
#     edges                   - turn edge weights on
#-----------------------------------------------------------------------------
WEIGHTING SPECIFICATIONS  = READ=testa-out-r.exoII,Var_Name=Temp,time_index=5

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# MISC OPTIONS = <options>
#
# This line is OPTIONAL.
#
# This line is used to handle any miscellaneous information that doesn’t
# really fit anywhere else. The options are:
#     checks_off              - OPTIONAL, This turns off some of the
#                               error checking that nem_slice performs
#                               when it is finding an elements side ids.
#                               NOTE: this option should be used with
#                               caution.
#     face_adj                - OPTIONAL, In an elemental distribution,
#                               use a face concept of adjacency to generate
#                               the graph that will be passed to Chaco.
#                               This definition only allows adjacency
#                               between elements if they share an entire
#                               face. The default is for elements to be
#                               adjacent if the share at least one node.
#     groups {group designator} - Define groups that will be passed into
#                               Chaco seperately. The groups are designated
#                               by element block ids. This option should be
#                               used when there is a problem handling
#                               elements from different blocks on the same
#                               processor. The rules for the group designator
#                               are as follows:
#                               - The beginning and end of the group
#                                 designator are set using brackets "{","}"
#                               - Blocks are grouped using the slash "/"
#                                 character
#                               - Ids are separated with white space, comma,
#                                 or by the hyphen "-" character
#                               - Any blocks not included in the list, are
#                                 added to a separate group
#                               - Duplicates in the list are permitted, but
#                                 the last group to which a block is placed
#                                 is where the block will go
#                               - Block IDs not in the exodus file are
#                                 quietly ignored
#                               Example:
#    Assume block IDs= 1-20 31-45
#
#    descriptor               group1          group2     group3
#  - {1-20}                   1-20             31-45
#  - {30-45 3/ 10-12}         3, 30-45         10,11,12   1,2,4-20
#  - {1-20/40-45/5-10 21-41}  1-4,11-20        42,43,45   5-10 31-41
#-----------------------------------------------------------------------------
MISC OPTIONS              = checks_off, face_adj, groups {1 / 2}
```

Appendix F: Input Files for Example Problems

Input File for Example 1

```
---------------------------------------------------
General Problem Specifications
---------------------------------------------------
Problem type                    = porous_flow
Input FEM file                  = theis2d_mesh.exoII
LB file                         =
Output FEM file                 = theis2d_out.exoII
Number of processors              = 1
Cartesian or Cylindrical when 2D = Cartesian
Interpolation order               = linear
Stabilization                     = default
Mass Lumping                      = off
Debug                             = 3

---------------------------------------------------
                 Solution Specifications
---------------------------------------------------
Solution Type                      = transient
Order of integration/continuation = 1
Step Control                       = on
Relative Time Integration Error   = 1.0e-3
Initial Parameter Value            =
Initial Step Size                  = 1.
Maximum Number of Steps            = 50
Maximum Time or Parameter Value   = 8640000.

---------------------------------------------------
                 Solver Specifications
---------------------------------------------------

Override Default Linearity Choice = nonlinear

--------------- nonlinear solver subsection: -----

Number of Newton Iterations                   = 8
Use Modified Newton Iteration                 = no
Enable backtracking for residual reduction    = no
Choice for Inexact Newton Forcing Term        = 0
Calculate the Jacobian Numerically            = no
Solution Relative Error Tolerance          = 1.0e-6
Solution Absolute Error Tolerance          = 1.0e-6

------------- linear solver subsection: ------

Solution Algorithm                = gmres
Convergence Norm                  = 0
Preconditioner                    = no_overlap_ilu
Polynomial                        = LS,3
Scaling                           = none
Orthogonalization                 = classical
Size of Krylov subspace           = 64
Maximum Linear Solve Iterations  = 1000
Linear Solver Normalized Residual Tolerance =1.0e-8

