LBNL-6871E

USER'S MANUAL OF THE TOUGH+ CORE CODE v1.5: A GENERAL-PURPOSE SIMULATOR OF NON-ISOTHERMAL FLOW AND TRANSPORT THROUGH POROUS AND FRACTURED MEDIA

G.J. Moridis and K. Pruess

Earth Sciences Division,
Lawrence Berkeley National Laboratory,
Berkeley, CA 94720

August 2014

This work was supported by the Assistant Secretary for Fossil Energy, Office of Natural Gas and Petroleum Technology, through the National Energy Technology Laboratory, under the U.S. Department of Energy, Contract No DE-AC02-05CH11231.

PAGE LEFT INTENTIONALLY BLANK

User's Manual of the TOUGH+ Core Code v1.5: A General-Purpose Simulator of Non-Isothermal Flow And Transport Through Porous And Fractured Media

G.J. Moridis and K. Pruess
Earth Sciences Division, Lawrence Berkeley National Laboratory
University of California, Berkeley, California

Abstract

TOUGH+ v1.5 is a numerical code for the simulation of multi-phase, multi-component flow and transport of mass and heat through porous and fractured media, and represents the third update of the code since its first release [*Moridis et al*., 2008]. TOUGH+ is a successor to the TOUGH2 [*Pruess et al.*, 1991; 2012] family of codes for multi-component, multiphase fluid and heat flow developed at the Lawrence Berkeley National Laboratory. It is written in standard FORTRAN 95/2003, and can be run on any computational platform (workstations, PC, Macintosh).

TOUGH+ v1.5 employs dynamic memory allocation, thus minimizing storage requirements. It has a completely modular structure, follows the tenets of Object-Oriented Programming (OOP), and involves the advanced features of FORTRAN 95/2003, i.e., modules, derived data types, the use of pointers, lists and trees, data encapsulation, defined operators and assignments, operator extension and overloading, use of generic procedures, and maximum use of the powerful intrinsic vector and matrix processing operations.

This report presents the core TOUGH+ v1.5 code, i.e., the part of the code that is common to all its applications. It provides a description of the underlying physics and thermodynamics of non-isothermal flow, of the mathematical and numerical approaches, as well as a detailed explanation of the general (common to all applications) input requirements, options, capabilities and output specifications. The core code cannot run by itself: it needs to be coupled with the code for the specific TOUGH+ application option that describes a particular type of problem. The additional input requirements specific to a particular TOUGH+ application options and related illustrative examples can be found in the corresponding User's Manual.

PAGE LEFT INTENTIONALLY BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PAGE LEFT INTENTIONALLY BLANK

1.0. Introduction

1.1. Background

TOUGH+ v1.5 is a numerical code for the simulation of multi-phase, multi-component flow and transport of mass and heat through porous and fractured media, and represents the third update of the code since its first release in a version focusing on the analysis of system behavior of hydrate-bearing sediments [*Moridis et al*., 2008]. TOUGH+ is a successor to the TOUGH2 [*Pruess et al*., 1999; 2012] family of codes for multi-component, multiphase fluid and heat flow developed at the Lawrence Berkeley National Laboratory (LBNL). It is written in standard FORTRAN 95/2003, and can be run on any computational platform (workstations, PC, Macintosh).

TOUGH+ v1.5 has a completely modular architecture. Any member of the TOUGH+ family of codes comprises three components: (a) the core TOUGH+ code that is common to all applications related to the study of non-isothermal processes of flow and transport through geologic media, (b) the code that is *unique* to a particular type of

application/problem (e.g., the properties and flow of a crude oil, the flow of water and air through geologic media, etc.), and (c) supplemental TOUGH+ code units that describe special physics and processes that are encountered in particular types of problems (e.g., code units that describe real gas properties, non-Darcian flow processes, salinity effects on the properties of water, etc.) and are *used by more than one* application options.

Thus, the core TOUGH+ code – which is distributed as a separate entity by LBNL– cannot conduct any simulations by itself, but needs additional units of supplemental and problem-specific code before it can become operational. The additional code solves the equation of state (EOS) corresponding to the specific problem; it is called an ***application option*** or simply an ***option*** in the TOUGH+ nomenclature and is distributed as a separate entity/product by LBNL. The term ***option*** – rather the older term ***module*** or ***EOS*** that were used in the TOUGH2 [*Pruess et al*., 1999; 2012] nomenclature – is used to avoid confusion, as the word *module* has a particular meaning in the FORTRAN 95/2003 language of TOUGH+.

For example, to solve the problem of flow of water/brine and real gas flow through a tight fractured porous medium (e.g., in a fractured shale reservoir), the TOUGH+ v1.5 code must be coupled with the REALGASBRINE v1.0 option [*Moridis and Freeman*, 2014]. A discussion of the processes that are particular to this problem, as well as a detailed explanation of the additional (over those needed by the core code) input data requirements, a description of the output options, and appropriate illustrative examples with sample input and output files are included in the User's Manual of the corresponding application option.

1.2. The TOUGH+ v1.5 Core Code

While the underlying principles, physics and thermodynamics are similar to those used in the TOUGH2 family of codes [*Pruess et al.*, 1999; 2012], the code in TOUGH+ v1.5 is a radical departure from the earlier language and architecture of TOUGH2. The FORTRAN 95/2003 language of TOUGH+ has enabled a drastically different architecture. It employs dynamic memory allocation, thus minimizing storage requirements. It follows the tenets of Object-Oriented Programming (OOP), and involves entirely new data structures that describe the objects upon which the code is based. The basic objects are defined through derived data types, and their properties and processes are described in modules and sub-modules, i.e., entities that incorporate the object attributes and parameters in addition to the procedures (corresponding to the older concepts of "functions" and "subroutines" in FORTRAN 77) that describe its behavior and processes. As discussed earlier, the TOUGH+ v1.5 code has a completely modular structure that is designed for maximum traceability and ease of expansion.

By using the capabilities of the FORTRAN95/2003 language, the new OOP architecture involves the use of pointers, lists and trees, data encapsulation, defined operators and assignments, operator extension and overloading, use of generic procedures, and maximum use of the powerful intrinsic vector and matrix processing operations (available in the extended mathematical library of FORTRAN 95/2003). This leads to increased computational efficiency, while allowing seamless applicability of the code to multi-processor parallel computing platforms. The result is a code that is transparent and compact, and frees the developer from the tedium of tracking the disparate attributes that define the objects, thus enabling a quantum jump in the complexity of problem that can be

tackled. An additional feature of the FORTRAN 95/2003 language of TOUGH+ is the near complete interoperability with C/C++, which allows the interchangeable use of procedures written in either FORTRAN 95/2003 or C/C++, makes possible the seamless coupling with external packages (such as the geomechanical commercial code FLAC3D [*Itasca*, 2002] or ROCMECH [*Kim and Morids*, 2013]) and interaction with pre- and post-processing graphical environments.

Note that TOUGH+ v1.5 still uses a large number of the inputs (and the input formats) used by the conventional TOUGH2 code to fulfill the functional requirement (part of the code design) of backward compatibility of the TOUGH+ family codes with older input data files used in TOUGH2 [*Pruess et al.,* 1999; 2012] simulations. However, more advanced input data structures and formats are introduced in this version to support and describe capabilities unavailable in earlier code versions. More powerful input data file structures will be available in planned future releases of TOUGH+.

By solving the coupled equations of mass and heat balance, TOUGH+ v1.5 – coupled with the appropriate application option – can model the non-isothermal phase behavior and flow of fluids and heat in complex geologic media. The code can simulate problems covering the range from laboratory- to field-scale. The only limitations on the size of the domain to be simulated are imposed by the underlying physics and by the limitations of the computing platform. Thus, if the volume of the domain and its subdivisions are such that (a) a representative volume can be defined and (b) the fluid flow through the porous/fractured media can be adequately described mathematically, then TOUGH+ can be used for the solution of the problem if the appropriate option is available.

This report on the core code of TOUGH+ v1.5 provides a detailed description of the underlying equations of mass and energy balance in general multi-phase, multi-component systems involving porous and fractured porous media. Flow through such media is described by using multi-phase versions of Darcy's law, with appropriate extensions to describe non-linear behavior in gas flow through low-permeability media [*Klinkenberg*, 1901]. The report also discusses the source on information for the description of the thermophysical properties of commonly used fluids, such as water, real gases, etc., and includes additional subjects that are common to a wide range flow and transport processes such as multi-phase, multi-component diffusion, the pressure and temperature dependence of porosity (and the corresponding effect on permeability), gas solubility in water, flow through fractured media, and wettability-related issues (relative permeability and capillary pressure) in deformable media, including the effect of solid phases such as precipitating salts or the formation of phases such as ice and gas hydrates.

The spatial and temporal discretization of the integrodifferential equations of mass and energy balance equations results in a set of strongly nonlinear (in general) algebraic equations that need to be satisfied in every subdivision (element) of the discretized domain. These fully implicit equations are linearized using the Newton-Raphson iteration, and the resulting Jacobian matrix equation is solved – using one of the several solutions options, including a family of preconditioned conjugate gradient solvers and a direct solver – until an acceptable convergence of the solution at each timestep is attained. The accurate solution of the mass and energy balance equations being the basic method employed in TOUGH+ v1.5 (and in all members of the TOUGH2 family of codes [*Pruess et al.*, 1999; 2012]), the timestep control is always subjugated to the requirement for an

accurate solution and is a process driven mainly by the progress of the simulation rather than by the user specifications.

In addition to the detailed presentation of the underlying physics, thermodynamics, mathematics and numerical approaches, this report provides a thorough discussion of the various inputs that are common to all TOUGH+ numerical simulation of flow and transport processes through porous media. These include the data needed for the appropriate sizing of the dynamically dimensioning arrays of TOUGH+ v1.5, identification of processes that may be included or omitted (e.g., diffusion, porosity-permeability dependence, etc.), and descriptions of the discretized simulation domain in terms of elements and connections, of the properties of the various porous media in the domain, of the computational parameters that control the mathematical approaches and the execution specifics, of the definition of the initial and boundary conditions, and of the output options and specifications. As discussed earlier, data specific to the TOUGH+ v1.5 application options are not discussed here, as they are included in the corresponding User's Manuals along with sample problems.

2.0 Concepts, Underlying Physics, and Governing Equations

2.1. Modeled Processes and Underlying Assumptions

The TOUGH+ v1.5 general-purpose simulator can be used as the basis to model all the known processes and phenomena associated with the flow and transport of fluids and heat through porous and/or fractured media, such as:

(1) The flow of gases and liquids in the geologic system

(2) The corresponding heat flow and transport

(3) The partitioning of the mass components among the possible phases

(4) Heat exchanges due to

   a. Conduction

   b. Advection/convection

   c. Radiation

   d. Chemical reactions

e. Latent heat related to phase changes (ice melting or water fusion, water evaporation or vapor condensation)

f. Gas dissolution

(5) Equilibrium or kinetic chemical reactions,

(6) The multi-phase transport of solutes and colloids, accounting for advection, molecular diffusion, and sorption

(7) The effects of solutes on the system behavior

Note that this list is not comprehensive, but only indicative of some of the most obvious applications of the TOUGH+ family of codes. A significant effort has been invested in the incorporation into the code of the most recent advances in physics thermodynamics, mathematics and numerical analysis, and in keeping the simplifying assumptions involved in the development of the underlying models of the code to a minimum. Thus, the main assumption involved in TOUGH+ is that the laws governing the flow of fluids (Darcian and non-Darcian) and heat are known and valid in the simulated domain under the conditions of the study.

In the present section we present some fundamental principles and governing equations that are universally applicable to (and present in) all problems of flow and transport investigated by the TOUGH+ family of codes. More specific equations that are applicable to the particular problems investigated by the individual members of the TOUGH+ family (each describing a particular equation of state – EOS) will be presented in detail and discussed in the User's Manual accompanying each of these application options.

2.2. The Mass and Energy Balance Equation

Following *Pruess et al.* [1999; 2012], mass and heat balance considerations in every subdomain (gridblock) into which the simulation domain is been subdivided by the integral finite difference method dictates that

$$\frac{d}{dt}\int_{V_n} M^\kappa dV = \int_{\Gamma_n} \mathbf{F}^\kappa \cdot \mathbf{n}\, d\tilde{A} + \int_{V_n} q^\kappa dV\,, \tag{2.1}$$

where:

V, V_n volume, volume of subdomain n [L^3];

M^κ mass accumulation term of component κ [kg m^{-3}];

A, Γ_n surface area, surface area of subdomain n [L^2];

$\mathbf{F}^\kappa$ Darcy flux vector of component κ [kg m^{-2}s^{-1}];

$\mathbf{n}$ inward unit normal vector;

q^κ source/sink term of component κ [kg m^{-3}s^{-1}];

t time [T].

2.3. Mass Accumulation Terms

Under equilibrium conditions, the mass accumulation terms M^κ for the mass components in equation (2.3) are given by

$$M^\kappa = \sum_{\beta \equiv 1,\ldots,N_\beta} \phi S_\beta \rho_\beta X^\kappa_\beta\,, \quad \kappa \equiv 1,\ldots,N_\kappa \tag{2.2}$$

where

ϕ porosity [*dimensionless*];

ρ_β density of phase β [kg m^{-3}];

S_β saturation of phase β [*dimensionless*];

X_β^κ mass fraction of component κ in phase β [kg/kg]

N_κ number of components κ in phase β [kg/kg]

N_β number of phases β [kg/kg]

By convention, in this document phases are denoted by capital letters in TOUGH+ v1.5, while components are denoted by lower case letters. The phases that are involved in the most common of the TOUGH+ applications will be Aqueous (A), Gaseous (G) and liquid Organic (O). Other possible phases may be Ice (I), Liquid CO_2, Hydrates (H), etc. Common components include water (w), gaseous species (g) such as air, CH_4, CO_2, etc., oil (o) and/or solutes (s).

2.4. Heat Accumulation Terms

The heat accumulation term includes contributions from the rock matrix and all the phases is given by the equation

$$M^\theta = (1-\phi)\rho_R C_R T + \sum_{\beta=1,\ldots,N_\beta} \phi S_\beta \rho_\beta U_\beta \quad , \tag{2.3}$$

where

ρ_R rock density [kg m^{-3}];

C_R heat capacity of the dry rock [J kg^{-1} K^{-1}];

U_β specific internal energy of phase β [J kg^{-1}];

The specific internal energy of the gaseous phase is a very strong function of composition, is related to the specific enthalpy of the gas phase H_G, and is given by

$$U_G = \sum_{\kappa=1,\ldots,N_\kappa} X_G^\kappa u_G^\kappa + U_{dep} \quad \left(= H_G - \frac{P}{\rho_G} \right), \tag{2.4}$$

where u_G^κ [J/kg] is the specific internal energy of component κ under the conditions of the gaseous phase, and U_{dep} is the specific internal energy departure of the gas mixture [J/kg].

The internal energy of the aqueous phase accounts for the effects of gas and solute solution, and is estimated from

$$U_A = X_A^w u_A^w + \sum_{\substack{\kappa=1,\ldots,N_\kappa \\ \kappa \neq w}} X_A^\kappa \left(u_A^\kappa + U_{sol}^\kappa \right), \tag{2.5}$$

where u_A^w and u_A^κ [J kg^{-1}] are the specific internal energies of H_2O and of all other components κ at the conditions prevailing in the aqueous phase, respectively, and U_{sol}^κ [J/kg] are the specific internal energies corresponding to the dissolution of components κ (other than H_2O). The term u_A^κ is determined from

$$u_A^\kappa = h_A^\kappa - \frac{P}{\rho_\kappa} = \int_{T_0}^{T} C_\kappa dT - \frac{P}{\rho_\kappa} \tag{2.6}$$

where T_0 is a reference temperature, h_A^κ is the specific enthalpy of component κ, and C_κ is the temperature-dependent heat capacities of component κ [J kg^{-1} K^{-1}].

The development of the equations of mass and energy balance of any other phase (e.g., liquid organic, liquid CO_2, solid ice, etc.) is entirely analogous.

2.5. Flux Terms

The mass flux $\mathbf{F}^{\kappa}$ [kg/m^2/s] of a component κ includes contributions from all ***mobile*** phases present in the system, i.e.,

$$\mathbf{F}^{\kappa} = \sum_{\beta=1,\ldots,N_{m\beta}} \mathbf{F}_{\beta}^{\kappa} \text{ , where } \mathbf{F}_{\beta}^{\kappa} = X_{\beta}^{\kappa}\, \mathbf{F}_{\beta} \text{ , } \kappa \equiv 1,\ldots,N_{\kappa} \text{ , } \tag{2.7}$$

and $N_{m\beta}$ is the number of mobile phases. $\mathbf{F}_{\beta}$ is the flux of phase [kg/m^2/s], and can be described by several equations, the most common of which is Darcy's law:

$$\mathbf{F}_{\beta} = -k \frac{k_{r\beta}\rho_{\beta}}{\mu_{\beta}} \left(\nabla P_{\beta} - \rho_{\beta} \mathbf{g} \right), \tag{2.8}$$

where

k rock intrinsic permeability [m^2];

$k_{r\beta}$ relative permeability of the aqueous phase [dimensionless];

μ_{β} viscosity of the aqueous phase [Pa s];

P_{β} pressure of the aqueous phase [Pa];

$\mathbf{g}$ gravitational acceleration vector [m s^{-2}].

Note that Darcy's law is applicable to flows in which the Reynolds number $N_R < 1$, i.e., when the fluid flow is laminar, and when there are negligible flow slippage effects. This covers the overwhelming majority of problems of flow and transport through geologic porous media. Turbulent flows and slippage effects (occurring in low-permeability media) require other equations, which are discussed in detail in the User's Manuals of the specific TOUGH+ options describing such problems.

The Darcy's law in Equation (2.8) indicates a linear relationship between flux and pressure differential between two points in space. An important point is that Equation (2.8) involves the phase pressures, which (and their interrelationships) need to be properly defined. In general, the relationship between the phases of any two phases is described by the general equation

$$P_{w\beta} = P_{n\beta} + P_{cnw}, \tag{2.9}$$

where $P_{w\beta}$ is the pressure of the wetting phase, $P_{n\beta}$ is the pressure of the non-wetting phase, and P_{cnw} is the capillary pressure between the two phases (by convention a negative number in TOUGH+ v1.5). The gas phase is always a non-wetting phase, but the wetting behavior of all other phases depends on the surface chemistry of the porous medium, i.e., its affinity for a given phase. Usually, unconsolidated media and reservoir rocks are water-wet, i.e., the aqueous phase preferentially coats the grains surfaces, to which it is attached very strongly as indicated by a very large contact angle. Some petroleum reservoir rocks (especially older ones, with long contact of the rock grains with an organic phase) are oil-wet, and there are also rocks of mixed wettability.

For example, in a two-phase (aqueous and gas) system, the aqueous pressure P_A is given by

$$P_A = P_G + P_{cGW}, \text{ where } P_G = \sum_{\kappa=1,\ldots,N_\kappa} P_G^\kappa . \tag{2.10}$$

Here, P_G is the gas pressure [Pa], P_{cGW} is the capillary pressure [Pa] between the two phases, and P_G^κ are the partial pressures [Pa] of the various gaseous component κ in the gas phase, respectively.