---------------------------------------------------
                 Chemistry Specifications
---------------------------------------------------
Energy  equation source terms    = off
Species equation source terms    = on
Pressure (atmospheres)           = 1.0
Thermal Diffusion                 = off
Multicomponent Transport         = mixture_avg
Chemkin file                      = chem.bin
Surface chemkin file              = surf.bin
Transport chemkin file            = tran.bin

--------------------------------------------------
                 Porous Flow Specifications
--------------------------------------------------
Binary Diffusion                 = off
CVFEM                            = off
Lobatto Quadrature               = off
```

```
---------------------------------------------------
        Material ID Specifications
---------------------------------------------------
Number of Materials              = 1

POROUS_MEDIUM        = 0   "Zone_1"
         G_VECTOR        =  0.0, 0.0, 9.81
         REF_PLANE       =  0.0  0.0  -1.0  -100.0
         PERM            =  2.1517e-11
         POROSITY        =  0.3
         MATRIX_COMPRESS =  3.2e-08
         T_INIT          = 10.0
         PG_INIT         =  0.101325E6
         WATER
           HEAD_INIT       = 200.
         ELEM_BLOCK_IDS  =  1
END Material ID Specifications

---------------------------------------------------
          Boundary Condition Specifications
---------------------------------------------------
Number of Generalized Surfaces   = 0
Number of BC                 = 2
BC = HEAD_BC DIRICHLET NS 1 INDEPENDENT
f_constant_head_bc 1
BC_DATA = 200.00
BC  = WD_BC NEUMANN SS 2 INDEPENDENT 9.528288e-1 0

---------------------------------------------------
         Initial Guess/Condition Specifications
---------------------------------------------------
#Set Initial Condition/Guess  = constant
Apply function               = f_head_init_cond
Time Index to Restart From   = 1

---------------------------------------------------
             Output Specifications
---------------------------------------------------
User Defined Output              = no
Parallel Output                  = no
Scalar Output                    = yes
Time Index to Output To          = 1
Nodal variable output times:
     every 1 steps
Number of nodal output variables  = 1
Nodal variable names:
      Water_Density
Number of global output variables = 0
Global variable names:

Test Exact Solution Flag          = 0
Name of Exact Solution Function   = f_theis_soln

--------------------------------------------------
      Auxiliary Output Specifications
--------------------------------------------------
Auxiliary output file           = theis2d_aux.exoII
Number of auxiliary nodal variables = 4

Auxiliary nodal variable and function name(s):
AUX_OUT = rho_l        FUNCTION = Rho_liq
AUX_OUT = Head         FUNCTION = Hyd_head
AUX_OUT = P_liq        FUNCTION = P_liq
AUX_OUT = Head_Theis   FUNCTION = f_theis_soln

-------------------------------------------------
    Data Specification for User's Functions
-------------------------------------------------
Number of functions to pass data to = 1

Function = f_theis_soln 2
FN_DATA = INT 0
FN_DATA = 1.0e-04  1.0e-04  1.0e-04  2.9656e-4
```

```
######## END OF SALSA INPUT FILE ############
```

Input File for Example 2

```
-------------------------------------------------
General Problem Specifications
-------------------------------------------------
Problem type              = porous_flow
Input FEM file            = vapor_extract_3d.exoII
LB file                   =
Output FEM file           =
vapor_extract_out.exoII
Number of processors            = 1
Cartesian or Cylindrical when 2D = Cartesian
Interpolation order             = linear
Stabilization                   = default
Mass Lumping                    = off
Debug                           = 8

-------------------------------------------------
        Solution Specifications
-------------------------------------------------
Solution Type                    = steady
Order of integration/continuation = 1
Step Control                     = on
Relative Time Integration Error   = 1.0e-3
Initial Parameter Value          =
Initial Step Size                = 1.
Maximum Number of Steps          = 200
Maximum Time or Parameter Value   = 21600.

--------------------------------------------------
                 Solver Specifications
--------------------------------------------------

Override Default Linearity Choice = nonlinear

------------ nonlinear solver subsection: ---------

Number of Newton Iterations                  = 8
Use Modified Newton Iteration                = no
Enable backtracking for residual reduction    = no
Choice for Inexact Newton Forcing Term        = 0
Calculate the Jacobian Numerically           = no
Solution Relative Error Tolerance            =
1.0e-6
Solution Absolute Error Tolerance            =
1.0e-6

------------ linear solver subsection: ------------

Solution Algorithm              = gmres
Convergence Norm                = 3
Preconditioner                  = no_overlap_ilu
Polynomial                      = LS,3
Scaling                         = none
Orthogonalization               = classical
Size of Krylov subspace         = 64
Maximum Linear Solve Iterations  = 1000
Linear Solver Normalized Residual Tolerance =1.0e-7

-------------------------------------------------
                 Chemistry Specifications
-------------------------------------------------
Energy  equation source terms    = off
Species equation source terms    = on
Pressure (atmospheres)           = 1.0
Thermal Diffusion                = off
Multicomponent Transport         = mixture_avg
Chemkin file                     = chem.bin
Surface chemkin file             = surf.bin
Transport chemkin file           = tran.bin

-----------------------------------------------
          Porous Flow Specifications
-----------------------------------------------
Binary Diffusion                = off
CVFEM                           = on
Lobatto Quadrature              = on

----------------------------------------------
        Material ID Specifications
----------------------------------------------
Number of Materials              = 1

ANISOTROPIC_POROUS_MEDIUM        = 0   "Zone_1"
         G_VECTOR        =  0.0, 0.0, 0.0
         PERM            =  1.e-11  1.e-12  1.e-11
         REL_PERM_FUNC   =  VG_krel
         POROSITY        =  0.4
         MATRIX_COMPRESS =  1.e-8
         CAP_PRESS_FUNC  =  VG_cap_press
         SAT_INIT        = 0.05
         T_INIT          = 10.0
         AIR
           PA_INIT       = 0.101325e+06
         ELEM_BLOCK_IDS  =  1
END Material ID Specifications

---------------------------------------------------
          Boundary Condition Specifications
---------------------------------------------------
Number of Generalized Surfaces   = 0
Number of BC                 = 5
BC =PA_BC DIRICHLET NS 6 INDEPENDENT 0.101325e+06 0
BC =PA_BC DIRICHLET NS 5 INDEPENDENT 0.101325e+06 0
BC = PA_BC NEUMANN SS 1 INDEPENDENT 0.0e+00 0
BC = PA_BC NEUMANN SS 2 INDEPENDENT 3.97887e-02 0
BC = PA_BC NEUMANN SS 3 INDEPENDENT 0.0e+00 0

-------------------------------------------------
         Initial Guess/Condition Specifications
-------------------------------------------------
Set Initial Condition/Guess  = constant
Apply function               = no
Time Index to Restart From   = 1

-------------------------------------------------
          Output Specifications
--------------------------------------------------
User Defined Output               = no
Parallel Output                   = no
Scalar Output                     = yes
Time Index to Output To             = 1
Nodal variable output times:
        every 1 steps
Number of nodal output variables  = 1
Nodal variable names:
      air_pressure
Number of global output variables = 0
Global variable names:

Test Exact Solution Flag          = 0
Name of Exact Solution Function   = f_xx_yy

--------------------------------------------------
       Data Specification for User’s Functions
--------------------------------------------------
Number of functions to pass data to = 2