The solubility of a gaseous component κ in the aqueous phase is related to P_G^κ through Henry's law,

$$P_G^\kappa = H^\kappa X_A^\kappa, \tag{2.11}$$

where $H^\kappa = H^\kappa(T, m_A^s)$ [Pa] is the temperature- and salt concentration-dependent Henry's factor, and m_A^s is the molality of the dissolved salts in the aqueous phase. TOUGH+ incudes a library of such H^κ functions that describe the water solubility of the 11 gases in its internal gas property database.

The performance of these Henry's factors has been determined to be very satisfactory over a wide spectrum of applications that cover an extended temperature and salinity range. However, more demanding problems involving high pressures, gas mixtures and the presence of more complex electrolytes cannot be adequately represented by the simple H^κ factors described above. In these cases, TOUGH+ provides an option of a more accurate (but more computationally demanding) estimation of the gas solubility in water from the equality of fugacities in the aqueous and the gas phase. Such an option is available only in specific TOUGH+ options.

In TOUGH+, the solubility of various components (ionic and non-ionic, such as electrolytes and organic substances, respectively) in the aqueous phase is determined by appropriate species-specific equations of solubility as functions of temperature and pressure. Similarly, the solubility of various components into non-aqueous phases (e.g., an organic liquid phase in petroleum reservoirs, or a liquid CO_2 phase) is described by species- and phase-specific equations. With the exception of the H^κ-based solubility of gases in the aqueous phase (from Equation (2.11), which is available as a standard in the

TOUGH+ options that require it), the solubilities of substances in phases are discussed in detail in the User's Manuals of the corresponding application options.

The mass flux of the gaseous phase ($\beta \equiv G$) incorporates advection and diffusion contributions, and is given by

$$\mathbf{F}_G^\kappa = -k_G \frac{k_{rG}\rho_G}{\mu_G} X_G^\kappa \left(\nabla P_G - \rho_G \mathbf{g}\right) + \mathbf{J}_G^\kappa, \quad \kappa \equiv 1,\ldots,N_\kappa \tag{2.12}$$

where

$$k_G = k_0\left(1+\frac{b}{P}\right); \tag{2.13}$$

k_G medium permeability to gas [m^2];

k_0 absolute permeability at large gas pressures or in liquid flow [m^2];

b *Klinkenberg* [1941] *b*-factor accounting for gas slippage effects [Pa];

k_{rG} relative permeability of the gaseous phase [*dimensionless*];

μ_G viscosity of the gaseous phase [Pa·s].

Equations (2.12) and (2.13) introduce a non-linearity in the flow equation, which is no longer Darcian and can now account for gas slippage effects. However, this non-linearity is easy to implement, is only important in low-permeability media, and is available as an option in its standard implementation (which involves a constant Klinkenberg parameter *b*) in the TOUGH+ core code. Estimates of *b* can be obtained from the tables listed in *Wu et al.* [1998] and various equation options listed in *Moridis and Freeman* [20014], e.g., the equation proposed by *Jones* [1972]:

$$\frac{b}{b_{ref}} = \left(\frac{k}{k_{ref}}\right)^{-0.36}, \tag{2.14}$$

in which the subscript *ref* denotes the known properties of a reference medium. More complex gas slippage scenarios involving other methods for computing a variable *b* are available in specific TOUGH+ options, and they are discussed in detail in the corresponding User's Manuals (e.g., see *Moridis and Freeman* (2014]).

The term $\mathbf{J}_G^\kappa$ is the diffusive mass flux of component κ in the gas phase [kg/m^2/s], and is described by

$$\mathbf{J}_G^\kappa = -\phi \underbrace{\tau_0 \tau_G}_{\tau_{TG}} D_G^\kappa \rho_G \nabla X_G^\kappa, \quad \kappa \equiv 1,\ldots,N_\kappa \tag{2.15}$$

where D_G^κ is the multicomponent molecular diffusion coefficient of component κ in the gas phase in the absence of a porous medium [m^2/s], τ_{TG} is the total gas tortuosity, τ_0 is the medium-related component of tortuosity and τ_G is the gas-saturation related component of tortuosity (both dimensionless). TOUGH+ includes several methods to compute the τ_0 and τ_β of a mobile phase β. If a constant value τ_0 is not provided as input, then the default is the model of *Millington and Quirk* [1961], according to which $\tau_0 = \phi^{1/3} S_\beta^{7/3}$. The various methods to compute τ_β are discussed in Section 5.1.

The diffusive mass fluxes of the various components κ in the gas phase are related through the relationship of *Bird et al.* [1960]

$$\sum_{k=1,\ldots,N_\kappa} \mathbf{J}_G^\kappa = 0, \tag{2.16}$$

which ensures that the total diffusive mass flux of the gas phase is zero with respect to the mass average velocity when summed over the gaseous components. Then the total gas phase mass flux is the product of the Darcy velocity and density of the gas phase.

Similarly, the flux of a dissolved species (e.g., salt – denoted by the *s* superscript in the following equation – dissolved in the aqueous phase) is described by

$$\mathbf{F}_A^s = X_A^s \mathbf{F}_A + \mathbf{J}_A^s, \text{ where } \mathbf{J}_A^s = -\phi S_A \underbrace{\tau_0 \tau_A}_{\tau_{TA}} D_A^s \rho_A \nabla X_A^s , \tag{2.17}$$

D_A^s is the molecular diffusion coefficient of the dissolved species *s* in water, and τ_A is the tortuosity of the aqueous phase. Note that mechanical dispersion is not accounted for in Equations (2.12) and (2.17), and is not discussed in this description of the core TOUGH+ v1.5 code, but the subject is fully addressed in the User's Manuals of TOUGH+ options in which dispersion is important.

The heat flux accounts for conduction, advection and radiative heat transfer, and is given by

$$\mathbf{F}^\theta = -\bar{k}_\theta \nabla T + f_\sigma \sigma_0 \nabla T^4 + \sum_{\beta \equiv 1,\ldots,N_{m\beta}} h_\beta \mathbf{F}_\beta , \tag{2.18}$$

where

$\bar{k}_\theta$ composite thermal conductivity of the rock-fluids ensemble [W m^{-1} K^{-1}];

h_β specific enthalpy of phase $\beta \equiv 1,\ldots,N_{m\beta}$ [J kg^{-1}];

f_σ radiance emittance factor [dimensionless];

σ_0 Stefan-Boltzmann constant [5.6687×10^{-8} J m^{-2} K^{-4}].

Similar to Equation (2.4), the specific enthalpy of the gas phase is computed as

$$H_G = \sum_{\kappa=1,\ldots,N_\kappa} X_G^\kappa h_G^\kappa + H_{dep} , \tag{2.19}$$

where h_G^κ is the specific enthalpy of component κ in the gaseous phase, and H_{dep} is the specific enthalpy departure of the gas mixture [J kg^{-1}]. The specific enthalpy of the aqueous phase is estimated from

$$H_A = X_A^w h_A^w + \sum_{\substack{\kappa=1,\ldots,N_\kappa \\ \kappa \neq w}} X_A^\kappa \left(h_A^\kappa + H_{sol}^\kappa \right), \tag{2.20}$$

where h_A^w and h_A^κ [J kg^{-1}] are the specific enthalpies of H_2O and of all other components κ at the conditions prevailing in the aqueous phase, respectively, and H_{sol}^κ [J kg^{-1}] are the specific internal energies corresponding to the dissolution of components κ (other than H_2O). The development of the equation of the specific enthalpy of any other phase (e.g., liquid organic, liquid CO_2, solid ice, etc.) is entirely analogous.

2.6. Source and Sink Terms

In sinks with specified mass production rate, the withdrawal of mass component κ is described by

$$\hat{q}^\kappa = \sum_{\beta \equiv 1,\ldots,N_{m\beta}} X_\beta^\kappa q_\beta , \quad \kappa \equiv 1,\ldots,N_\kappa \tag{2.21}$$

where q_β is the mass production rate of the *mobile* phase β [kg/m^3]. For a prescribed production rate, the phase flow rates q_β are determined internally according to different methods (e.g., the relative mobility at the element where the sink is located) available in **TOUGH+**. For source terms (well injection), the addition of a mass component κ occurs at desired (and known) rates $\hat{q}^\kappa$ ($\kappa \equiv 1,\ldots,N_\kappa$).

The corresponding heat exchange Q^{θ} associated with the addition or withdrawal of mass at any given source or sink is described by

$$Q^{\theta} = \sum_{\kappa \equiv 1,\ldots,N_{\kappa}} \left(\sum_{\beta \equiv 1,\ldots,N_{m\beta}} X_{\beta}^{\kappa} q_{\beta} h_{\beta}^{\kappa} \right) \tag{2.22}$$

where h_{β}^{κ} is the specific enthalpy of the mass component κ that is partitioned in mobile phase β [J/kg] – see Equations (2.19) and (2.20).

2.7. Thermophysical Properties

2.7.1. Water

The properties and parameters of liquid water and steam in TOUGH+ are provided by (a) fast regression equations based on data from *NIST* [2000] and (b) steam table equations from the most recent IAPWS97 formulations [*Wagner et al*., 2000; *IAPWS*, 2007]. These equations are accurate up to 2000 ^{o}C and 100 MPa, and computationally more efficient that those in the earlier versions of TOUGH+ (i.e., those in *Moridis et al*. [2008]). The viscosity and the thermal conductivity of the water substance in its various phases were computed using the recent correlations of thermal conductivity of *IAPWS* [2008] and *IAPWS* [2011a], respectively.

The enthalpy, sublimation pressure and fusion/melting pressure of ice (on the ice-vapor and ice-liquid water equilibrium lines of the water phase diagram), as well as the densities of the ice and of the liquid water in these regions, are computed from the correlations of *IAPWS* [2009; 2011b; 2012]. The thermal conductivity of ice was

computed using the heat capacity polynomial equation with the coefficients reported in *Yaws* [1999].

2.7.2. Real Gases

The properties of the gas phase are provided by one of the three cubic equations of state that are available in the supplemental code unit **`T_RealGas_Properties.f95`** (see Section 3.6) of TOUGH+ v1.5: (a) the Peng-Robinson equation [*Peng and Robinson*, 1976], (b) the Soave-Redlich-Kwong equation [*Soave*, 1972], and the standard Redlich-Kwong equation [*Redlich and Kwong*, 1949]. This package includes a database of the fundamental properties of 12 gases, and it computes their compressibility, density, fugacity, specific enthalpy and internal energy (ideal and departure) over a very wide range of pressure and temperature. Additionally, the package computes the gas viscosity and thermal conductivity using the method of *Chung et al.* [1988], and binary diffusivities from the method of *Fuller et al.* [1969] and *Riazi and Whitson* [1993]. TOUGH+ allows computation of all these properties not only in pure gases (i.e., involving a single gaseous component), but also in gas mixtures of either constant composition (in which the gas phase can be treated as a single pseudo-component of fixed composition) or of variable composition (in which case the various gaseous components are tracked individually).

This real-gas package in TOUGH+ also allows determination of gas solubility in water either by using a set of temperature-dependent Henry's coefficients, or by equating fugacities in the gas and aqueous phases through a process that involves the computation of the activity coefficients (in the aqueous phase) and of the fugacities. For most applications involving low-solubility gases (especially single-component ones) or

relatively low P and T, extensive experience with scoping calculations has indicated that the temperature-dependent Henry's coefficient H^m can provide reliable estimates of gas solubility. Henry's method is less reliable in the case of dissolution of multi-component gases in the presence of electrolytes (such as salts) of high ionic strength. In these cases, determination of gas solubility through the fugacities and activity coefficients provides the necessary accuracy of the dissolution estimates, albeit at the cost of a significant larger computational load. Only the solubility option that is appropriate for the problem at hand is activated in the various TOUGH+ application options that involve coexistence of gas and aqueous phases.

2.7.3. Other Phases and Components

The physical and thermal properties of all other phases and components that may be involved in the various TOUGH+ applications cannot be generalized. Thus, they are described by (a) appropriate computation methods that are coded in the corresponding TOUGH+ application option, and (b) by the corresponding parameter values that are provided in the TOUGH+ input file. These are described fully in the User's Manuals of the individual members of the TOUGH+ family involving such phases and components.

2.8. Porosity and Intrinsic Permeability Changes

2.8.1. Changes caused by P- and T-dependence

The effect of pressure change on the porosity of the matrix is described by three options. The first involves the standard equation

$$\phi = \phi_0 F_{PT}, \;\; F_{PT} = \exp\left[\alpha_P \Delta P + \alpha_T \Delta T\right] \approx 1 + \alpha_P \Delta P + \alpha_T \Delta T, \tag{2.23}$$

where $\Delta P = P - P_0$, $\Delta T = T - T_0$, the subscript '*0*' denotes a reference state, α_T is the thermal expansivity of the porous medium (1/K) and α_p is the pore compressibility (1/Pa), which can be either a fixed number or a function of pressure [*Moridis et al*., 2008; 2009; 2012]. The second option describes the *P*-dependence of ϕ as a polynomial function of *P*. The third option (discussed in detail in Section 2.10) describes the ϕ-dependence on *P* in unconsolidated media that gain significant mechanical strength from the presence of solid phases such as ice or hydrates (see *Moridis* [2014]). The pore compressibility α_P of such media is a function of the saturation SS of the solid phase(s). The thermal dependence of ϕ is still described by the exponential factor $\exp\left[\alpha_T \left(T - T_0\right)\right]$.

The ϕ - k relationship in the matrix is described by the general expression of *Rutqvist and Tsang* [2003] as:

$$\frac{k}{k_0} = \exp\left[\gamma\left(\frac{\phi}{\phi_0} - 1\right)\right], \tag{2.24}$$

where γ is an empirical permeability reduction factor that ranges between 5 (for soft unconsolidated media) and 29 (for lithified, highly consolidated media). Note that the equations described here are rather simple and apply to matrix ϕ and k changes when the

changes in p and T are relatively small. These equations are not applicable when large pressure and temperature changes occur in the matrix, cannot describe the creation of new (secondary) fractures and cannot describe the initiation, propagation and the characteristics of fractures as the fluid pressures, the temperatures, the fluid saturations and the stresses change. For such problems, it is advisable to use the T+M model [*Kim and Moridis*, 2013] that couples the flow and thermal processes in the various TOUGH+ options with the ROCMECH geomechanical code. This coupled model accounts for the effect of changing fluid pressures, saturations, stresses, and temperatures on the geomechanical regime and provides an accurate description of the evolution of ϕ and k over the entire spectrum of P and T covered during the simulation.

2.8.1. Changes caused by the evolution of solid phases

When solid phases are deposited in a porous medium, through chemical precipitation or freezing of pore fluids, the ability of the porous medium to transmit fluids can change profoundly. The deposition of solids in a porous medium reduces the void space available for fluids. Such reduction in porosity will give rise to a reduction in permeability as well.

There is an extensive literature, going back to the 1920s, about the manner in which permeability declines as portions of the pore space are filled by solids, and a bewildering variety of porosity-permeability correlations have been obtained from experimental and theoretical studies [*Scheidegger*, 1974, and references therein; *Morrow et al*., 1981; *Vaughan*, 1987; *Verma and Pruess*, 1988; *Phillips*, 1991; *Pape et al*., 1999; *Xu et al*., 2004]. Within the scope of the work undertaken here it is not possible to perform a thorough review of different permeability reduction models, and to evaluate

their suitability for representing permeability reduction due to formation of hydrate and/or ice. Dearth of relevant information prevents considering whether any of the models developed for solid precipitation in porous media are valid to systems involving solid phases such as ice or hydrates. Instead, this brief discussion addresses salient features of pore channels to highlight the most important effects, and then explain the rationale behind the preliminary choices made in this study.

It is obvious that permeability effects depend not just on the magnitude of porosity change, but on geometric properties of the pore channels, and on where and how solid deposition in those channels occurs. The lack of unanimity among different investigators about the correlation between porosity and permeability change reflects the great diversity of pore channel geometries and precipitation processes in porous media. The simplest models conceptualize porous media as bundles of capillary tubes, which gives rise to a simple power law dependence of permeability k on porosity ϕ,

$$\frac{k}{k_0} = F_{\phi S} = \left(\frac{\phi}{\phi_0}\right)^n \tag{2.25}$$

where the term $F_{\phi S}$ is a *permeability adjustment factor* that describes the effects of (a) the presence of solid phases other than the medium grains (such as ice, hydrate or precipitating salts), and (b) changes in porosity on permeability, and the subscript "00" denotes properties at a reference state.

The exponent n typically will be in the range from 2 to 3 [*Phillips*, 1991], giving a rather mild dependence of permeability on porosity. However, in media with inter-granular porosity, pore channels generally have a convergent-divergent geometry,

consisting of a succession of '*pore throats*' with small radius and '*pore bodies*' with large radius (**Figure 2.1**).

If solids are deposited uniformly along the pore walls, or are preferentially deposited in the throats, then even relatively minor amounts of deposition can give rise to a dramatic decrease in permeability. Such behavior has been observed in field and laboratory-scale systems, including the diagenesis of sandstones [*Pape et al*., 1999], precipitation around geothermal injection wells [*Xu et al*., 2004], and hydrothermal flows in laboratory specimen [*Morrow et al*., 1981; *Vaughan*, 1987].

In these systems, a rather modest amount of precipitate, that leaves most of the original pore space available for fluids, nonetheless caused order-of-magnitude changes in absolute permeability. Such behavior can be understood from '*tubes-in-series*' models of pore space, as shown in **Figure 2.2** [*Verma and Pruess*, 1988].

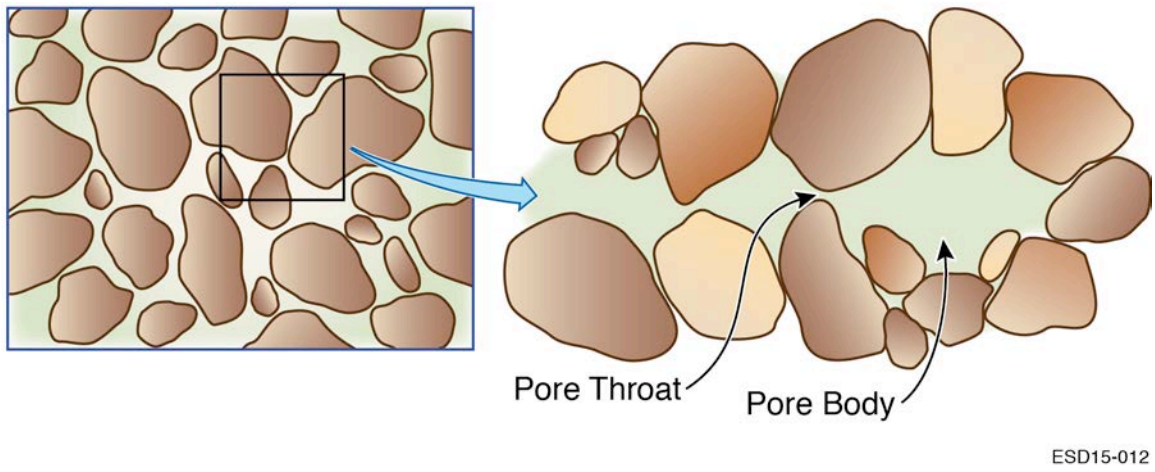

Figure 2.1. Schematic of pore channels, showing convergent-divergent geometry with a succession of pore throats and pore bodies.