Function = VG_krel 2
FN_DATA = INT 0
FN_DATA = 3.34 1.982 0.277 0.0 1

Function = VG_cap_press 2
FN_DATA = INT 0
FN_DATA = 3.34 1.982 0.277 0.0 1

######## END OF SALSA INPUT FILE #####
```

Input File for Example 3

```
-------------------------------------------------
General Problem Specifications
-------------------------------------------------
Problem type                = porous_flow
Input FEM file              = heatpipe_2.25m.exoII
LB file                     =
Output FEM file             = heatpipe_out.exoII
Number of processors             = 1
Cartesian or Cylindrical when 2D = Cartesian
Interpolation order              = linear
Stabilization                    = default
Mass Lumping                     = off
Debug                            = 2

--------------------------------------------------
                Solution Specifications
--------------------------------------------------
Solution Type                     = transient
Order of integration/continuation = 1
Step Control                      = on
Relative Time Integration Error   = 0.075
Initial Parameter Value           =
Initial Step Size                 = 1.
Maximum Number of Steps           = 500
Maximum Time or Parameter Value   = 6.048e6

---------------------------------------------------
                Solver Specifications
---------------------------------------------------

Override Default Linearity Choice = nonlinear

---------------- nonlinear solver subsection: -----

Number of Newton Iterations                   = 15
Use Modified Newton Iteration                 = no
Enable backtracking for residual reduction    = no
Choice for Inexact Newton Forcing Term        = 0
Calculate the Jacobian Numerically            = no
Solution Relative Error Tolerance          = 1.0e-4
Solution Absolute Error Tolerance          = 1.0e-5

---------------- linear solver subsection: --------
-

Solution Algorithm               = gmres
Convergence Norm                 = 0
Preconditioner                   = no_overlap_ilu
Polynomial                       = LS,3
Scaling                          = row_sum
Orthogonalization                = classical
Size of Krylov subspace          = 64
Maximum Linear Solve Iterations  = 500
Linear Solver Normalized Residual Tolerance =1.0e-4
----------------------------------------------
                Chemistry Specifications
----------------------------------------------
Energy  equation source terms    = off
Species equation source terms    = off
Pressure (atmospheres)           = 1.0
Thermal Diffusion                = off
Multicomponent Transport         = mixture_avg
Chemkin file                     = chem.bin
Surface chemkin file             = surf.bin
Transport chemkin file           = tran.bin
----------------------------------------------
                Porous Flow Specifications
----------------------------------------------
Binary Diffusion                = on
CVFEM                           = on

----------------------------------------------
        Material ID Specifications
----------------------------------------------
Number of Materials     = 1

POROUS_MEDIUM           = 0   "porous-column"
G_VECTOR        =  0.0, 0.0, 0.0
         PERM            =  1.0e-12
         REL_PERM_FUNC   =  cubic_krel
```

```
         POROSITY        =  0.40
         MATRIX_COMPRESS =  8.e-8
CAP_PRESS_FUNC  =  cubic_cap_press
         WATER
           SAT_INIT      =  0.5
         AIR
           PG_INIT       =   0.10133E6
ENERGY
           T_INIT          = 70.0
           THERMAL_CONDUCT = VARIABLE_PROP
           CP              = 700.0
           DENSITY         = 2600.0
ELEM_BLOCK_IDS  =  1
END Material ID Specifications
--------------------------------------------------
          Boundary Condition Specifications
--------------------------------------------------
Number of Generalized Surfaces   = 0
Number of BC                  = 4
BC   = WD_BC DIRICHLET NS 10 INDEPENDENT 969.625  0
BC   = PA_BC DIRICHLET NS 10 INDEPENDENT 7.021E4  0
BC   =  T_BC DIRICHLET NS 10 INDEPENDENT 70.0     0
BC   =  T_BC NEUMANN   SS 20 INDEPENDENT -100.0   0