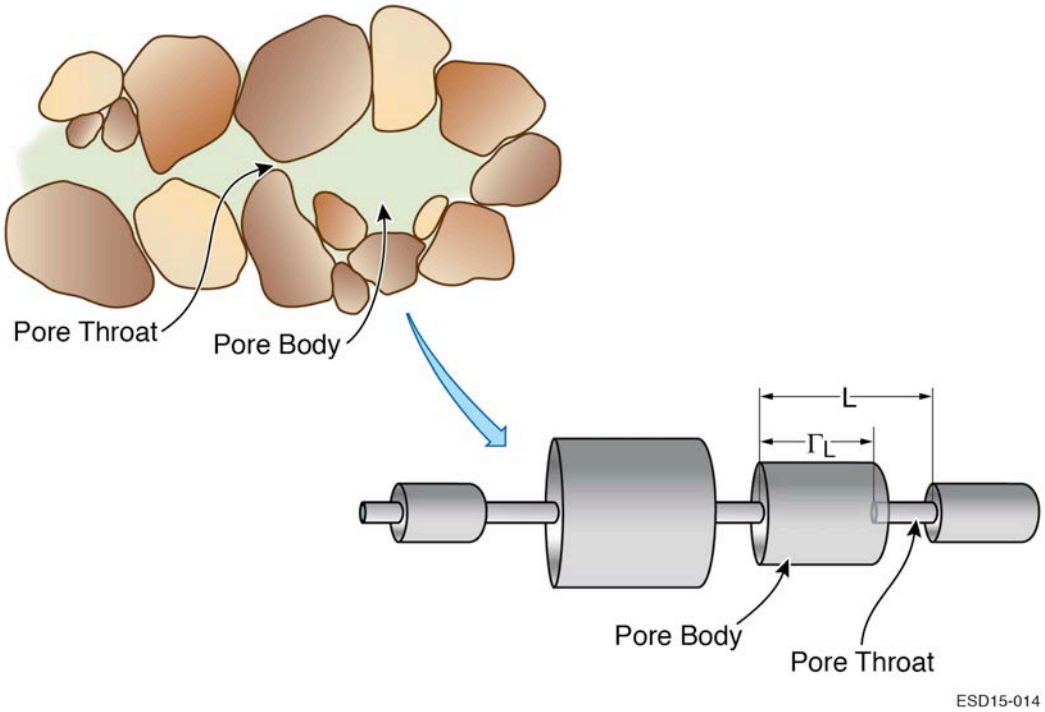

Figure 2.2. Tubes-in-series model of pore channels.

If one assumes that solids are deposited as a layer of uniform thickness on the pore walls, then permeability will be reduced to zero when the throats become clogged, while plenty of (disconnected) porosity remains in the pore bodies. This leads to the concept of a non-zero '*critical porosity*' ϕ_c at which k is reduced to zero, with a permeability reduction as given in Equation (2.25) [*Verma and Pruess*, 1988; *Xu et al*., 2004].

$$\frac{k}{k_0} = F_{\phi S} = \left(\frac{\phi - \phi_c}{\phi_0 - \phi_c} \right)^n \tag{2.26}$$

Fractal models also give a very strong dependence of permeability on porosity, with exponents in relationships such as Equation (2.25) as large as n = 10 or more [*Pape et al*., 1999]. The above discussion clearly indicates the need fundamental research for the determination of the sites within the porous media at which solid phases (such as ice and hydrates) form preferentially.

2.9. Multiphase Diffusion

2.9.1. General Considerations

Here we expand on the subject of diffusion, which has already been referred to (and included in a mathematical from) in Equations (2.12) to (2.17). The discussion here, as well as the diffusion formulation and its treatment in TOUGH+ v1.5, hews closely to those in TOUGH2 [*Pruess et al.*, 1999; 2012].

Molecular diffusion plays a minor role in many subsurface flow processes, but may become a significant and even dominant mechanism for mass transport when adjective velocities are small and/or the time frame of the simulated process is long. Diffusive flux is usually written as being proportional to the gradient in the concentration of the diffusing component (Fick's law)

$$\mathbf{J} = -d\nabla C \tag{2.27}$$

where d is an effective diffusivity, which in general will depend on properties of the diffusing component, the pore fluid, and the porous medium. The concentration variable C may be chosen in a number of different ways, e.g., mass per unit volume, moles per unit volume, mass or mole fraction, etc. [*Bird et al.*, 1960; *de Marsily*, 1986], with an appropriate selection of units for the corresponding diffusivity d.

The basic Fick's law in Equation (2.27) works well for diffusion of tracer solutes that are present at low concentrations in a single-phase aqueous solution at rest with respect to the porous medium. However, many subtleties and complications arise when multiple components diffuse in a multiphase flow system. Effective diffusivities in general may depend on all concentration variables, leading to non-linear behavior especially when

some components are present in significant (non-tracer) concentrations. Additional nonlinear effects arise from the dependence of tortuosity on phase saturations, and from coupling between advective and diffusive transport. For gases, the Fickian model has serious limitations even at low concentrations, which prompted the development of the "dusty gas" model that entails a strong coupling between advective and diffusive transport [*Mason and Malinauskas*, 1983; *Webb*, 1998], and accounts for molecular streaming effects (Knudsen diffusion) that become very important when the mean free path of gas molecules is comparable to pore sizes. Further complications arise for components that are both soluble and volatile, in which case diffusion in aqueous and gaseous phases may be strongly coupled via phase partitioning effects. An extreme case is the well-known enhancement of vapor diffusion in partially saturated media, which is attributed to pore-level phase change effects [*Cass et al*., 1984; *Webb and Ho*, 1998a, b].

2.9.2. Diffusion Formulation in TOUGH+

Because of the difficulties mentioned above, it is not possible to formulate a model for multiphase diffusion that would be accurate under all circumstances. The pragmatic approach used in Equations (2.15) and (2.17) describes the diffusive flux of component κ in mobile phase β (= liquid, gas) by the general equation

$$\mathbf{J}_\beta^\kappa = -\phi\tau_0\tau_\beta\rho_\beta D_\beta^\kappa \nabla X_\beta^\kappa , \tag{2.28}$$

where all terms are as previously defined. For ease and simplicity, it is convenient to introduce a single diffusion strength factor that combines all material constants and tortuosity factors into a single effective multiphase diffusion coefficient, as follows:

$$\mathscr{D}_\beta^\kappa = \phi \tau_0 \tau_\beta \rho_\beta D_\beta^\kappa \tag{2.29}$$

For general two-phase conditions involving an aqueous and a gas phase, the total diffusive flux is then given by

$$\mathbf{J}^\kappa = -\mathscr{D}_A^\kappa \nabla X_A^\kappa - \mathscr{D}_G^\kappa \nabla X_G^\kappa \tag{2.30}$$

The saturation dependence of tortuosity is not well known at present. For soils the *Millington and Quirk* [1961] model has frequently been used [*Jury et al*., 1983; *Falta et al*., 1989], which yields non-zero tortuosity coefficients as long as the phase saturation is non-zero. It stands to reason that diffusive flux should vanish when a phase becomes discontinuous at low saturations, suggesting that saturation-dependent tortuosity should be related to relative permeability; e.g. $\tau_\beta(S_\beta) \approx k_{r\beta}(S_\beta)$.

For components that partition between liquid and gas phases, more complex behavior may be expected. For example, consider the case of a volatile and water-soluble compound diffusing under conditions of low gas saturation where the gas phase is discontinuous. In this case we have $k_{rG}(S_G) = 0$ (because $S_G < S_{rg}$), and $k_{rA}(S_A = 1 - S_G) < 1$, so that a model equating saturation-dependent tortuosity to relative permeability would predict weaker diffusion than in single-phase liquid conditions. For compounds with significant volatility this would be unrealistic, as diffusion through isolated gas pockets would tend to enhance overall diffusion relative to single-phase liquid conditions.

2.10. Wettability-Related Phenomena

2.10.1. Relative permeability

In multiphase flow, each fluid phase occupies only part of the pore space, and its effective permeability is reduced due to interference with the other phase(s). This effect is represented by means of *permeability reduction factors* or *relative permeabilities*, customarily denoted by k_{rA} and k_{rG} for liquid (aqueous) and gas, respectively, such that effective permeability k_β to phase β (= A, G) is given by

$$k_\beta = k\, k_{r\beta}, \quad \text{where} \quad k = k_0\, F_{\phi S} = k_{00}\, k_{rr}\, F_{\phi S}, \tag{2.31}$$

and k_{rr} is the *k relative magnitude* that relates the permeability k_0 of a given medium to k_{00} of the reference medium at the same P and T. The factor $F_{\phi S}$ – see Equation (2.25) – denotes the effect of the evolution of solid phases (e.g., through salt precipitation or ice formation) and is equal to one if no solid phases are present. The term k_{rr} is introduced to account for situations in which the reference medium is different from the one under consideration, as is often the case when insufficient data are available and parameter estimation is based on scaling – such as the one described by Equation (2.33) – using known media as references. For a reference medium different from the one under consideration, $k_{rr} = k_0/k_{00}$. It is obvious that $k_{rr} = 1$ when the same medium is used as reference.

The relative permeabilities are functions of the phase saturations S_β (fraction of pore space occupied by phase β), $k_{r\beta} = k_{r\beta}(S_\beta)$, and are usually obtained by measurement on laboratory specimen of porous media. **Figure 2.3** gives examples of commonly used liquid and gas relative permeabilities. In Section 6, the reader can find a detailed

description of the various relative permeability options that are available in TOUGH+ v1.5 for the description of multi-phase flow.

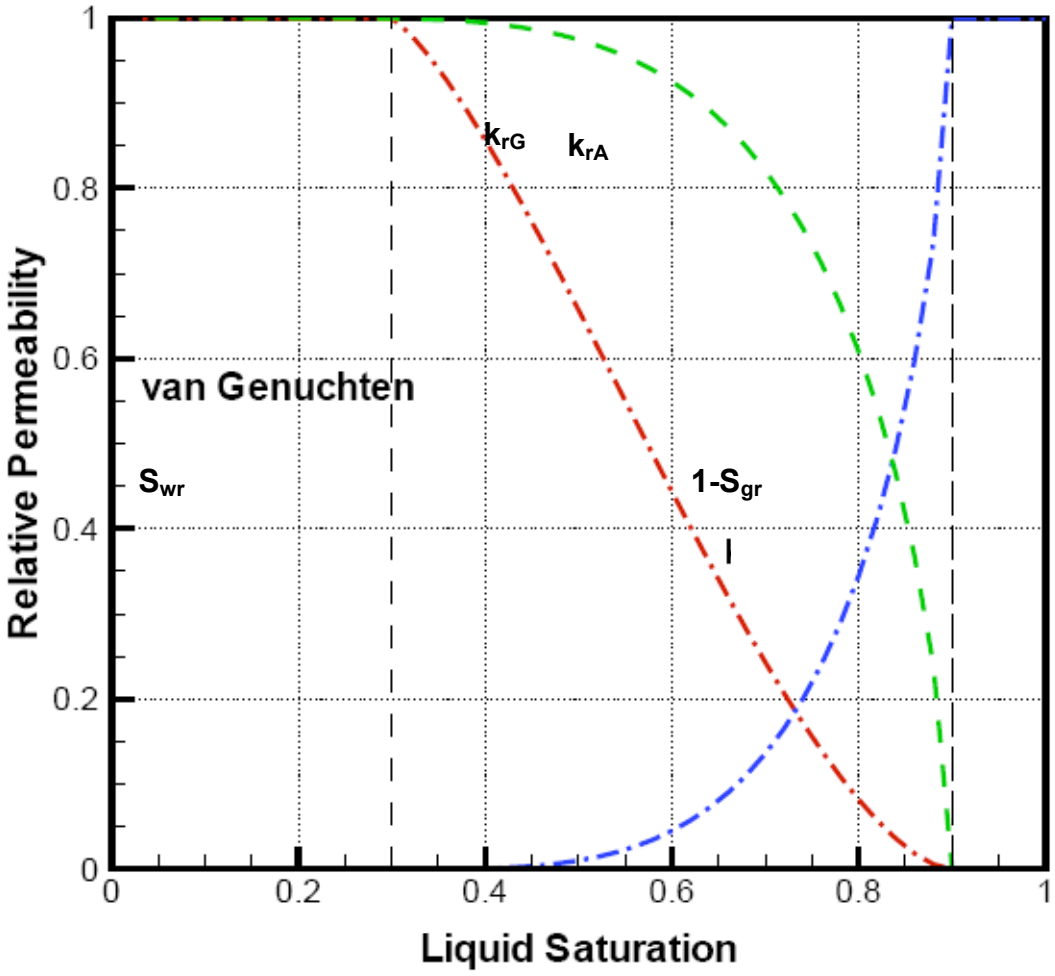

Figure 2.3. Liquid and gas relative permeabilities based on the van Genuchten model [*Finsterle*, 1999].

2.10.2. Capillary pressure

Surface tension effects between different phases give rise to '*capillary pressures*', denoted by P_{cap}. Most mineral surfaces are preferentially wetted by water and, under partially-

saturated conditions, the pressure of the wetting (aqueous) phase inside a porous medium will be less than that of the non-wetting (gas) phase (see Equation 2.9). The pressure difference between the two phases is termed capillary pressure P_{cap}, because it relates to the phenomenon of water level rise in a capillary tube. As indicated earlier, by convention P_{cap} has a negative value in TOUGH+ v1.5.

Issues relating to capillary pressure can be conveniently discussed with reference to the pore size distribution of the porous medium. **Figure 2.4** shows a schematic probability density function (pdf) for pore sizes, which expresses the probability *p(r)* of having pores with radius *r*. At a given capillary pressure P_{cap}, pores with radius $r' \le r$ may be water-filled, where the cutoff radius r is related to the capillary pressure by the Young-Laplace equation (reference)

$$P_{cap} = \frac{2\sigma}{r}\cos(\omega), \tag{2.32}$$

where σ is the surface tension (energy per unit surface area) at the water-gas interface, and ω is the contact angle, which usually is close to zero for preferentially water-wet minerals.

In introducing Equation (2.30) we stated that pores with $r' \le$ r may be water filled, but whether indeed all pores with $r' \le r$ will be water-filled at a prevailing capillary pressure P_{cap} given Equation (2.30) – and those with $r' > r$ will all be gas-filled – depends upon issues of *pore accessibility* that are not captured by the pore size distribution. The issue of accessibility arises because pores of certain radius may be entirely surrounded by larger pores, or by smaller pores, so that either water may not enter them during a wetting process, or may not be removed from them during a draining process. Accessibility gives rise to the well-known phenomenon of *capillary hysteresis*, where at a given magnitude of

capillary pressure, water saturation will generally be larger during a drainage process than during a wetting process [*de Marsily*, 1986]. Although capillary hysteresis is a well-established effect, it is seldom taken into account in modeling applications; this is partially due to numerical difficulties associated with it, partially because on the dependence of the corresponding relative permeability and capillary pressure on the pathway to the current state (as opposed to just the phase saturations), and partially because information on applicable parameters is rarely available.

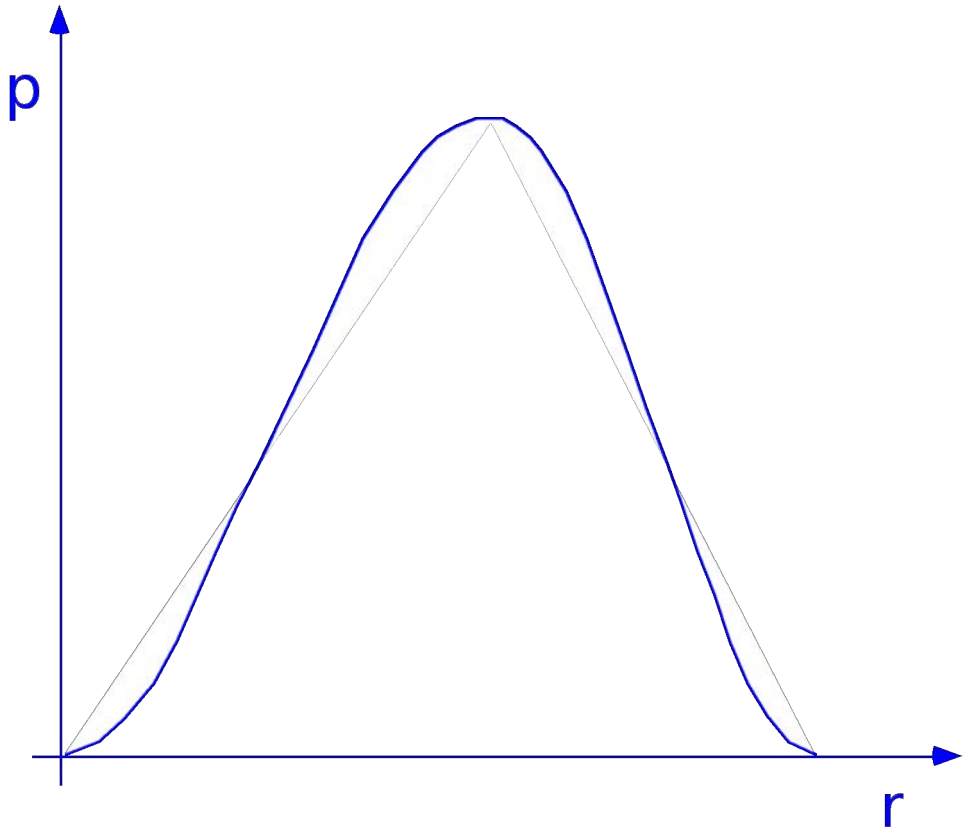

Figure 2.4. Schematic of probability density function p(r) for pore size distribution.

2.10.3. Effect of solid phase deposition on capillary pressure

The discussion in this section follows closely that in *Moridis et al.* [2008; 2009; 2012].

The effects of solids deposition become considerably more complicated, and involve more than just permeability change, when multiple fluid phase are present, such as water and gas. The capillary pressure P_{cap} may be profoundly altered when solids are deposited.

Formation of solid phases will alter the pore size distribution, generally reducing pore sizes, and thereby giving rise to stronger capillary pressures (see Equation 2.25). In order to estimate these changes, we require information on the original pore size distribution of the medium, and on the manner in which the pore size distributions will be altered during solids deposition. As no such information is presently available for the medium studied here, we proceed in a more phenomenological manner, and relate changes in capillary pressures to overall changes in porosity and permeability of the medium. Examining a variety of unconsolidated media, *Leverett* [1941] determined a dependence of capillary pressure on permeability and porosity, as follows.

$$P_{cap}(S_A) = \sqrt{\frac{k_{00}}{k} \cdot \frac{\phi}{\phi_{00}}}\, P_{cap,00} \tag{2.33}$$

where $P_{cap,00}$ is the capillary pressure corresponding to **a reference medium at the reference conditions**, at which the permeability and porosity of the porous medium are k_{00} and ϕ_{00}, respectively. Equation (2.33) is used in the present analysis, in the following manner. We represent *active* solids (e.g., ice that may melt or hydrate that may dissociate, as opposed to solid minerals which are inert) by means of a *solid saturation*, denoted by

$S_S = S_H + S_I$, which measures the fraction of active pore space occupied by solids. The fraction of pore space available to fluid phases is $S_A + S_G$, and we have the constraint

$$S_A + S_G \quad = \quad 1 - S_S \tag{2.34}$$

Let $P_{cap,00}$ denote the capillary pressure function applicable to a porous medium free of solid saturation ($S_S = 0$) with a reference porosity ϕ_{00}. The total current porosity ϕ and the active (or available) porosity ϕ_a available to fluids are then defined by the equation

$$\phi_a = \underbrace{\left(F_{PT}\ \phi_{rr}\ \phi_{00}\right)}_{\phi}\ (S_A + S_G) \ \Rightarrow \frac{\phi_a}{\phi_{00}} = F_{PT}\ \phi_{rr}\ (S_A + S_G) = F_{PT}\ \phi_{rr}\ \left(1\text{-}S_S\right)\ , \tag{2.35}$$

where the term F_{PT} is a *porosity adjustment factor* that accounts for the effects of pressure and temperature on porosity, and ϕ_{rr} is the ϕ *relative magnitude*, which relates the porosity ϕ_0 of a given medium to the porosity ϕ_{00} of the reference medium at the same reference P and T. This term is introduced to account for situations in which the reference medium is different from the one under consideration, as is often the case when insufficient data are available and parameter estimation is based on scaling using known media as references. For a reference medium different from the one under consideration, $\phi_{rr} = \phi_0/\phi_{00}$. It is obvious that $\phi_{rr} = 1$ when the same medium is used as reference.