-------------------------------------------------
         Initial Guess/Condition Specifications
-------------------------------------------------
Set Initial Condition/Guess  = constant
Apply function               = no
Time Index to Restart From   = 1
--------------------------------------------------
           Output Specifications
--------------------------------------------------
User Defined Output               = no
Parallel Output                   = no
Scalar Output                     = yes
Time Index to Output To             = 1
Nodal variable output times:
           every 864000. seconds
#every 140 steps
Number of nodal output variables  = 3
Nodal variable names:
      Water_Density
      Air_Pressure
      Temperature
Number of global output variables = 0
Global variable names:
Test Exact Solution Flag          = 0
Name of Exact Solution Function   = f_xx_yy

--------------------------------------------------
          Auxiliary Output Specifications
--------------------------------------------------
Auxiliary output file           =
heatpipe_aux.exoII
Number of auxiliary nodal variables = 9
Auxiliary nodal variable and function name(s):
AUX_OUT = Sl      FUNCTION = Sat_liq
AUX_OUT = Ptot    FUNCTION = P_total
AUX_OUT = Pcap    FUNCTION = P_cap
AUX_OUT = Ywl     FUNCTION = Mass_frac_water_in_liq
AUX_OUT = Ywg     FUNCTION = Mass_frac_water_in_gas
AUX_OUT = rhol    FUNCTION = f_auxout_rhol
AUX_OUT = rhog    FUNCTION = f_auxout_rhog
AUX_OUT = Vlx     FUNCTION = Liq_DarcyFlux_x
AUX_OUT = Vgx     FUNCTION = Gas_DarcyFlux_x

---------------------------------------------------
          Data Specification for User's Functions
---------------------------------------------------
Number of functions to pass data to = 2

Function = cubic_krel 2
FN_DATA = INT 0
FN_DATA = 0.15 0.0

Function = cubic_cap_press 2
FN_DATA = INT 0
FN_DATA = 0.15 0.0 0.05878 1.417 -2.12 1.263
######## END OF SALSA INPUT FILE ##############
```

Input File for Example 4

```
--------------------------------------------------
General Problem Specifications
--------------------------------------------------
Problem type                = porous_flow
Input FEM file              = cyl_incl_mesh.exoII
LB file                          =
Output FEM file                  =
cyl_incl_out.exoII
Number of processors             = 1
Cartesian or Cylindrical when 2D = Cartesian
Interpolation order              = linear
Stabilization                    = default
Mass Lumping                     = off
Debug                            = 2

-------------------------------------------------
                 Solution Specifications
-------------------------------------------------
Solution Type                     = steady
Order of integration/continuation = 1
Step Control                      = on
Relative Time Integration Error   = 1.0e-3
Initial Parameter Value           =
Initial Step Size                 = 1.
Maximum Number of Steps           = 200
Maximum Time or Parameter Value   = 1.e10

--------------------------------------------------
                 Solver Specifications
--------------------------------------------------

Override Default Linearity Choice = nonlinear

------------- nonlinear solver subsection: -------

Number of Newton Iterations                  = 4
Use Modified Newton Iteration                = no
Enable backtracking for residual reduction   = no
Choice for Inexact Newton Forcing Term       = 4
Calculate the Jacobian Numerically           = no
Solution Relative Error Tolerance      = 1.0e-6
Solution Absolute Error Tolerance      = 1.0e-6

--------------- linear solver subsection: --------

Solution Algorithm               = cgstab
Convergence Norm                 = 0
Preconditioner                   = ilu
Polynomial                       = LS,3
Scaling                          = none
Orthogonalization                = classical
Size of Krylov subspace          = 64
Maximum Linear Solve Iterations  = 600
Linear Solver Normalized Residual Tolerance =1.0e-6

--------------------------------------------------
                 Chemistry Specifications
--------------------------------------------------
Energy  equation source terms    = off
Species equation source terms    = off
Pressure (atmospheres)           = 1.0
Thermal Diffusion                = off
Multicomponent Transport         = mixture_avg
Chemkin file                     = chem.bin
Surface chemkin file             = surf.bin
Transport chemkin file           = tran.bin