When changes in P and T are not large in geomechanically stable media, then F_{PT} can be estimated from the following equation:

$$F_{PT} = \frac{\phi}{\phi_{rr}\ \phi_{00}} = \frac{\phi}{\phi_0} = \exp\left[\alpha_P\ \Delta P + \alpha_T\ \Delta T\right] \approx 1 + \alpha_P\ \Delta P + \alpha_T\ \Delta T\ , \tag{2.36}$$

where $\Delta P = P - P_0$, $\Delta T = T - T_0$, α_P and α_T are the pore compressibility and thermal expansivity, respectively (see discussion in Section 6.2). For large ΔP and/or ΔT in compressible or geomechanically unstable media, F_{PT} can be estimated from a full

geomechanical model that relates the resulting changes in geomechanical stresses and strains to changes in porosity.

The argument in the capillary pressure function $P_{cap,0}$ on the r.h.s of Equation (2.33) is the aqueous saturation S_A, referred to total fluid porosity. We measure liquid saturation on a scale that refers to total *active* (fluid plus hydrate- and ice-filled) pore space in the ice- and hydrate-free porous medium. In the medium with solid saturation S_S, S_A corresponds to a *scaled* saturation

$$S_A^* = \frac{S_A}{S_A + S_G} \tag{2.37}$$

relative to fluid-filled pore space, and this is the value to be used in the estimation of $P_{cap,0}$ in the r.h.s of Equation (2.44). In the next session we will discuss the estimation of $k_{r\varphi}$ in the presence of solid phases (i.e., ice and/or hydrate) for use in Equation (2.44).

2.10.4. Effect of solid phase deposition on relative permeability

From Equation (2.31), the partitioning of effective permeability to a fluid phase β into a porous medium- and solid-saturation dependent part ($k_0\ F_{\varphi S}$) and a fluid saturation-dependent part $k_{r\beta}$ is a matter of convention and convenience. It leads to a conceptual ambiguity in the representation of permeability reduction from solid deposition in multiphase flow. Indeed, such permeability reduction may be attributed either to a change in *absolute* or *intrinsic* permeability (as described by the product $k_0\ F_{\varphi S}$), as is done for single-phase flow, or it may be attributed to a change in the fluid relative permeability $k_{r\beta}$.

When hydrate and/or ice forms inside a partially water-saturated porous medium, such formation clearly must start in the water-filled portion of the pore space, but may not

remain limited to the water-filled portion, as the solid crystals may grow and extrude into primarily gas-filled pores. In the absence of specific pore-scale information about where hydrate and/or ice will likely form, it is not possible to ascertain the applicability of relationships such as Equation (2.25) to the permeability reduction associated with hydrate formation and/or freezing. Even if applicable, appropriate parameters for the problem under study here are lacking.

We proposed two alternative models to describe the wettability processes (relative permeability and capillary pressure) in hydrate- and/or ice-bearing media [*Moridis et al.,* 2008; 2011; 2012; *Moridis*, 2014]. The first model, hereafter referred to as the "*Original Porous Medium*" (OPM) model, is based on the treatment of (a) porosity as unaffected by the emergence of hydrates and/or ice (although subject to change due to changes in P and T), (b) of the intrinsic permeability of the porous media as unchanging during the evolution of the solid phases, and (c) of the fluid flow as a relative permeability issue controlled by the saturations of the various phases in the pores. The second family of models, hereafter referred to as the "*Evolving Porous Medium*" (EPM) models, considers the evolution of the solid phases (hydrate and ice) as tantamount to the creation of a new porous medium with continuously changing porosity and intrinsic permeability, the pore space of which is occupied only by the two fluid phases (aqueous and gas).

2.10.4.1. The OPM model. This simpler model represents permeability reduction as relative permeability effects, and does not require any new parameters to be introduced. More specifically, this model assumes that in the presence of solid phase(s), relative permeability to each fluid phase is given by the same function $k_{r\beta}(S_\beta)$ as in the absence of

solids. This means that aqueous phase relative permeability $k_{rA} = k_{rA}(S_A)$ depends only on the aqueous saturation S_A, and is the same, regardless of how the remaining fraction ($1-S_A$) of the pore space is divided between gas and solid phases. A similar comment applies to gas relative permeability $k_{rG} = k_{rG}(S_G)$.

Setting aside for a moment the issue of ($k_{r\phi}$ k_{rS}), permeability reduction for the fluid phases then occurs simply because, when S_S increases, fluid phase saturations S_A and S_G generally must decrease also, as dictated by the constraint in Equation (2.32). This prescription is tantamount to asserting that liquid phase flow behaves as though solids deposition occurs entirely in what would otherwise be gas-filled pore space, while gas phase flow behaves as though solids deposition occurs entirely in what otherwise would be liquid-filled pore spaces. It is obvious that solids deposition cannot simultaneously occur only in liquid and only in gas-filled pore spaces, which points to a limitation of the proposed permeability reduction model. We nonetheless feel that a model that introduces no new and uncertain parameters is preferable to a model that does.

The permeability adjustment factor is computed from the following expression:

$$F_{\phi S} = k_{r\phi}\, k_{rS}\,, \tag{2.38}$$

where $k_{r\phi}$ is the *permeability ϕ-factor* that describes the effect of changes in ϕ on permeability, and k_{rS} is the *permeability S-factor* that relates reduction in the intrinsic permeability to the presence of solid phases (such as ice, hydrates or precipitating salts).

In the OPM model, $k_{rS} = 1$ by definition, and the permeability ϕ-factor in Equation (2.31) can be computed as

$$k_{r\phi} = \begin{cases} 1 & \text{when the effect of } \phi \text{ changes on } k \text{ is neglected} \\ \exp\left[\gamma\left(F_{PT}-1\right)\right] & \text{when the effect of } \phi \text{ changes on } k \text{ is accounted for,} \end{cases} \quad (2.39)$$

where γ is an empirical parameter [*Rutqvist and Tsang*, 2002], and F_{PT} is computed from Equation (2.36).

In the OPM model, P_{cap} are estimated from Equation (2.33), in which:

- $\phi/\phi_{00} = \phi_{rr} F_{PT}$ is computed from Equation (2.36)
- $k_{00}/k = 1/k_{rr} k_{r\phi}$ is computed from Equations (2.31) and (2.38), and
- $P_{cap,00}$ is computed based on the scaled saturations S^* of Equation (2.37)

Thus, the final expression for estimating the capillary pressure in the OPM model is:

$$P_{cap} = \sqrt{\frac{\phi_{rr}}{k_{rr}} \cdot \frac{F_{PT}}{k_{r\phi}}} \cdot P_{cap,00}(S^*) \quad (2.40)$$

Additional scaling can be introduced by using the active porosity ϕ_a and ϕ_a/ϕ_{00} from Equation (2.35) – as opposed to ϕ/ϕ_{00} from Equation (2.36) – in the computation of Equation (2.33).

2.10.4.2. The EPM models. In recognition of, and attempting to overcome, the limitations of the OPM permeability reduction model, we have proposed two EPM models and performed sensitivity studies using the absolute (intrinsic) permeability modifications that will be discussed in this section. While the EPM models provide valuable insights, a more consistent and defensible model – based on both theoretical analyses and laboratory and field studies – for the effects of emerging solid phases on fluid permeabilities should be developed in the future.

EPM Model #1. With intrinsic permeability modification based on relative permeabilities, what absolute permeability should be used in the Leverett scaling Equation (2.33). An attractive possibility would be to set $k = k_0 (k_{rA} + k_{rG})$. This, however, is not acceptable because the sum of liquid and gas relative permeabilities depends not just on the solid saturation (S_S), but on S_A and S_G individually.

As a plausible alternative, we consider the permeability reduction when the fluid-available pore space is either entirely liquid-filled or entirely gas-filled, which leads to

$$k_{rS} = k_{rA}\left(S_A = 1 - S_S\right) \quad \text{or} \quad k_{rS} = k_{rG}\left(S_G = 1 - S_S\right).$$

Note that either expression depends only on solid saturation S_S. As an estimate of the k_{rS}, we then take the average of the two,

$$k_{rS} = \frac{1}{2}\left[k_{rA}\left(S_A = 1 - S_S\right) + k_{rG}\left(S_G = 1 - S_S\right)\right] \tag{2.41}$$

Equation (2.38) provides a simple estimate of the permeability ϕ-factor. Then the phase effective permeabilities are computed using Equation (2.31), in which:

- $k_{r\beta}$ is computed based on the scaled saturations from Equation (2.37),
- $F_{\phi S}$ is computed from Equation (2.38),
- $k_{r\phi}$ is computed from Equation (2.39), and
- k_{rS} is computed from Equation (2.41).

The capillary pressure in the EPM #1 model is estimated using Equation (2.44), in which the various terms are computed as follows:

- ϕ_a/ϕ_{00}, computed from Equation (2.35), is used instead of ϕ/ϕ_{00},
- $k_{00}/k = 1/k_{rr} F_{\phi S}$ is computed from Equations (2.38) and (2.38),
- $k_{r\phi}$ is computed from Equation (2.39), and

- k_{rS} is computed from Equation (2.41).

Thus, the final expression for estimating the capillary pressure in the EPM #1 model is:

$$P_{cap} = \sqrt{\frac{\phi_{rr}}{k_{rr}} \cdot \frac{F_{PT}\left(1-S_S\right)}{k_{r\phi}k_{rS}}} \cdot P_{cap,00}(S^*) \tag{2.42}$$

EPM Model #2. The only difference between this model and the EPM #1 model is in the equation used to estimate the k_{rS} term, with all other equations applying unchanged. In the EPM#2 model, the quantity $F_{\phi S} = k_{r\phi}\, k_{rS}$ in Equation (2.38) is provided by Equation (2.26), leading to

$$k_{rS} = \left[\frac{\phi_0\ \left(1\text{-}S_S\right)-\phi_c}{\phi_0-\phi_c}\right]^n \tag{2.43}$$

The term $k_{r\phi}$ is obtained from Equation (2.39). Thus, the effective permeabilities in the EPM #2 model are computed from Equation (2.31). Similarly, the capillary pressure in the EPM #2 model is estimated using Equation (2.42) and k_{rS} from Equation (2.43).

2.10.5. Pore compressibility of unconsolidated media in the presence of cementing solid phases

While the pore compressibility α_P in Equation (2.23) can be considered as a constant or even as a function of pressure during fluid flow through consolidated (lithified) porous media and/or in unconsolidated media, this approach is inadequate when cementing solid phases (such as ice and/or hydrates) are present in the pores. This is because the presence of these solid phases imparts stiffness and increases the geomechanical strength of the solid phase-impregnated porous medium, the porosity ϕ of which reacts much slower to

variations in pressure P. Thus, to accurately represent the evolution of ϕ as a function of P in these cases needs to account for the effect of the saturation S_S of such solid phases.

The most appropriate method for accounting for the effect of S_S on the porosity of unconsolidated media is by solving the coupled flow-geomechanical problem, estimating variations in P, T and phase saturation, and computing the corresponding changes in stresses and strains. These are then used to compute changes in ϕ and k. Such coupling is a possibility in the TOUGH+ v1.5 code, which allows the use of the commercial geomechanical model `FLAC3D` [*ITASCA*, 2002] to evaluate the interaction between flow and geomechanical properties. This model is automatically invoked if the corresponding executable file `FLAC3D.exe` is present in the TOUGH+ v1.5 directory and appropriate inputs are provided to the TOUGH+ code (see detailed discussion in Section 5).

If the `FLAC3D` model is not available or is not invoked (a frequent choice, given the large execution times required for such fully coupled flow-geomechanical problems), it is possibly to describe the effect of cementing solid phases S_S on the porosity ϕ and the intrinsic permeability k of unconsolidated media by employing an empirical model that describes the media compressibility as:

$$\alpha_P = \exp\left\{\ln\alpha_{PL} + (\ln\alpha_{PU} - \ln\alpha_{PL})\left[1 - B_x(2.25, 2.25, S_S^*)\right]\right\}, \tag{2.44}$$

where

$$S_S^* = \frac{S_S - S_{S\min} + \delta}{S_{S\max} - S_{S\min} + 2\delta}, \tag{2.45}$$

α_{PL} is the lower limit of the medium compressibility (corresponding to the full stiffening/strengthening effect of the presence of cementing solid phases such as ice and/or hydrates), α_{PU} is the upper limit of the medium compressibility (corresponding to

the absence of cementing solid phases), B_x is the incomplete beta function, S_{Smin} is the largest S_S saturation at which $\alpha_P = \alpha_{PU}$, S_{Smax} is the lowest S_S saturation at which $\alpha_P = \alpha_{PL}$, and δ is a smoothing factor. Equation (2.44) is based on geomechanical and geophysical data derived from laboratory and field observations, and results in the curve of **Figure 2.5** that scans between the α_{PU} and the α_{PL} compressibility limits.

The relative porosity ϕ/ϕ_0 is estimated from Equation (2.36), which applies unchanged, but with the composite compressibility α_P computed from Equation (2.44). **Figure 2.6** shows the relationship between the relative porosity ϕ/ϕ_0 and the pressure drop ΔP in an unconsolidated medium, and describes the cementing effect of solid phases on the medium behavior.

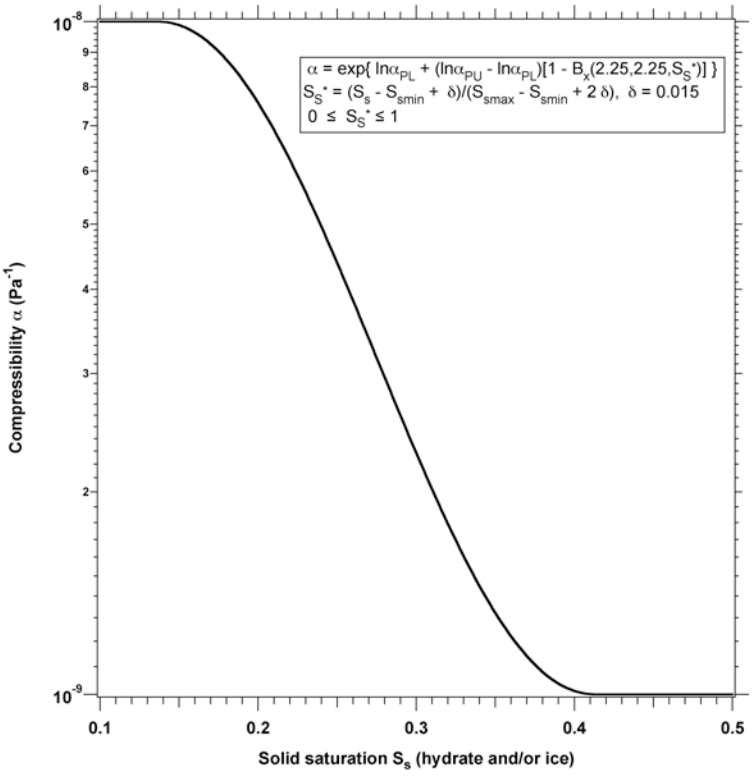

Figure 2.5. Compressibility of an unconsolidated porous medium impregnated with cementing solid phases (ice and/or hydrates). In this example, $S_{Smin} = 0.15$, $S_{Smax} = 0.4$, $\alpha_{PU} = 10^{-8}$ Pa^{-1}, $\alpha_{PL} = 10^{-9}$ Pa^{-1} and $\delta = 0.015$ [*Moridis et al.,* 2008; 2009; 2011].

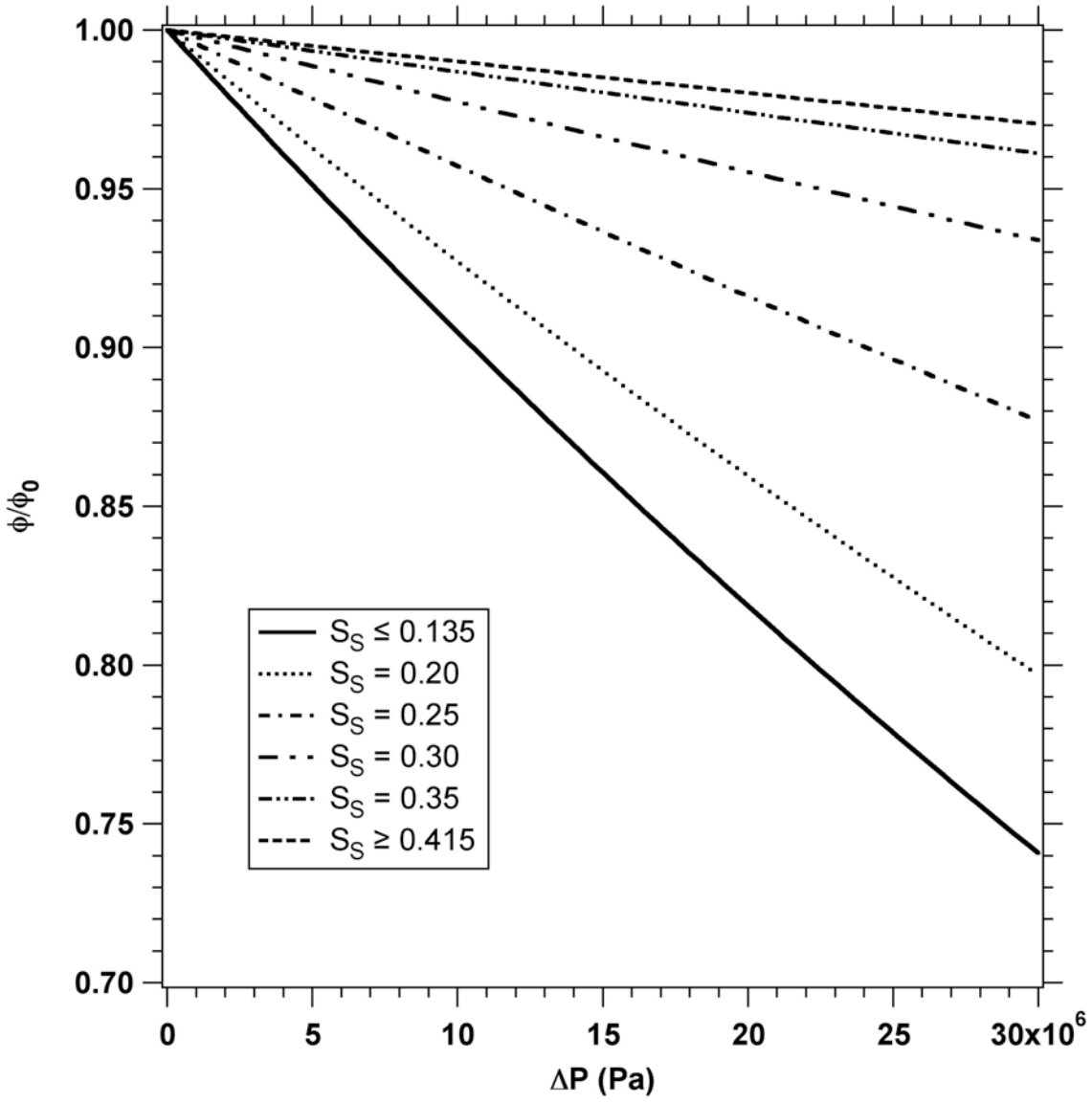

Figure 2.6. Effect of the varying compressibility described in **Figure 2.5** on the porosity of an unconsolidated porous medium undergoing depressurization for various levels of saturation S_S of cementing solid phases [*Moridis et al.,* 2008; 2009; 2011].

2.11. Description of Flow in Fractured Media

The discussion in this section hews very closely to that of *Pruess et al.* [1999; 2012]. Figure 2.7 illustrates the classical double-porosity concept for modeling flow in fractured-porous media as developed by *Warren and Root* [1963]. Matrix blocks of low permeability are embedded in a network of interconnected fractures. Global flow in the reservoir occurs only through the fracture system, which is described as an effective porous continuum. Rock matrix and fractures may exchange fluid (or heat) locally by means of '*interporosity flow*', which is driven by the difference in pressures (or temperatures) between matrix and fractures. Warren and Root approximated the interporosity flow as being quasi-steady, with rate of matrix-fracture interflow proportional to the difference in (local) average pressures.

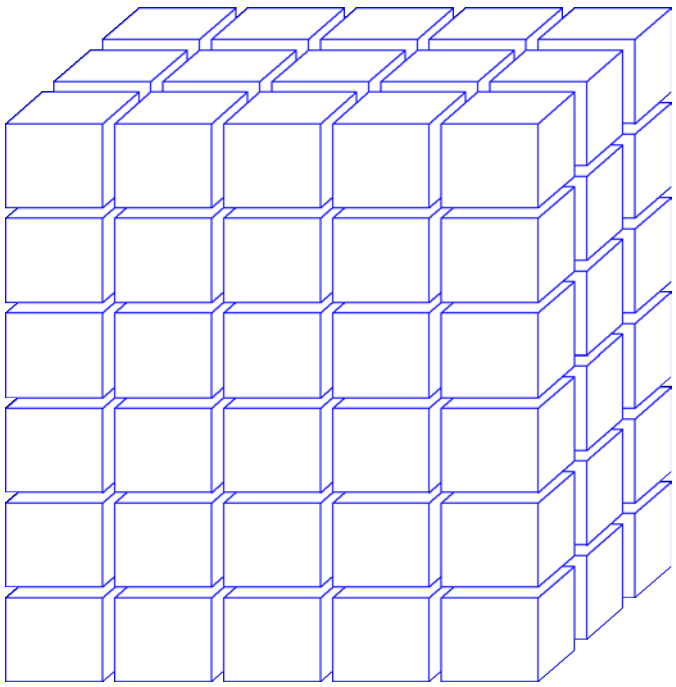

Figure 2.7. Idealized double porosity model of a fractured porous medium [*Pruess*, 1983].

The quasi-steady approximation is applicable to isothermal single-phase flow of fluids with small compressibility, where pressure diffusivities are large, so that pressure changes in the fractures penetrate quickly all the way into the matrix blocks. However, for multiphase flows, or coupled fluid and heat flows, the transient periods for interporosity flow can be very long (tens of years). In order to accurately describe such flows it is necessary to resolve the driving pressure, temperature, and mass fraction gradients at the matrix/fracture interface. In the method of "multiple interacting continua" (MINC) [*Pruess and Narasimhan*, 1982; 1985; *Pruess*, 1983), resolution of these gradients is achieved by appropriate subgridding of the matrix blocks, as shown in **Figure 2.8**. The MINC concept is based on the notion that changes in fluid pressures, temperatures, phase compositions, etc., due to the presence of sinks and sources (production and injection wells) will propagate rapidly through the fracture system, while invading the tight matrix blocks only slowly. Therefore, changes in matrix conditions will (locally) be controlled by the distance from the fractures. Fluid and heat flow from the fractures into the matrix blocks, or from the matrix blocks into the fractures, can then be modeled by means of one-dimensional strings of nested grid blocks, as shown in **Figure 2.8**.

In general it is not necessary to explicitly consider subgrids in all of the matrix blocks separately. Within a certain reservoir subdomain (corresponding to a finite difference grid block), all fractures will be lumped into continuum # 1, all matrix material within a certain distance from the fractures will be lumped into continuum # 2, matrix material at larger distance becomes continuum # 3, and so on. Quantitatively, the subgridding is specified by means of a set of volume fractions `VOL(j)`, `j = 1, ..., J`, into which the primary porous medium grid blocks are partitioned.

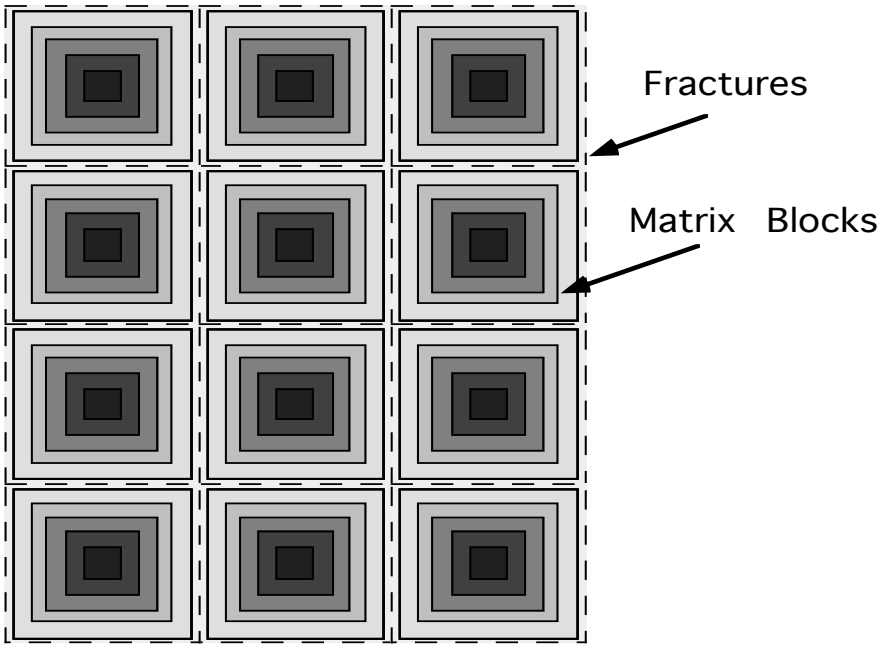

Figure 2.8. Subgridding in the method of "multiple interacting continua" (MINC) [*Pruess*, 1983].

The MINC-process in the **`MeshMaker.f95`** (a companion code distributed with TOUGH+, see Section 7) operates on the element and connection data of a porous medium mesh to calculate, for given data on volume fractions, the volumes, interface areas, and nodal distances for a secondary fractured medium mesh. The information on fracturing (spacing, number of sets, shape of matrix blocks) required for this is provided by a *proximity function* `PROX`(x) which expresses, for a given reservoir domain V_o, the total fraction of matrix material within a distance x from the fractures. If only two continua are specified (one for fractures, one for matrix), the MINC approach reduces to the conventional double-porosity method. Full details are given in a separate report [*Pruess*, 1983].

The MINC-method as implemented in the **`MeshMaker.f95`** code can also describe global matrix-matrix flow. **Figure 2.9** shows the most general approach, often

referred to as *dual permeability*, in which global flow occurs in both fracture and matrix continua. It is also possible to permit matrix-matrix flow only in the vertical direction. For any given fractured reservoir flow problem, selection of the most appropriate gridding scheme must be based on a careful consideration of the physical and geometric conditions of flow. The MINC approach is not applicable to systems in which fracturing is so sparse that the fractures cannot be approximated as a continuum. A thorough discussion on the various approached for the treatment of fractured media in TOUGH2 [*Pruess et al*., 1999; 2012] and in TOUGH+ v1.5 (which closely follows the TOUGH2 approach) can be found in *Doughty et al.* [1999].

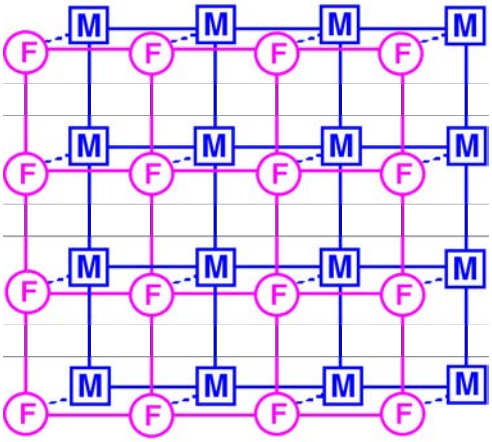

Figure 2.9. Flow connections in the "dual permeability" model. Global flow occurs between both fracture (F) and matrix (M) grid blocks. In addition there is F-M interporosity flow [*Pruess et al.*, 1999].

3.0. Design and Implementation of TOUGH+ v1.5

3.1. Primary Variables

The thermodynamic state and the distribution of the mass components among the possible phases in a TOUGH+ v1.5 simulation are determined from the equation of state (EOS) solved in the specific TOUGH+ application option. Following the standard approach employed in the TOUGH2 [*Pruess et al.*, 1999] family of codes, in TOUGH+ the system is defined uniquely by a set of N_κ primary variables (where κ denotes the number of mass and heat components under consideration) that completely specifies the thermodynamic state of the system [*Pruess et al*., 1999; 2012]. N_κ is the sum of the number of the mass components (TOUGH+ being a compositional simulator) augmented by one – the heat balance equation (see Section 2.2), with the T being (usually) the corresponding primary

variable that is included in the computations even in isothermal problems because it would not be possible to estimate the fluid thermophysical properties of fluids without it.

The primary variables are the minimum number of *independent* variables, knowledge of which permits the simultaneous solution for all the state (=dependent, secondary) variables associated with the problem, thus uniquely defining the system. For example, knowledge of pressure and temperature allows the complete definition of the thermophysical properties of water in its liquid and vapor phases, thus P and T are appropriate primary variables to define the aqueous and the vapor phase. However, these are inappropriate primary variables during the liquid-vapor phase coexistence because they are no longer independent, as there is a well-defined and unique relationship between P and T along the saturation line.

The importance of selecting appropriate primary variables in the simulations of fluid and heat flow processes cannot be overemphasized, especially when phase changes are involved. Inappropriate selection of primary variables can lead not only to slow and inefficient computations, but also to complete failure of the simulation for lack of convergence. Although the number N_K of the primary variables is initially set at the maximum expected in the course of the simulation and does not change during the simulation, the thermodynamic quantities used as primary variables can change in the process of simulation to allow for the seamless consideration of emerging or disappearing phases and components. This is because the change of phases and the evolution and disappearance of components almost involves sensitivity to different parameters, which inevitably necessitates change of the primary variables. Switching primary variables as the state of the fluids changes has been standard practice in TOUGH2 [*Pruess et al*., 1999; 2012], and continues to be so in TOUGH+ v1.5. Experience thus far has indicated that the

change in primary variables, coupled with the selection of the appropriate ones, is a very robust method that is capable of solving even the most demanding problems of multi-phase, multi-component flow and transport in porous/fractured media.

The primary variables used in the various TOUGH+ v1.5 application options vary with the type of problem EOS being solved and cannot be generalized. Tables of the specific primary variables used for each state (phase co-existence) of fluids are listed in the User's Manuals of the application options that solve the corresponding problem.

3.2. Space and Time Discretization

The continuum equations (2.3) are discretized in space using the integral finite difference method (IFD) [*Edwards*, 1972; *Narasimhan and Witherspoon*, 1976]. Introducing appropriate volume averages, we have

$$\int_{V_n} M \, dV \;=\; V_n \, M_n \,, \tag{3.1}$$

where M is a volume-normalized extensive quantity, and M_n is the average value of M over V_n. Surface integrals are approximated as a discrete sum of averages over surface segments A_{nm}:

$$\int_{\Gamma_n} \mathbf{F}^{\kappa} \bullet \mathbf{n} \, d\Gamma \;=\; \sum_m A_{nm} F_{nm} \,, \tag{3.2}$$

Here F_{nm} is the average value of the (inward) normal component of **F** over the surface segment A_{nm} between volume elements V_n and V_m. The discretization approach used in the integral finite difference method and the definition of the geometric parameters are illustrated in **Figure 3.1**.

The discretized flux is expressed in terms of averages over parameters for elements V_n and V_m. For the basic Darcy flux term in Equation (2.15), we have

$$F_{\beta,nm} = -k_{nm}\left[\frac{k_{r\beta}\,\rho_\beta}{\mu_\beta}\right]_{nm}\left[\frac{P_{\beta,n}-P_{\beta,m}}{D_{nm}}-\rho_{\beta,nm}\,g_{nm}\right], \tag{3.33}$$

where the subscripts (*nm*) denote a suitable averaging at the interface between grid blocks n and m (interpolation, harmonic weighting, upstream weighting). $D_{nm} = D_n + D_m$ is the distance between the nodal points *n* and *m*, and g_{nm} is the component of gravitational acceleration in the direction from *m* to *n*. Discretization of diffusive fluxes raises some subtle issues, and is discussed separately in Section 3.5.

Substituting Equations. (3.l) and (3.2) into the governing Equation (2.3), a set of first-order ordinary differential equations in time is obtained.

$$\frac{d\,M_n^\kappa}{d\,t} = \frac{1}{V_n}\sum_m A_{nm}\,F_{nm}^\kappa + q_n^\kappa \tag{3.4}$$

Time is discretized as a first-order finite difference, and the flux and sink and source terms on the right-hand side of Equation (3.4) are evaluated at the new time level, $t^{k+1} = t^k + \Delta t$, to obtain the numerical stability needed for an efficient calculation of strongly nonlinear problems (such as the ones involving multiphase flow and phase changes). This treatment of flux terms is known as *fully implicit*, because the fluxes are expressed in terms of the unknown thermodynamic parameters at time level t^{k+l}, so that these unknowns are only implicitly defined in the resulting equations [*Peaceman*, 1977].

The time discretization results in the following set of coupled non-linear, algebraic equations

$$R_n^{\kappa,k+1} = M_n^{\kappa,k+1} - M_n^{\kappa,k} - \frac{\Delta t}{V_n}\left(\sum_m A_{nm} F_{nm}^{\kappa,k+1} + V_n q_n^{\kappa,k+1}\right) = 0 \qquad (3.5)$$

where we have introduced residuals $R_n^{\kappa,k+1}$. For each volume element (grid block) V_n, there are N_κ equations, so that for a system discretized into N_E grid blocks, Equation (3.5) represents a total of $N_\kappa \times N_E$ coupled non-linear equations.

3.3. The Newton-Raphson Iteration

The unknowns of Equation (3.5) are the $N_\kappa \times N_E$ independent primary variables $\{x_i;\ i = 1, \ldots, N_\kappa \times N_E\}$ which completely define the state of the flow system at time level t^{k+1}. These equations are solved by Newton/Raphson iteration, which is implemented as follows. We introduce an iteration index p and expand the residuals $R_n^{\kappa,k+1}$ in Equation (3.5) at iteration step $(p + 1)$ in a Taylor series in terms of those at index p, i.e.,

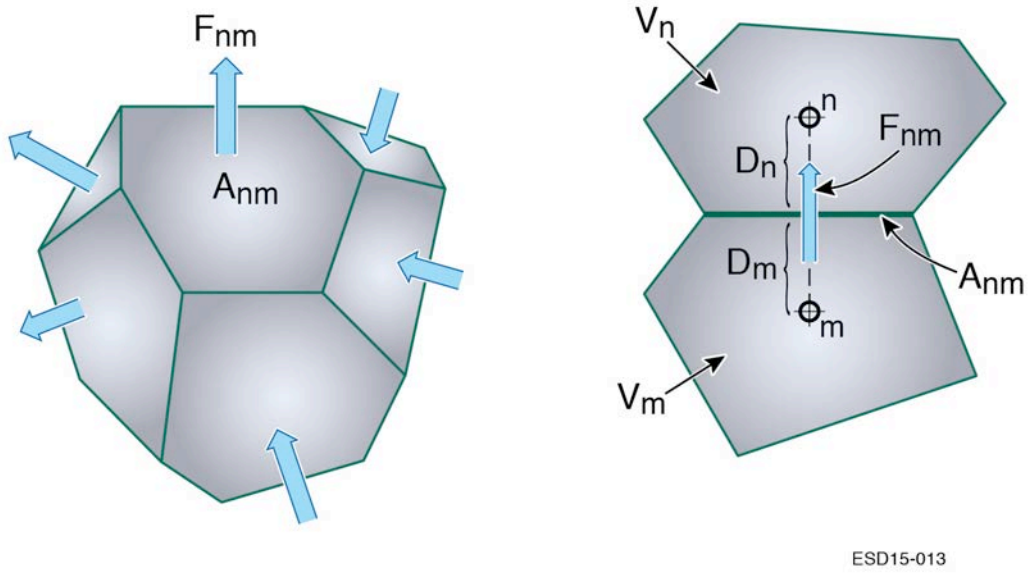

Figure 3.1. Space discretization and geometry data in the integral finite difference method.

$$R_n^{\kappa,k+1}\left(x_{i.p+1}\right)=R_n^{\kappa,k+1}\left(x_{i.p}\right)\ +\ \sum_{\mathrm{i}}\left.\frac{\partial R_n^{\kappa,k+1}}{\partial x_i}\right|_p\left(x_{i.p+1}-x_{i.p}\right)\ +\ \ldots\ =\ 0 \tag{3.6}$$

Retaining only terms up to first order, we obtain a set of N_κ x N_E linear equations for the increments $\left(x_{i,p+1}-x_{i,p}\right)$:

$$-\sum_{\mathrm{i}}\left.\frac{\partial R_n^{\kappa,k+1}}{\partial x_i}\right|_p\left(x_{i,p+1}-x_{i,p}\right)\ +\ \ldots\ =\ R_n^{\kappa,k+1}\left(x_{i,p}\right) \tag{3.7}$$

All terms $\partial R_n/\partial x_i$ in the Jacobian matrix are evaluated by numerical differentiation.

Equation (3.7) is solved by sparse direct matrix methods or iteratively by means of preconditioned conjugate gradients [*Moridis and Pruess*, 1995; 1998; *Pruess et al.*, 1999; 2012]. Iteration is continued until the residuals $R_n^{\kappa,k+1}$ are reduced below a preset convergence tolerance according to:

$$\left|\frac{R_{n,p+1}^{\kappa,k+1}}{M_{n,p+1}^{\kappa,k+1}}\right|\ \le\ \varepsilon_1 \tag{3.8}$$

The default (relative) convergence criterion is ε_1 = 10^{-5} (TOUGH+ input parameter `rel_convergence_crit`, see Section 10). When the accumulation terms are smaller than ε_2 (TOUGH+ input parameter `abs_convergence_crit`, default ε_2 = 1, see Section 10), an absolute convergence criterion is imposed,

$$\left|R_{n,p+1}^{\kappa,k+1}\right|\le\ \varepsilon_1\varepsilon_2 \tag{3.9}$$

The number of iterations to convergence varies with the nonlinearity of the problem. For well-behaved problems, convergence is usually attained in 3-4 iterations. If convergence cannot be achieved within a certain number of iterations (default = 8, see Section 10), the time step size Δt is reduced and a new iteration process is started.