--------------------------------------------------
                 Porous Flow Specifications
--------------------------------------------------
Binary Diffusion                = off
CVFEM                           = off
Lobatto Quadrature              = off
-------------------------------------------------
        Material ID Specifications
------------------------------------------------
Number of Materials              = 2
POROUS_MEDIUM                   = 0   "exterior"
         G_VECTOR        =  0.0, 1e-5, 0.0
         PERM            =  1.0e-13
         REL_PERM_FUNC   =  VG_krel
         POROSITY        =  0.3
         MATRIX_COMPRESS =  1.e-8
         CAP_PRESS_FUNC  =  VG_cap_press
         PG_INIT         =  0.101325E6
         T_INIT          =  20.
         WATER
           WD_INIT       =  996.1
         ELEM_BLOCK_IDS  =  8,12,14,18
END Material ID Specifications

POROUS_MEDIUM                   = 1   "inclusion"
         PERM            =  1.0e-12
         REL_PERM_FUNC   =  VG_krel
         POROSITY        =  0.3
         MATRIX_COMPRESS =  1.e-8
         CAP_PRESS_FUNC  =  VG_cap_press
         PG_INIT         =  0.101325E6
         T_INIT          =  20.
         WATER
           WD_INIT       =  996.1
         ELEM_BLOCK_IDS  =  10,16
END Material ID Specifications
-----------------------------------------------
          Boundary Condition Specifications
------------------------------------------------
Number of Generalized Surfaces   = 0
Number of BC          = 2
BC    =  WD_BC DIRICHLET NS 1 INDEPENDENT 996.1 0
BC    =  WD_BC DIRICHLET NS 2 INDEPENDENT 996.0 0

---------------------------------------------------
         Initial Guess/Condition Specifications
---------------------------------------------------
Set Initial Condition/Guess  = constant
Apply function               = no
Time Index to Restart From   = 1

---------------------------------------------------
                         Output Specifications
---------------------------------------------------
User Defined Output               = yes
Parallel Output                   = no
Scalar Output                     = yes
Time Index to Output To             = 1
Nodal variable output times:
   every 1 steps
Number of nodal output variables  = 1
Nodal variable names:
      Water_Density
Number of global output variables = 0
Global variable names:

Test Exact Solution Flag          = 0
Name of Exact Solution Function   = f_xx_yy

---------------------------------------------------
            Auxiliary Output Specifications
---------------------------------------------------
Auxiliary output file    = cyl_incl_aux.exoII
Number of auxiliary nodal variables = 5
Auxiliary nodal variable and function name(s):
AUX_OUT = Pliq      FUNCTION = P_liq
AUX_OUT = Head      FUNCTION = Hyd_head
AUX_OUT = Dflx_x    FUNCTION = Liq_DarcyFlux_x
AUX_OUT = Dflx_y    FUNCTION = Liq_DarcyFlux_y
AUX_OUT = Dflx_z    FUNCTION = Liq_DarcyFlux_z

----------------------------------------------
    Data Specification for User's Functions
----------------------------------------------
Number of functions to pass data to = 5

Function = VG_krel 2
FN_DATA = INT 0
FN_DATA = 3.34 1.982 0.277 0.0 0
```

```
Function = VG_cap_press 2
FN_DATA = INT 0
FN_DATA = 3.34 1.982 0.277 0.0 0

Function = VG_krel 2
FN_DATA = INT 1
FN_DATA = 3.34 1.982 0.277 0.0 0

Function = VG_cap_press 2
FN_DATA = INT 1
FN_DATA = 3.34 1.982 0.277 0.0 0

Function = f_ss_flux 2
FN_DATA = INT 1 2 3 4 5 6 7
FN_DATA = STRING Liq

###### END OF SALSA INPUT FILE #####
```

Input File for Example 5

```
-------------------------------------------------
General Problem Specifications
--------------------------------------------------
Problem type            = mass_conv_diff
Input FEM file          = test_new_mesh.exoII
LB file                    =
Output FEM file            = test_new.exoII
Number of processors      = 1
Cartesian or Cylindrical when 2D = Cartesian
Interpolation order           = linear
Stabilization                     = default
Debug                               = 3