3.4. Implications of the Space Discretization Approach

It is appropriate to add some comments about the space discretization technique in TOUGH+. The entire geometric information of the space discretization in Equation (3.5) is provided in the form of a list of grid block volumes V_n, interface areas A_{nm}, nodal distances D_{nm} and components g_{nm} of gravitational acceleration along nodal lines. There is no reference whatsoever to a global system of coordinates, or to the dimensionality of a particular flow problem.

The discretized equations are in fact valid for arbitrary irregular discretizations in one, two or three dimensions, and for porous as well as for fractured media. This flexibility should be used with caution, however, because the accuracy of solutions depends upon the accuracy with which the various interface parameters in equations such as (3.3) can be expressed in terms of average conditions in grid blocks. A general requirement is that there exists approximate thermodynamic equilibrium in (almost) all grid blocks at (almost) all times [*Pruess and Narasimhan*, 1985]. For systems of regular grid blocks referenced to global coordinates in cylindrical (r,z) and/or Cartesian (x,y,z) systems, Equation (3.5) is identical to a conventional finite difference formulation [*Peaceman*, 1977; *Moridis and Pruess*, 1992].

3.5. Space Discretization of Diffusive Fluxes

Space discretization of diffusive flux in multiphase conditions raises some subtle issues. A finite difference formulation for total diffusive flux, Equation (3.10), may be written as

$$\left(\mathbf{J}^{\kappa}\right)_{nm} = \left(\mathscr{D}_A^{\kappa}\right)_{nm} \frac{\left(X_A^{\kappa}\right)_m - \left(X_A^{\kappa}\right)_n}{D_{nm}} - \left(\mathscr{D}_G^{\kappa}\right)_{nm} \frac{\left(X_G^{\kappa}\right)_m - \left(X_G^{\kappa}\right)_n}{D_{nm}} \tag{3.10}$$

This expression involves the as yet unknown diffusive strength coefficients $(\Sigma_A^{\kappa})_{nm}$ and $(\Sigma_G^{\kappa})_{nm}$ at the interface, which must be expressed in terms of the strength coefficients in the participating grid blocks. Invoking conservation of diffusive flux across the interface between two grid blocks leads in the usual way to the requirement of harmonic weighting of the diffusive strength coefficients.

However, such weighting may in general not be applied separately to the diffusive fluxes in gas and liquid phases, because these may be strongly coupled by phase partitioning effects. This can be seen by considering the extreme case of diffusion of a water-soluble and volatile compound from a grid block in single-phase gas conditions to an adjacent grid block that is in single-phase liquid conditions. Harmonic weighting applied separately to liquid and gas diffusive fluxes would result in either of them being zero, because for each phase effective diffusivity is zero on one side of the interface. Thus total diffusive flux would vanish in this case, which is unphysical. In reality, tracer would diffuse through the gas phase to the gas-liquid interface, would establish a certain mass fraction in the aqueous phase by dissolution, and would then proceed to diffuse away from the interface through the aqueous phase. Similar arguments can be made in the less extreme situation where liquid saturation changes from a large to a small value rather than from 1 to 0, as may be the case in the capillary fringe, during infiltration events, or at fracture-matrix interfaces in variably saturated media.

TOUGH+ features the fully coupled approach employed in TOUGH2 [*Pruess et al.*, 1999; 2012], in which the space-discretized version of Equation (3.10) of the total

multiphase diffusive flux Equation (2.49) is re-written in terms of an effective multiphase diffusive strength coefficient and a single mass fraction gradient. Choosing the liquid mass fraction for this we have

$$\left(\mathbf{J}^{\kappa}\right)_{nm} = -\left[\left(\sum\nolimits_{A}^{\kappa}\right)_{nm} + \left(\sum\nolimits_{G}^{\kappa}\right)_{nm} \frac{\left(X_G^{\kappa}\right)_m - \left(X_G^{\kappa}\right)_n}{\left(X_A^{\kappa}\right)_m - \left(X_A^{\kappa}\right)_n}\right] \frac{\left(X_A^{\kappa}\right)_m - \left(X_A^{\kappa}\right)_n}{D_{nm}}, \qquad (3.11)$$

where the gas phase mass fraction gradient has been absorbed into the effective diffusive strength term (in braces). Flux conservation at the interface then leads to the requirement of harmonic weighting for the full effective strength coefficient. In order to be able to apply this scheme to the general case where not both phases may be present on both sides of the interface, we always define both liquid and gas phase mass fractions in all grid blocks, regardless of whether both phases are present. Mass fractions are assigned in such a way as to be consistent with what would be present in an evolving second phase.

This procedure is applicable to all possible phase combinations, including the extreme case where conditions at the interface change from single-phase gas to single-phase liquid. Note that, if the diffusing tracer exists in just one of the two phases, harmonic weighting of the strength coefficient in Equation (3.11) will reduce to harmonic weighting of either Σ_A^{κ} or Σ_G^{κ}. The simpler scheme of separate harmonic weighting for individual phase diffusive fluxes is retained as an option.

3.6. Code Units of the TOUGH+ v1.5 Code

TOUGH+ v1.5 is written in standard FORTRAN 95/2003. It has been designed for maximum portability, and runs on any computational form (Unix and Linux workstations,

PC, Macintosh) for which such compilers are available. Running TOUGH+ involves compilation and linking the code created by combining

(a) the units of the core code with

(b) the *supplemental* code units, corresponding to the specific application option that the simulator aims to address, and with

(c) code the units that are part of the TOUGH+ code ensemble but are invoked only to carry out computations needed by the application option.

The units of the core TOUGH+ are the following:

(1) **`T_Allocate_Memory.f95`**

Code unit that is responsible for the dynamic memory allocation (following input describing the size of the problem) and dimensioning of most arrays needed by the code, in addition to memory deallocation of unnecessary arrays.

(2) **`T_Utility_Functions.f95`**

Code unit that includes utility functions (including a variety of mathematical functions, table interpolation routines, sorting algorithms, etc.).

(3) **`T_Media_Properties.f95`**

Code unit that describes the hydraulic and thermal behavior of the geologic medium (porous or fractured), i.e., multiphase capillary pressure and relative permeability, and interface permeability, mobility and thermal conductivity.

(4) **`T_H2O_Properties.f95`**

Code unit that includes (a) all the water-related constants (parameters), and (b) procedures describing the water behavior and thermophysical properties/processes in its entire thermodynamic phase diagram. Because water is almost universally present in geologic media, this is the most commonly used supplemental TOUGH+ v1.5 code unit.

(5) **T_Geomechanics.f95**

Code unit that describes the geomechanically-induced changes on the flow properties of the porous media. These include porosity ϕ changes caused by pressure and/or temperature variations, intrinsic permeability k changes caused by porosity changes, and scaling of capillary pressures P_{cap} to reflect changes in ϕ and k. The ϕ and k changes are computed using either simplified of full geomechanical models. When the simplified model is invoked, ϕ is a function of (a) P and the pore compressibility α_P and (b) of T and the pore thermal expansivity α_T, while (c) k changes are estimated using empirical relationships (see Section 8). Changes in ϕ and k can also be computed by using a full geomechanical model, which can be optionally coupled with TOUGH+.

(6) **T_Main.f95**

Main program that organizes the calling sequence of the high-level events in the simulation process, and includes the writing of important general comments in the standard output files, timing procedures, and handling of files needed by the code and/or created during the code execution.

(7) **T_Matrix_Solvers.f95**

A linear algebra package that includes all the direct and iterative solvers available in TOUGH+ (see Section 10).

(8) **T_Executive.f95**

The executive unit of TOUGH+. It includes the procedures that advance the time in the simulation process, estimate the time-step size for optimum performance, populate the matrix arrays and invoke the solvers of the Jacobian, invoke special linear algebra for matrix pre-processing in cases of very demanding linear algebra problems, compute mass and energy balances, compute rates in sources and sinks, compute binary diffusion coefficients, write special output files, and conduct other miscellaneous operations.

(9) **T_Inputs.f95**

This code unit includes the procedures involved in the reading of the general input files needed for TOUGH+ simulations. It does not include any procedure reading the data needed by the application option (see later discussion).

All the code units listed above are common to **all** TOUGH+ simulations. TOUGH+ also includes *supplemental* code units that are part of the wider TOUGH+ code ensemble and are available for specialized computations needed by the various application options. These are the following:

(1) **T_RealGas_Properties.f95**

Code unit that includes (a) a complete database of the parameter values of 12 gases that are needed to estimate all their properties (see below) needed for TOUGH+ simulations, and (b) procedures describing the equation of state (EOS) of real gases (pure or mixtures) using any of the Peng-Robinson, Redlich-Kwong, or Soave-Redlich-Kwong cubic EOS model. The procedures in this code unit compute the following parameters and processes: compressibility, density, fugacity, enthalpy (ideal and departure), internal energy (ideal and departure), entropy (ideal and departure), thermal conductivity, viscosity, binary diffusion coefficients, solubility in water, and heat of dissolution in water.

(2) **T_Salinity_Effects.f95**

Code unit that computes all necessary properties and parameters in application options that involve salinity (e.g., brines). It estimates the salt solubility in H_2O, the halite density and enthalpy, the effect of salinity on the density, viscosity and enthalpy of the aqueous phase, as well as on the vapor pressure of H_2O.

(3) **T_NonDarcian_Flow.f95**

Code unit that computes all parameters and variables needed for the application of non-Darcian flow through porous and fractured media by accounting for inertial (turbulent) or viscous (slippage) effects. Thus, this unit reads all the non-Darcian

flow inputs, and then uses them to compute all the parameters of the turbulent flow options (*Forcheimer* [1901] or *Barre and Conway* [2007]), of slippage flow (Klinkenberg flow [*Klinkenberg*, 1941], Knudsen diffusion [*Freeman et al*., 2011] or the Dusty Gas Model [*Mason and Malinauskas*, 1983; *Webb*, 1998]).

Finally, to develop a fully functional code, the core and supplemental code units of TOUGH+ are combined with the application *option-specific* code units. These have a standard function, structure and nomenclature (which uses the application option name `OptionName` as part of the unit name), and are described below:

(1) **T_*OptionName*_Definitions.f95**

Code unit providing default parameter values describing the basic attributes of the application option/equation of state (i.e., option name, number of components, number of phases, etc.). At the initiation of the simulation, the default parameter values supplied by this unit are compared to those provided by the user in the standard input file to ensure correct dimensioning in the dynamic memory allocation process (see Section 5.1).

(2) **T_*OptionName*_Properties.f95**

Code unit that describes the thermophysical properties and processes of phases and components other than those available in the core and supplemental TOUGH+ codes (i.e., the water and the real gas properties). For example, such a code unit would describe the oil properties or CO_2 properties in application options that would require such computations.

(3) T_*OptionName*_Specifics.f95

Code unit that includes procedures specific to the application option, such as the reading of EOS-specific inputs, the preparation of EOS-specific output files, the computation of mass and volume balances and the estimation of EOS-specific parameters. Generic procedures and operator extension – which override (*overload*) the standard procedures used by TOUGH+ for the simulation of non-

hydrate problems – are defined in this code unit, which does not include any procedures describing the equation of state involved in this application option.

(4) T_*OptionName*_EOS.f95

Code unit that describes the equation of state of the application option, assigns initial conditions, computes the thermophysical properties of the medium and of the phases (i.e., all the secondary variables), and determines phase changes and the state of the system. This code unit also includes the procedure that computes the elements of the Jacobian matrix for the Newton-Raphson iteration.

For example, the code units of the HYDRATE v1.5 application option [*Moridis*, 2014] are: **T_Hydrate_Definitions.f95**, **T_Hydrate_Properties.f95**, **T_Hydrate_Specifics.f95** and **T_Hydrate_EOS.f95**.

Additionally, TOUGH+ v1.5 is distributed with the MESHMAKER V1.5 code (also written in FORTRAN 95/2003), which used to be part of the main code in the TOUGH2 simulators [*Pruess et al*., 1999; 2012], but is a separate entity in the TOUGH+ family of codes. MESHMAKER is used for the space discretization (gridding) of the domain of the problem under study (see Section 7).

NOTE: In compiling TOUGH+ v1.5, *it is important that the free-format source code option be invoked for proper compilation of the FORTRAN 95/2003 code*.

4.0. Input Data Requirements and Structure

In this section, we discuss in detail the general input requirements for the TOUGH+ v1.5 simulations. These are the inputs that are common to any simulation, regardless of the application option that is being used. The input data for TOUGH+ v1.5 are a superset of the input data used in conventional TOUGH2 and older TOUGH+ simulations in order to ensure backward compatibility (a functional requirement for TOUGH+ v1.5). While most of the general inputs are similar in type, input format, and parameter representation to earlier versions of the code, new input data structures and advanced formats are used for the inputs for new capabilities that are unique to TOUGH+ v1.5, as well as for the inputs of the TOUGH+ application options. The introduction of advanced constructs and formats for the entire input data set is in progress, and these will be made available in future TOUGH+ releases.

4.1. Input Procedure

The input procedure in the current version of TOUGH+ remains similar in many aspects to that of TOUGH2 [*Pruess et al.*, 1999; 2012] and earlier versions of TOUGH+ [*Moridis et al*., 2008; 2009; 2012]. Input data can be provided in a flexible manner by means of one or several ASCII data files. Unless otherwise indicated, all TOUGH+ inputs are in standard metric (SI) units, such as meters, seconds, kilograms, ˚C and in the corresponding derived units, such as Newtons, Joules, Pascal (= N/m^2 for pressure), etc.

In the TOUGH+ standard input file, data are organized in data blocks that are defined by keywords. Quite often, only the first five characters of the keywords typed in columns 1-5 (see **Table 4.1**) are read, because these are sufficient to recognize the data block. While the contents of the various blocks are described in detail in Sections 5-12, here we describe some important records/keywords, and provide some general comments about their occurrence and arrangement in the input file.

The user is directed to the Appendix, where a sample input file is listed for his/her reference in the study of the input process.

4.1.1. Data Block/Keyword **`TITLE`**

The first record of the input file in any TOUGH+ simulation is **`TITLE`**, which includes a header of up to 132 characters. This record is necessary for any simulation to begin.

4.1.2. Keyword/Record **`ENDCY`**

A record with the **`ENDCY`** keyword typed in columns 1-5 closes the TOUGH+ input file and instructs the code to initiate the simulation.

4.1.3. Keyword/Record **ENDFI**

The presence of the **ENDFI** keyword in columns 1-5 is an alternative (to **ENDCY**) ending keyword in a TOUGH+ standard input file. Presence of **ENDFI** keyword causes the simulation to be skipped after printing basic input information. This is a useful option when the simulation is limited to an attempt to obtain some basic information of the properties and conditions of the system in its initial state.

4.1.4. Structure of TOUGH+ Standard Input Files

Every TOUGH+ v1.5 input file must (a) begin with the record/keyword **TITLE** and (b) end either with the record **ENDCY**, or, alternatively, with the record **ENDFI** (if no flow simulation is to be carried out). Data records beyond **ENDCY** (or **ENDFI**) are ignored.

Some data blocks, such as those specifying reservoir domains (**ROCKS or MEDIA**), volume elements (**ELEME**), connections (**CONNE**), and sinks/sources (**GENER**), have a variable number of records, while others have a fixed number of records. Unless otherwise indicated, a blank record indicates the end of the variable-length data blocks.

The data block **MEMORY** *must* follow the data block **TITLE** because it provides all the necessary information for the dynamic memory allocation and array dimensioning in TOUGH+ v1.5. The data blocks between the **MEMORY** and the **ENDCY**/ **ENDFI** keywords can be provided in arbitrary order, except for the data block **ELEME**, which must precede (if either is present) the blocks **CONNE** and **EXT-INCON**. The blocks **ELEME** and **CONNE** must either be both provided through the standard input file, or must both be absent, in which case alternative means for specifying geometry data are employed (see Section 7).

The data block **GENER** can be omitted if there are no sinks and sources in the problem. If the keyword **START** or **RANDO**MN is present (see Section 4.1.2), the data block **INCON** can be incomplete, with elements (grid blocks) in arbitrary order, or can be absent altogether.

Elements for which no initial conditions are specified in **INCON** are then assigned domain-specific initial conditions from (a) block **INDOM** (if present), or (b) from the data block **EXT-INCON** (if present), or (c) from the 'generic' initial conditions given in block **PARAM**, along with default porosities given in block **ROCKS**. If **START** or **RANDO**MN is not present, **INCON** must contain information for all elements, in exactly the same order as they are listed in block **ELEME**.

Between data blocks, the standard TOUGH+ input file may include an arbitrary number of records that do not begin with any of the TOUGH+ keywords. This is useful for inserting comments about problem specifications directly into the input file. TOUGH+ v1.5 gathers all these comments and prints the first 50 such records in the standard output file.

Much of the data handling in TOUGH+ is accomplished by means of disk files, which can be edited and modified using any text editor. The initialization of the arrays for geometry, generation, and initial condition data is always made from the disk files **MESH** (or **MINC**), **GENER**, and **INCON**. A user can either provide these files at execution time, or they can be written from TOUGH+ v1.5 input data during the initialization phase of the program, or they can be obtained from the standard TOUGH+ v1.5 simulation outputs at the end of a simulation for use in a continuation run. .

If the data blocks **GENER** and/or **INCON** are not provided in the standard input file, and if no disk files **GENER** and/or **INCON** are present, defaults take effect (no generation; initial conditions from block **INDOM**, or from block **EXT-INDOM**, or defaults from block **PARAM**). To ensure that these defaults are used and that erroneous inputs are not introduced, the disk files **GENER** and/or **INCON** *from a previous run* must be removed from the execution environment/directory. A safe way to use default generation and initial conditions is to specify *dummy* data blocks in the input file, consisting of one record with **GENER** or **INCON**, followed by a blank record.

The format of data blocks **ELEME**, **CONNE**, **GENER**, and **INCON** is basically the same (see Section 7) when these data are provided as disk files and when they are given as part of the input file. However, specification of these data through the input file rather than as disk files offers some added conveniences, which are useful when a new simulation problem is initiated. For example, a sequence of identical items (volume elements, connections, sinks or sources) can be specified through a single data record. Additionally, indices that are used internally for cross-referencing elements, connections, and sources are generated internally by TOUGH+ rather than having them provided by the user. The **INCON**, **GENER**, and **INCON** disk files written by TOUGH+ can be merged into an input file without changes, keeping the cross-referencing information.

Table 4.1. TOUGH+ v1.5 input data blocks

Keyword (+)	***Sec.***	***Function***
TITLE (1st record)	4.1.1	Data record (single line) with simulation title
MEMORY (2nd record)	5.1	Dynamic memory allocation
OptionName	(#)	Parameters describing properties and behavior of the specific application option
ROCKS or **MEDIA**	6.2	Hydrogeologic parameters for various reservoir domains
RPCAP or **WETTA**BILITY	6.3	Optional; parameters for relative permeability and capillary pressure functions
DIFFUSION	6.4	Optional; diffusivities of mass components
***ELEME**	7.1	List of grid blocks (volume elements)
***CONNE**	7.2	List of flow connections between grid blocks
INDOM	8.1	Optional; initial conditions for specific reservoir domains
***INCON**	8.2	Optional; list of initial conditions for specific grid blocks
EXT-INCON	8.3	Optional; list of initial conditions for specific grid blocks
BOUNDARIES	8.6	Optional; provides time-variable conditions at specific boundaries
***GENER**	9.1	Optional; list of mass or heat sinks and sources
PARAM	10.1	Computational parameters; time stepping and convergence parameters; program options
SOLVR	10.2	Optional; specifies parameters used by linear equation solvers.
TIMES	11.2	Optional; specification of times for generating printout
SUBDOMAINS	11.3	Optional; specifies grid subdomains for desired time series data
INTERFACES	11.4	Optional; specifies grid interfaces for desired time series data
SS_GROUPS	11.5	Optional; specifies sink/source groups for desired time series data
ENDCY (last record)	4.1.3	Record closes TOUGH+ input file and initiates simulation
ENDFI (last record)	4.1.4	Alternative for closing TOUGH+ input file which causes flow simulation to be skipped.