---------------------------------------------
                Solution Specifications
---------------------------------------------
Solution Type                    = transient
Order of integration/continuation = 2
Step Control                     = on
Relative Time Integration Error   = 1.0e-3
Initial Parameter Value               =
Initial Step Size                  = 0.5
Maximum Number of Steps         = 1000
Maximum Time or Parameter Value   = 61.0e+0
--------------------------------------------
             Solver Specifications
--------------------------------------------
Override Default Linearity Choice = nonlinear
----------- nonlinear solver subsection: ---
Number of Newton Iterations               = 20
Use Modified Newton Iteration            = no
Enable backtracking for residual reduction  = no
Choice for Inexact Newton Forcing Term      = 0
Calculate the Jacobian Numerically        = no
Solution Relative Error Tolerance       = 1.0e-5
Solution Absolute Error Tolerance       = 1.0e-5
------------ linear solver subsection: --------
Solution Algorithm                  = gmres
Convergence Norm                    = 0
Preconditioner                  = no_overlap_ilu
Polynomial                           = LS,3
Scaling                             = row_sum
Orthogonalization                   = classical
Size of Krylov subspace             = 128
Maximum Linear Solve Iterations     = 400
Linear Solver Normalized Residual Tolerance =1.0e-4
---------------------------------------------------
                Chemistry Specifications
---------------------------------------------------
Energy  equation source terms    = off
Species equation source terms   = off
Pressure (atmospheres)    = 1
Thermal Diffusion                 = off
Multicomponent Transport          = mixture_avg
Chemkin file                      = chem.bin
Surface chemkin file              = surf.bin
Transport chemkin file            = tran.bin
--------------------------------------------------
        Auxiliary Input Specifications
--------------------------------------------------
Auxiliary input file          =
Auxiliary variable time index = 1
Number of auxiliary nodal variables = 0
Auxiliary nodal variable name(s):
---------------------------------------------------
        Material ID Specifications
-------------------------------------------------
Number of Materials                = 1
POROUS_MEDIUM              = 0   "Material A"
         DENSITY             =  1.0
         U_INIT                  =  0.5
         V_INIT                  =  0.0
         MOIST_CONT    = 1.0
         DISP_TRAN        = 0.01
         DISP_LONG        = 1.0
         NUM_SPECIES     = 1
         SPECIES_NAME  1       C
         DIFF_COEFF       C       1.0e-12
         WTSPECIES          C       1.0
         ELEM_BLOCK_IDS  =  1
END Material ID Specifications
---------------------------------------------------
          Boundary Condition Specifications
---------------------------------------------------
Number of Generalized Surfaces   = 0
Number of BC                       = 1
BC   = Y_BC DIRICHLET NS 10 INDEPENDENT  0.0
       SPECIES_LIST = 1
-------------------------------------------------
        Initial Guess/Condition Specifications
---------------------------------------------------
Set Initial Condition/Guess  = constant 0.0
Apply function            = user_init_cond
Time Index to Restart From   = 1
Nodal variable output times:
every 10 units
Number of nodal output variables  = 1
Nodal variable names:
      Mass_Fraction
Number of global output variables = 0
Global variable names:
Test Exact Solution Flag          = 1
Name of Exact Solution Function   = f_pt_src
----------------------------------------------------
               Output Specifications
----------------------------------------------------
User Defined Output                = no
Parallel Output                    = no
Scalar Output                      = yes
Time Index to Output To  = 1
Nodal variable output times:
every 10 units
Number of nodal output variables  = 1
Nodal variable names:
      Mass_Fraction
Number of global output variables = 0
Global variable names:

Test Exact Solution Flag          = 1
Name of Exact Solution Function   = f_pt_src