#: This described in the individual User's Manual of the TOUGH+ v1.5 application option
*: Data can be provided as separate disk files and omitted from input file.
+: The bold face part of the keyword (left column) suffices for data block recognition

Data describing time-variable boundary conditions can be entered in a tabular form in the data block **BOUNDARIES**. In addition to the output files that are produced by any TOUGH+ v1.5 simulation, TOUGH+ v1.5 provides the option of output files with time-series data on variables that describe (a) the status of selected subdomains of the simulated domain and (b) flows through user-defined interfaces within the domain and/or through groups of sources and sinks (wells). The input data needed for the definition of these subdomains, interfaces and well groups are provided through the data blocks **SUBDOMAINS**, **INTERFACES** and **SS_GROUPS**, respectively (see **Table 4.1**). For *continuation runs*, the presence of the output files corresponding to the **SUBDOMAINS**, **INTERFACES** and **SS_GROUPS** data blocks – created during the last simulation that is to be continued – is of critical importance because they contain information needed by the TOUGH+ code for the seamless extension of the simulation period.

PAGE LEFT INTENTIONALLY BLANK

5.0. Memory Specification and Allocation

5.1. Data Block MEMORY

This is the data block that reads the data for the dynamic memory allocation in the TOUGH+ simulation. This block must always follow the **TITLE** record.

Record MEMORY.1

This record must always include the following header:

```
MEMORY
```

Record MEMORY.2

Reads the character variable EOS_Name according to FORMAT(A15). This variable describes the Equation Of State (EOS) that describes the application option simulated by TOUGH+ in the problem at hand. This parameter is used strictly to allocate memory to the pertinent storage arrays and to compare the user-specified values of the parameters provided in the next record (Record MEMORY.3) against defaults specified in the code.

Record MEMORY.3

The following information is provided in MEMORY.3 using a free format:

```
NumCom, NumEq, NumPhases, binary_diffusion
```

These parameters are defined as follows:

NumCom	Integer denoting the number of mass components (see Section 2)
NumEq	Integer denoting the number of equations
NumPhases	Integer denoting the number of phases.
binary_diffusion	Logical variable indicating whether binary diffusion is active (binary_diffusion=.TRUE.) or ignored (binary_diffusion=.FALSE.).

The permissible combinations of the values of these parameters are discussed in details in the User Manuals of each of the TOUGH+ application options.

Record MEMORY.4

In MEMORY.4, the following parameters are read using a free format:

```
coordinate_system, Max_NumElem,
Max_NumConx, ElemNameLength,
active_conx_only, boundaries_in_matrix
```

These parameters are defined as follows:

coordinate_system
: Character variable describing the coordinate system used in the study. It can assume the values 'Cartesian' or 'Cylindrical'.

Max_NumElem
: Integer variable defining the maximum number of elements (cells, gridblocks) in the discretized simulation domain

Max_NumConx
: Integer variable defining the maximum number of connections in the discretized simulation domain.

Information on the concepts of elements and connections can be found in Section 8, and in *Pruess et al.* [1999]. As a general rule, `MaxNum_Conx > ND*MaxNum_Elem`, where ND is the dimensionality of the problem.

`ElemNameLength`

Integer variable defining the number of characters in the element names. It may be either 5 or 8. The default in TOUGH+ v1.5.

`active_conx_only`

Logical variable indicating whether the simulation will be halted after determining the active connections in the grid. This feature is useful when running a simulation that uses a subset of the elements of a large grid without correspondingly adjusting the connections, and is designed to reduce the very large memory requirements for the connection-related dynamic arrays.

When `active_conx_only =.TRUE.`, the simulation stops once the active connections (involving only the elements defined in the element list) are determined. The active connections are stored in a new file called **`Active_Connection_File`**. Then the simulation can be run using the new connection list, thus having much lower memory requirements. For a thorough discussion of elements and connections in TOUGH+, see Section 7.

`boundaries_in_matrix`

Logical variable indicating how inactive elements (describing constant-conditions boundaries, see Section 8.5, and denoted either by a negative or very large volume, or by setting the variable `elem_activity='I '`, see Section 7.2) are to be treated when solving the equations of mass and energy balance.

When `boundaries_in_matrix =.TRUE.`, then all inactive elements are assigned very large volumes (10^{50} m^3) to maintain constant conditions during the simulation, and are included in the Jacobian matrix. Otherwise, only the active elements are included in the Jacobian.

This feature is useful when using older TOUGH2 input files, and when a parallel version of the code is employed. With newly developed input files, the `boundaries_in_matrix =.FALSE.` value is highly recommended.

Record `MEMORY.5`

The integer variable `Max_NumSS` (declaring the maximum number of expected sources and sinks) is read using a free format.

Record MEMORY.6

In this record, the integer variable Max_NumMedia is read using a free format. This variable represents the maximum number of geologic media with different properties to be considered in the simulations.

Record MEMORY.7

In MEMORY.7, the following logical variables are read using a free format:

```
element_by_element_properties,
porosity_perm_dependence,
scaled_capillary_pressure,
Option_tortuosity_CompuMethod
```

These parameters are defined as follows:

element_by_element_properties

Logical variable (flag) indicating whether each gridblock has its own hydraulic properties (ϕ and k), in which case they are read on an element-by-element basis (see Sections 7 and 8). This feature is necessary (a) in the simulation of very heterogeneous systems, and/or (b) when k changes in response to pressure and/or temperature variations, or in response to changes in the geomechanical regime of the system.

When element_by_element_properties =.FALSE., then the ϕ and k of a particular element are determined from the general properties of the corresponding porous medium (see Section 6). When element_by_element_properties =.TRUE., then element-specific ϕ and k are read as part of the initial conditions in the **INCON** data block (see Section 8).

porosity_perm_dependence

Logical variable (flag) indicating whether the intrinsic permeability k in a given element is to change as a function of changing porosity ϕ. As discussed earlier, ϕ can change in response to P and/or T changes according to relationships that are determined from simple or complex geomechanical models (see Section 2).

When porosity_perm_dependence =.FALSE., then k is unaffected by changes in ϕ. When porosity_perm_dependence =.TRUE., then k is readjusted internally to reflect the effect of changes in ϕ that are estimated using either an empirical model (see Equation 2.51, Section 2) or a full geomechanical model.

Note that when `porosity_perm_dependence =.TRUE.`, the variable `element_by_element_properties` is set internally to `.TRUE.` because activation of the `porosity_perm_dependence` feature results in element-specific hydraulic properties.

`scaled_capillary_pressure`

Logical variable (flag) indicating whether the capillary pressure P_{cap} will be scaled to reflect variations in ϕ and k.

Activation of this feature by setting `scaled_capillary_pressure =.TRUE.` may be needed (a) in highly heterogeneous systems in which the element-specific properties vary significantly from those described by the average (expected) values of the porous medium (as specified in Section 6), and/or (b) when the significant variations in ϕ and k are experienced in the course of the simulation (e.g., when `porosity_perm_dependence =.TRUE.`).

`Option_tortuosity_CompuMethod`

A character variable describing the method of estimation of the binary **gas** diffusivities. This input variable is important if one or more of the following processes are considered in the simulation: turbulent flow, diffusion, solute or colloid transport, a Dusty Gas Model [*Mason and Malinauskas*, 1983; *Webb*, 1998]; otherwise it is ignored. The following options are available:

`='Relative_Perm'`: For domains for which a tortuosity parameter τ_0 = `mediaTortu` $\neq 0$ is specified in data block **`MEDIA`** or **`ROCKS`** (see Section 6.1), $\tau_\beta = \tau_\beta\left(S_\beta\right) = k_{r\beta}$ – see Equations (2.15) and (2.17).

`='Saturation'`: For domains for which a tortuosity parameter τ_0 = `mediaTortu` $\neq 0$ is specified in data block **`MEDIA`** or **`ROCKS`** (see Section 6.1), $\tau_\beta = \tau_\beta\left(S_\beta\right) = S_\beta$ – see Equations (2.15) and (2.17).

`='Constant'`: When this option is invoked, the constant τ_0 = `mediaTortu` values provided in the data block **`MEDIA`** or **`ROCKS`** (Section 6.1) is used with no phase-saturation adjustment, i.e., $\tau_\beta = 1$.

NOTE: If `mediaTortu=0` in data block **`MEDIA`** or **`ROCKS`**, then the tortuosity τ_0 is computed from the Millington-Quirk model in Equation (2.21) as $\tau_{T\beta} = \tau_0 \tau_\beta = \phi^{1/3} S_\beta^{10/3}$. This value is then used in the computation of the effective diffusion coefficients.

Record MEMORY.8

In MEMORY.8, the following logical variables are read using a free format:

```
coupled_geochemistry, property_update
```

These parameters are defined as follows:

coupled_geochemistry
: Logical variable (flag) indicating whether the simulation involves coupled flow, thermal and geochemical processes. This feature is activated by setting coupled_geochemistry =.TRUE. when geochemical processes are considered in a TOUGH+ simulation involving coupled flow, thermal and geochemical processes. The default value is coupled_geochemistry =.FALSE..

property_update
: Character variable (flag) indicating the type of property update when coupled geochemical processes are involved in TOUGH+ simulations. This variable is the same as in the next record (MEMORY.9), where it is discussed in detail.

Record MEMORY.9

In MEMORY.9, the following logical variables are read using a free format:

```
coupled_geomechanics,
geomechanical_code_name,
property_update, num_geomech_param
```

These parameters are defined as follows:

coupled_geomechanics
: Logical variable (flag) indicating whether the simulation involves coupled flow, thermal and geomechanical processes. Activation of this feature by setting coupled_geomechanics =.TRUE. indicates the use of a complex geomechanical model to describe the relationship between hydraulic media properties (ϕ and k) and geomechanical parameters (such as stresses and strains). This complex geomechanical model overrides the simplified models (based on pore compressibility and expansivity) that are standard in TOUGH+ (see Section 2).

  When this flag is invoked, the geomechanical model invoked by the TOUGH+ code is that defined by the variable geomechanical_code_name (see discussion of the next parameter).

When coupled_geomechanics =.FALSE. (the default value), the simplified geomechanical models are invoked, even if the executable of a geomechanical code (named geomechanical_code_name) is present in the TOUGH+ directory.

When coupled_geomechanics =.TRUE., and the executable of a geomechanical code named geomechanical_code_name is present in the TOUGH+ directory, the following files are created: (a) **To_GMech**, containing the data (pressure, temperatures and phase saturations) that are provided to the **FLAC3D** geomechanical code [*Itasca*, 2002] from the TOUGH+ simulator for use in the computation of the geomechanical properties of the system, and (b) the **Fr_Gmech** file, containing the data (mainly stresses and strains) supplied by the **FLAC3D** code for use by the TOUGH+ simulator for the computation of the variable (geomechanically-dependent) hydraulic properties ϕ and k. These two files are necessary for communication between the two codes, because the lack of shared memory (disallowed because of intellectual property concerns) makes data exchange by means of these two external files as the only viable option. Additionally, the new file **Init_Stress** that stores the initial stresses and strains is created if this is a new run; in a continuation run, the initial (i.e., at t = 0, not at the time of the initiation of the continuation run) stresses and strains are read from the old file **In_Stresses**.

geomechanical_code_name

A character variable of maximum length 6 that provides the name of the executable of the geomechanical code that is coupled with the TOUGH+ code when coupled_geomechanics =.TRUE.. In the current implementation of the code, the geomechanical model used in conjunction with the common TOUGH+ application options is the **FLAC3D** commercial simulator (ITASCA, 2002), i.e., geomechanical_code_name = **FLAC3D**. However, the TOUGH+ can easily accommodate any other geomechanical simulator that conforms to its data exchange formats and requirements.

When coupled_geomechanics =.TRUE. and the executable of a geomechanical code named geomechanical_code_name is present in the TOUGH+ directory, it is treated as a C subroutine that is called by the TOUGH+ simulator. If (a) coupled_geomechanics =.TRUE. and (b) the geomechanical_code_name is different from **FLAC3D** , or does not correspond to any executable available in the TOUGH+ directory, or is blank, the TOUGH+ code resets coupled_geomechanics to =.FALSE..

property_update

Character variable (flag) denoting the manner of property update as a result of interdependent changes in the hydraulic (flow) and geomechanical

properties. The variable `property_update` can assume the following values: `'Continuous'`, indicating continuous property update (i.e., in every Newtonian iteration of every timestep) and participation in the Jacobian matrix; `'Iteration'`, indicating property updates in every Newtonian iteration of every timestep, but without any contribution to the Jacobian; and `'Timestep'`, indicating a single property update at the end of each timestep and no contribution to the Jacobian. The option `property_update` = `'Continuous'` yields the most accurate solutions that are accurate over any pressure range and media geomechanical property range but results in longer execution times, while the option `property_update` = `'Timestep'` leads to faster solutions, but which are acceptably accurate in less compressible media and for a mild ΔP.

If a `property_update` value other than the three described above is read, then an error message is printed and the simulation is aborted.

`NumGeomechParam`

Integer variable defining the number of geomechanical parameters that are to be provided by the geomechanical code (to be read by the TOUGH+ simulator from the **`Fr_Gmech`** file), and which will be used to estimate the updated ϕ and k. Geomechanical codes such as **`FLAC3D`** can provide a very large number of geomechanical properties and parameters of interest, but in the current version of the TOUGH+ only two (stress and strain) are needed, and these are stored separetely. Thus, `NumGeomechParam` is not needed and can be set to zero (indicating no additional memory utilization) when the **`FLAC3D`** geomechanical model is invoked for hydrate simulations that involve coupled geomechanical effects. However, the parameter is available for generality and maximum flexibility in future code developments that may include the use of additional geomechanical simulators and constitutive equations.

5.2. Internal Checks

The data provided in Record `MEMORY.3` (`NumCom`, `NumElem` and `NumPhases`) define the dynamic array dimensioning of the arrays that describe all the primary and secondary variables in TOUGH+ v1.5 simulations. These are compared to the default value ranges of the same parameters for the specific TOUGH+ application option (EOS) that are provided in the code unit **`T_`***`OptionName`***`_Definitions.f95`** (see Section

3.6). If any of these parameter values fall outside the acceptable range, this is a grave (unrecoverable) error; a detailed error message is printed into the standard output file and the simulation ceases.

Similarly, an error message is printed and the simulation is aborted if the default application option (EOS) name name(s) in **T_*OptionName*_Definitions.f95** heading of the *OptionName* data block (see Table 4.1) that provides the application option (EOS) name is in conflict with the.

PAGE LEFT INTENTIONALLY BLANK

6.0. Physical Properties of System

This section lists and describes the physical properties of the system that are common to any TOUGH+ simulation, including the rock properties (block **ROCKS** or **MEDIA**), the relative permeability and capillary pressure properties (block **RPCAP**), and the properties specifying multi-component diffusion (block **DIFFU**). Additionally, it describes the procedure for introducing block-by-block permeability modification.

6.1. Data Block ROCKS or MEDIA

This block introduces material parameters (flow and thermal) for all the different geologic media (porous or fractured) involved in the domain simulated by the appropriate TOUGH v1.5 application.

Record ROCKS.1

Format (A5, I5, 8E10.4)

```
mediaName, NAD, mediaDensG,
mediaPoros, (mediaPerm(i), i = 1,3),
mediaKThrW, mediaSpcHt, PoMedRGrain
```

`mediaName`
Material name (rock type).

`NAD`
Integer variable; if zero or negative, defaults will take effect for a number of parameters (see below);

≥ 1: will read another data record to override defaults.

≥ 2 and < 5: will read two more records with domain-specific parameters for relative permeability and capillary pressure functions.

= 5: In addition to the four records read for NAD > 2, an additional record will be read with the coefficients of the porosity polynomial $\phi/\phi_0 = F_0 + F_1\Delta P + F_2\Delta P^2 + ... + F_n\Delta P^n$, where ϕ_0 is the reference (initial) default porosity and $\Delta P = P\text{-}P_0$ is the deviation from the initial pressure P_0. This equation will be used instead of Equation (2.47) to estimate the effect of pressure on the medium porosity.

= 6: In addition to the four records read for NAD > 2, an additional record will be read with the coefficients of Equations (2.56) and (2.57) that describe the compressibility of an unconsolidated porous medium in the presence of cementing solid phases (such as ice and/or hydrates).

= 8: In addition to the four records read for NAD > 2, an additional record will be read with information on the specifics of inertial or slippage effects on fluid flow through this medium.

`mediaDensG`
Rock grain density [kg/m^3]

`mediaPoros`
Default porosity ϕ_0 (void fraction) for all elements belonging to domain `MediumName` for which no other porosity has been specified in block **`INCON`**. Option "**`START`**" is necessary for using the default porosity.

`mediaPerm(i),i=1,…,3`
Absolute permeabilities along the three principal axes, as specified by `ConxKi` in block **`CONNE`**.

`mediaKThrW`
Formation heat conductivity under fully liquid-saturated conditions [W/m/˚C].

`mediaSpcHt`
Rock grain specific heat [J/kg/˚C].

`PoMedRGrain`
Rock grain radius [m]. This is needed for the estimation of the surface reaction area in some TOUGH+ v1.5 application options involving kinetic chemical reactions. If `PoMedRGrain = 0.0e0` (e.g., when no value is provided), the TOUGH+ v1.5 code provides a grain radius estimate using the Kozeny-Carman approximation. This variable is needed in very specific TOUGH+ v1.5 applications options, and is discussed in the corresponding User's Manuals.

Record `ROCKS.1.1` (optional, when `NAD` ≥ 1 only)

Format (11E10.4)

```
mediaCompr, mediaExpan, mediaKThrD,
mediaTortu, mediaKlink, mediaOrgCF,
mediaCritSat, mediaPermExpon, mediaBeta,
mediaGama, PhiZeroStress
```

`mediaCompr`
Pore compressibility $\alpha_P = (1/\phi)(\partial\phi/\partial P)_T$ [Pa^{-1}], see Equation (2.47) – default $\alpha_P = 0.0$.

`mediaExpan`
Pore expansivity $\alpha_T = (1/\phi)(\partial\phi/\partial T)_P$ [1/ ˚C], see Equation (2.47) – default $\alpha_T = 0.0$.

`mediaKThrD` Formation heat conductivity under desaturated conditions [W $m^{-1}K^{-1}$] – default is `mediaKThrD = mediaKThrW`.

`mediaTortu`
Tortuosity factor for binary diffusion. If `mediaTortu = 0`, a porosity and saturation-dependent tortuosity will be calculated internally from the *Millington and Quirk* [1961] model, Equation (2.53).

`mediaKlink`
Klinkenberg parameter b [Pa^{-1}] for enhancing gas phase permeability according to the relationship of Eaution (2.13) – default is 0.

`mediaOrgCF`

Not used.