######## END OF  INPUT FILE ################
```

DISTRIBUTION

EXTERNAL DISTRIBUTION:

2 Commissariat a l'Energie Atomique Saclay
Attn: Dr. P. Maugis
SEMT/MTMS/99-090
CEA/SACLAY
91191 Gif sur Yvette Cedex
FRANCE

Prof. M. Kaviany
Dept. of Mech. Eng.
University of Michigan
Ann Arbor, MI 48109

Prof. Mary F. Wheeler
Center for Subsurface Modeling
Univ. of Texas, Austin
ACES Bldg. Rm 5.324
Austin, TX 78712

Prof. K. Vafai
Dept. of Mech. Eng.
Univ. of Calif., Riverside
Riverside, CA 92521

Prof. Y. C. Yortsos
Dept. of Chemical Eng.
Univ. of Southern California
Los Angeles, CA 90089

Dr. L. D. Stewart
Praxis Environmental Technologies, Inc.
1440 Rollins Road
Burlingame, CA 94010

2 University of California, Berkeley
Attn: K. S. Udell, A. C. Fernandez-Pello
Department of Mechanical Engineering
Berkeley, CA 94720

U.S. Army Engineers Waterways Exp. Stat.
Attn: J. P. Holland
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

INTERNAL DISTRIBUTION:

1	0151	T. O. Hunter, 9000
1	0841	Thomas Bickel, 9100
1	0824	Arthur Ratzel, 9110
1	0826	Wahid Hermina, 9113
1	0836	Justine Johannes, 9114
1	0836	Rick Givler, 9114
10	0836	Polly Hopkins, 9114
10	0836	Mario Martinez, 9114
1	0836	Harry Mofffat, 9114
1	0836	R. Allan Roach, 9114
1	0825	Walter Rutledge, 9115
1	0836	Eugene Hertel, 9116
1	0836	Richard Griffith, 9117
1	0836	Charles Hickox, 9117
1	0847	Harold Morgan, 9120
1	0836	Michael McGlaun, 9140
1	0835	Steven Kempka, 9141
1	0835	James Peery, 9142
1	0847	C. Mike Stone, 9142
1	0827	John Zepper, 9143
1	0321	William Camp, 9200
1	1110	David Womble, 9214
1	0316	Sudip Dosanjh, 9233
1	0316	Gary Hennigan, 9233
1	1111	Andrew Salinger, 9233
1	1111	John Shadid, 9233

1	0188	D. L. Chavez, 1030
1	1076	M. J. Rightley, 1745
1	0892	D. R. Adkins, 1764
1	0701	Wendy Cieslak, 6100
1	0701	Peter Davies, 6100
1	0735	E. K. Webb, 6115
1	0735	Mehdi Eliassi, 6115
1	0735	Sean McKenna, 6115
1	0735	C. K. Ho, 6115
1	0735	V. C. Tidwell, 6115
1	0750	H. R. Westrich, 6118
1	0703	J. B. Moreno, 6216
1	0736	Thomas Blejwas, 6400
1	1138	P. C. Reeves, 6533
1	0771	Margaret Chu, 6800
1	0720	Darryl Drayer, 6804
1	1395	M. Kathryn Knowles, 6821
1	0779	Merton Fewell, 6821
1	1395	Teklu Hadgu, 6821
1	0779	Alex Treadway, 6821
1	1395	Francis Hansen, 6822
1	1399	Stanley Orrell, 6850
1	0778	Peter Swift, 6851
1	0778	John Gauthier, 6851
1	0778	Michael Wilson, 6851
1	0776	Hong-Nian Jow, 6852
1	0776	Bill W. Arnold, 6852
1	0776	James Ramsey, 6852
	9018	Central Technical Files, 8945-1
2	0899	Technical Library, 9616
	612	Review and Approval Desk, 9612 For DOE/OSTI