`mediaCritSat`

Critical total mobile phase saturation at which the permeability of the medium that experience precipitation of solids becomes equal to zero; it is equal to the critical "open" porosity ϕ_c of a porous medium at which its permeability becomes zero – needed only when the EPM model is invoked (see Sections 2.8 and 2.10).

`mediaPermExpon`

Permeability reduction exponent for solid phase-bearing systems – needed only when the EPM model is invoked, see see Sections 2.8 and 2.10, Equation (2.43).

`mediaBeta`

The parameter β used for the computation of porosity as a function of geomechanical stresses σ according to the equation:

$$\phi = \phi_0 - \phi_{\sigma=0} \{\exp[\beta(\sigma - P)] - \exp[\beta(\sigma_o - P_0)]\}$$

where $\phi_{\sigma=0}$ is the medium porosity at zero stress, and the subscript *0* indicates initial conditions.

NOTE: This ϕ computational option can be invoked only when all of the following conditions are met: (a) the option `coupled_geomechanics =.TRUE.` is activated, i.e., when the TOUGH+ simulations are coupled with a full geomechanical model such as the **`FLAC3D`** code, (b) at least one medium $\beta \neq 0$, and (c) all media pore compressibilities $\alpha_P = 0$.

If `coupled_geomechanics =.FALSE.`, this parameter is ignored and the variations in ϕ are computed from one of the other computational options available in TOUGH+. The same occurs when `coupled_geomechanics =.TRUE.`, all media $\beta = 0$, and at the pore compressibility of at least one medium $\alpha_P \neq 0$.

`mediaGama`

The parameter γ used for the computation of intrinsic permeability k as an empirical function of variations in the porosity ϕ – See Equation (2.24).

`PhiZeroStress`

The porosity at zero stress $\phi_{\sigma=0}$. This variable is used only when the option `coupled_geomechanics =.TRUE.` is activated, i.e., when the TOUGH+ simulations are coupled with a full geomechanical model such as the **`FLAC3D`** code. If `coupled_geomechanics =.FALSE.`, this variable is ignored.

Record `ROCKS.1.2` (optional, `NAD` ≥ 2 only)

Format (I5, 5X, 7E10.4)

```
RelPermEquationNum,          (RelPermParam(i),i=
1,…,7)
```

`RelPermEquationNum`
Integer parameter indicating the type of the relative permeability function of the medium under consideration (see detailed discussion in Section 6.3.1).

`RelPermParam(i),i=1,…,7`
Real parameters corresponding to the relative permeability function described by the `RelPermEquationNum` option (see Section 6.3.1).

Record `ROCKS.1.3` (optional, NAD ≥ 2 only)

Format (I5, 5X, 7E10.4)

```
PcapEquationNum, (PcapParam(i),i=1,…7)
```

`PcapEquationNum`
Integer parameter indicating the type of the capillary pressure function of the medium under consideration (see detailed discussion in Section 6.3.2).

`PcapParam(i),i=1,…7`
Real parameters corresponding to the capillary pressure function described by the `PcapEquationNum` option (see Section 6.3.2).

Record `ROCKS.1.4` (optional, NAD = 5 only, to be used when the media porosity is described as a polynomial function of the pressure change ΔP)

Format (I5, 5X, 7E20.13)

```
PhiPolyOrder, (PhiCoeff (i),i=0,…,6)
```

`PhiPolyOrder`
Order n of the polynomial $\phi/\phi_0 = F_0 + F_1\Delta P + F_2\Delta P^2 + \ldots + F_n\Delta P^n$. For a constant ϕ, `PhiPolyOrder = 0.`

`PhiCoeff(i),i=0,…,6`
Coefficients F_n (n = 0,…,`PhiPolyOrder`) of the $\phi = \phi(\Delta P)$ polynomial.

Record `ROCKS.1.4` (optional, NAD = 6 only, to be used when cementing solid phases such as ice and/or hydrates are present in the pores of unconsolidated media – see Section 2.11.3)

Format (10E10.4)

```
LoComp, SatAtLoComp,
HiComp, SatAtHiComp, DeltaSat
```

`LoComp`

The lower limit of the medium compressibility α_{PL} [Pa^{-1}], corresponding to the full stiffening/strengthening effect of the presence of cementing solid phases such as ice and/or hydrates – see Equation (2.56).

`SatAtLoComp`

= S_{Smax}, i.e., the lowest S_S saturation at which $\alpha_P = \alpha_{PL}$ – see Equations (2.44) and (2.45).

`HiComp`

The upper limit of the medium compressibility α_{PU} [Pa^{-1}], corresponding to the absence of cementing solid phases– see Equation (2.44).

`SatAtHiComp`

= S_{Smin}, i.e., the largest S_S saturation at which $\alpha_P = \alpha_{PU}$ – see Equations (2.44) and (2.5).

`DeltaSat`

The smoothing factor δ – see Equation (2.45). A value of $\delta = 0.015$ is suggested – see Figure 2.5.

Repeat records `ROCKS.1`, `ROCKS.1.1`, `ROCKS.1.2`, `ROCKS.1.3` and `ROCKS.1.4` for all the porous/fractured media in the domain under investigation.

Record `ROCKS.2`

A blank record closes the ROCKS data block.

Note: *The number of media described in the data block* **`ROCKS`**/**`MEDIA`** *cannot exceed the number* `Max_NumMedia` *specified in the* `MEMORY` *data block (See Section 5.1). If this happens, an error message is printed and the simulation is aborted.*

6.3. Data Block RPCAP

This block introduces information on relative permeability and capillary pressure functions, which will be applied for all flow domains for which no data were specified in records ROCKS.1.2 and ROCKS.1.3. A catalog of relative permeability and capillary pressure functions is presented in Sections 6.3.1 and 6.3.2, respectively.

Record RPCAP.1

Format (I5, 5X, 7E10.4)

```
DefaultRelPermType, (RPD(i),i=1,7)
```

DefaultRelPermType
: Integer parameter describing the type of the default relative permeability function (see Section 6.3.1).

DefaultRelPermType, (RPD(i),i=1,7)
: Real parameters corresponding to the relative permeability function selected by the DefaultRelPermType parameter (see Section 6.3.1).

Record RPCAP.2

Format (I5, 5X, 7E10.4)

```
DefaultCapPresType, (CPD(I),i=1,7)
```

DefaultCapPresType
: Integer parameter describing the type of the default capillary pressure function (see Section 6.3.2).

(CPD(i), i=1,…,7)
: Real parameters corresponding to the capillary pressure function selected by the DefaultCapPresType parameter (see Section 6.3.2).

All the relative permeability and capillary pressure functions that are available in TOUGH+ for use in the various application options are listed in the following sections.

6.3.1. Two-Phase Relative Permeability Functions

6.3.1.1. `RelPermEquationNum = 1:` **Linear functions**
k_{rA} increases linearly from 0 to 1 in the range `RP(1)` $\le S_A \le$ `RP(3)`;
k_{rG} increases linearly from 0 to 1 in the range `RP(2)` $\le S_G \le$ `RP(4)`
Restrictions: `RP(3) > RP(1);` `RP(4) > RP(2).`

6.3.1.2. `RelPermEquationNum = 2:` **Power functions**
$k_{rA} = \left(S_A\right)^n$, $k_{rG} = 1$, where n = `RP(1)`

6.3.1.3. `RelPermEquationNum = 3:` **Corey's curves** [*Corey*,1954]

$$k_{rA} = \hat{S}^4, \qquad k_{rG} = \left(1-\hat{S}\right)^2\left(1-\hat{S}^2\right)$$

where $$\hat{S} = \frac{\left(S_A - S_{irA}\right)}{\left(1 - S_{irA} - S_{irG}\right)}$$

with S_{irA} = `RP(1);` S_{irG} = `RP(2)`
Restrictions: `RP(1) + RP(2) < 1`

6.3.1.4. `RelPermEquationNum = 4:` **Grant's curves** [*Grant*, 1977]

$$k_{rA} = \hat{S}^4, \qquad k_{rG} = 1 - k_{rA}$$

where $$\hat{S} = \frac{\left(S_A - S_{irA}\right)}{\left(1 - S_{irA} - S_{irG}\right)}$$

with S_{irA} = `RP(1);` S_{irG} = `RP(2)`
Restrictions: `RP(1) + RP(2) < 1`

6.3.1.5. `RelPermEquationNum = 5:` **All phases perfectly mobile**
$k_{rG} = k_{rA} = 1$ for all saturations; no parameters

6.3.1.6. `RelPermEquationNum = 6:` **Functions of Fatt and Klikoff** (1959)

$$k_{rA} = \left(S^*\right)^3, \qquad k_{rG} = \left(1 - S^*\right)^3$$

where: $S^* = \dfrac{(S_A - S_{irA})}{(1 - S_{irA})}$

with S_{irA} = `RP(1)`.

Restriction: `RP(1) < 1`.

6.3.1.7. `RelPermEquationNum = 7, -7:` **van Genuchten-Mualem model** [*Mualem*, 1976; *van Genuchten*, 1980]

$$k_{rA} = \begin{cases} \sqrt{S^*}\left\{1-\left(1-[S^*]^{1/\lambda}\right)^{\lambda}\right\}^2 & \text{if } S_A < S_{mxA} \\ 1 & \text{if } S_A \geq S_{mxA} \end{cases}$$

Gas relative permeability can be chosen from among several options. For `RelPermEquationNum = 7`, it is computed from one of the following three forms, of which the second is the *Corey* [1954] equation and the third is the modified Stone equation [*Stone*, 1970] – see Section 6.1.3.9:

$$k_{rG} = \begin{cases} 1 - k_{rG} & \text{if } S_{irG} = 0 \\ \left(1-\hat{S}\right)^2\left(1-\hat{S}^2\right) & \text{if } S_{irG} > 0 \text{ and } n_G \leq 0 \\ \left(\tilde{S}\right)^{n_G} & \text{if } S_{irG} > 0 \text{ and } n_G > 0 \end{cases}$$

For `RelPermEquationNum =-7`, the gas relative permeability is computed from the model of *Parker et al.* [1987]:

$$k_{rG} = \begin{cases} \sqrt{1-S^*}\left\{\left[1-\left(S^*\right)^{1/\lambda}\right]^{\lambda}\right\}^2 & \text{if } 1-S^* > 0 \\ 0 & \text{if } 1-S^* \leq 0 \end{cases}$$

The k_{rA} and k_{rG} estimates are subject to the following restrictions:

$$0 \leq k_{rA}, k_{rG} \leq 1$$

Here, $S^* = \dfrac{S_A - S_{irA}}{S_{mxA} - S_{irA}}$ and $\tilde{S} = \dfrac{S_G - S_{irG}}{1 - S_{irA}}$

Parameters: `RP(1)` = λ
`RP(2)` = S_{irA}
`RP(3)` = S_{mxA}
`RP(4)` = S_{irG}
`RP(5)` = n_G

Notation: Parameter λ is m in van Genuchten's notation, with $m = 1 - 1/n$.

6.3.1.8. `RelPermEquationNum = 8:` **Function of Verma et al.** [1985]

$$k_{rA} = \hat{S}^3, \qquad k_{rG} = A + B\hat{S} + C\hat{S}^2$$

where $$\hat{S} = \frac{(S_A - S_{irA})}{(S_{mxA} - S_{irA})}$$

Parameters as measured by *Verma et al*. [1985] for steam-water flow in an unconsolidated sand:

S_{irA}	=	`RP(1) = 0.2`
S_{mxA}	=	`RP(2) = 0.895`
A	=	`RP(3) = 1.259`
B	=	`RP(4) = -1.7615`
C	=	`RP(5) = 0.5089`

6.3.1.9. `RelPermEquationNum = 9:` Modified version of Stone's first three-phase relative permeability method [*Stone*, 1970].

$$k_{rA} = \max\left\{0, \min\left\{\left[\frac{S_A - S_{irA}}{1 - S_{irA}}\right]^n, 1\right\}\right\},$$

$$k_{rG} = \max\left\{0, \min\left\{\left[\frac{S_G - S_{irG}}{1 - S_{irA}}\right]^{n_G}, 1\right\}\right\},$$

$$k_{rH} = 0$$

Parameters are S_{irA} = `RP(1)`, S_{irG} = `RP(2)`, n = `RP(3)`, n_G = `RP(4)`.
Note: When `RP(4)=0.0e0`, it is reset internally to `RP(4)=RP(3)` .

If the user has access to the source code, he/she may modify the source code of TOUGH+ (segment **`T_Media_Properties.f95`**) to include other relative permeability relationships. However, for this task the user needs to be familiar with the concepts of generic processes (and overloading) in object oriented programming languages.

6.3.2. Two-Phase Capillary Pressure Functions

6.3.2.1. `PcapEquationNum = 1:` **Linear function**

$$P_{cap} = \begin{cases} -\mathrm{CP}(1) & \text{for } S_A \le \mathrm{CP}(2) \\ 0 & \text{for } S_A \le \mathrm{CP}(2) \\ -\mathrm{CP}(1)\dfrac{\mathrm{CP}(3)-S_A}{\mathrm{CP}(3)-\mathrm{CP}(2)} & \text{for } \mathrm{CP}(2) < S_A < \mathrm{CP}(3) \end{cases}$$

Restriction: `CP(3)` > `CP(2)`

6.3.2.2. `PcapEquationNum = 2:` **Function of *Pickens et al.*** [1979]

$$P_{cap} = -P_0 \left\{ \ln\left[\frac{A}{B}\left(1+\sqrt{1-B^2/A^2}\right)\right]\right\}^{1/x}$$

with

$$A = \left(1+\frac{S_A}{S_{A0}}\right)\frac{\left(S_{A0}-S_{irA}\right)}{\left(S_{A0}+S_{irA}\right)}, \qquad B = 1-\frac{S_A}{S_{A0}}$$

where P_0 = `CP(1)`, S_{irA} = `CP(2)`, S_{A0} = `CP(3)`, x = `CP(4)`

Restrictions: `0 < CP(2) < 1 ≤ CP(3)`; `CP(4)` ≠ `0`

6.3.2.3. `PcapEquationNum = 3:` **TRUST capillary pressure** [*Narasimhan et al.*, 1978]

$$P_{cap} = \begin{cases} -P_e - P_0\left[\dfrac{1-S_A}{S_A - S_{irA}}\right]^{1/\eta} & \text{for } S_A < 1 \\ 0 & \text{for } S_A < 1 \end{cases}$$

where P_0 = `CP(1)`, S_{irA} = `CP(2)`, η = `CP(3)`, P_e = `CP(4)`

Restrictions: `CP(2)` ≥ 0; `CP(3)` $\neq 0$

6.3.2.4. `PcapEquationNum = 4:` **Milly's function** [*Milly*, 1982]

$$P_{cap} = -97.783 \times 10^{A}, \quad A = 2.26\left(\frac{0.371}{S_A - S_{irA}} - 1\right)^{1/4}$$

where S_{irA} = `CP(1)`

Restriction: `CP(1)` ≥ 0

6.3.2.5. `PcapEquationNum = 6:` **Leverett's function** [*Leverett*, 1941; *Udell and Fitch*, 1985]

$$P_{cap} = -P_0 \cdot \sigma(T) \cdot f(S_A)$$

where

$\sigma(T)$: surface tension of water (supplied internally in TOUGH+)

$$f(S_A) = 1.417\left(1 - S^*\right) - 2.120\left(1 - S^*\right)^2 + 1.263\left(1 - S^*\right)^3$$

$$S^* = \frac{\left(S_A - S_{irA}\right)}{\left(1 - S_{irA}\right)}$$

Parameters: P_0 = `CP(1)`, S_{irA} = `CP(2)`

Restriction: $0 \leq$ `CP(2)` < 1

6.3.2.6. `PcapEquationNum = 7:` **van Genuchten function** [*van Genuchten*, 1980]

$$P_{cap} = -P_0\left[\left(S^*\right)^{-1/\lambda} - 1\right]^{1-\lambda}, \quad S^* = \frac{\left(S_A - S_{irA}\right)}{\left(S_{mxA} - S_{irA}\right)}$$

subject to the restriction: $-P_{\max} \le P_{cap} \le 0$

Parameters: `CP(1)` $= \lambda = 1 - 1/n$

`CP(2)` $= S_{irA}$ (should be chosen smaller than the corresponding parameter in the relative permeability function; see note below.)

`CP(3)` $= 1/P_0 = \alpha/\rho_w g$

`CP(4)` $= P_{max}$

`CP(5)` $= S_{mxA}$

Notation: Parameter λ is m in van Genuchten's notation, with $m = 1 - 1/n$.

Note on parameter choices: In the van Genuchten's derivation [1980] of the capillary pressure equation, the parameter S_{irA} for irreducible water saturation is the same in the relative permeability and capillary pressure functions. As a consequence, for $S_A \to S_{irA}$ we have $k_{rA} \to 0$ and $P_{cap} \to -\infty$, which is unphysical because it implies that the radii of capillary menisci go to zero as liquid phase is becoming immobile (discontinuous). In reality, no special capillary pressure effects are expected when liquid phase becomes discontinuous. Accordingly, we recommend to always define a smaller S_{irA} for the capillary pressure as compared to that for the relative permeability function.

6.3.2.7. `PcapEquationNum = 8:` ***Brooks-Corey*** **[1964] equation modified to account for effect of solid phases on capillary pressure**

$$P_{cap} = -F \cdot G \cdot P_{GE}(S^*)^v, \; S^* = \frac{\left(S_A - S_{irA}\right)}{\left(1 - S_{irA}\right)}, \; F = 1 + A \cdot Bx(a,b,S_H)$$

where

v = exponent with the following restrictions: v< 0 and |v| ≤ 1;
P_{GE} = gas entry pressure;

G = error function equation that smoothes curve near S=0;
F = factor that describes effect of hydrate on capillary pressure;
A = parameter > 0;
B_x = incomplete beta function;
a,b = input arguments for B_x;
S_S = sum of solid saturations.

Parameters: `CP(1)` = P_{GE}, `CP(2)` = v, `CP(3)` = S_{irA},
`CP(4)` = $P_{cap,max}$, `CP(5)` = A, `CP(6)` = a, `CP(7)` = b

6.3.2.8. `PcapEquationNum = 9:` No capillary pressure

$P_{cap} \equiv 0$ for all saturations; no parameters.

6.4. Data Block `DIFFUSION`

This block reads multicomponent diffusion coefficients using a `NAMELIST` format. This is a very powerful format that allows maximum clarity and flexibility, accepting free formats, arbitrary ordering of variables, insertions of comments anywhere in the input fields, and providing the option of ignoring any of the `NAMELIST` parameters by not assigning a value to it. For more information, the reader is directed to a textbook on FORTRAN 95/2003.

Record `DIFFUSION.1`

This record includes general data describing key diffusion parameters. The namelist in this record is named `Diffusion_Key_Parameters`, and has the following general form.

```
&Diffusion_Key_Parameters
     gas_diffusivity_equation_exponent  = x.xEx,
     P_at_RefDiffusivity                = x.xEx,
     Tk_at_RefDiffusivity               = x.xEx
     full_multiphase_diffusion          = .x.
     /
```

The parameters in the namelist `Diffusion_Key_Parameters` are defined as follows:

`gas_diffusivity_equation_exponent`

A double precision variable describing the dependence of gas diffusivity on temperature (see Equation 6.4). The default value is 1.80.

`P_at_RefDiffusivity`

Pressure at the reference diffusivity (in Pa). If `P_at_RefDiffusivity <= 0`, the default value is 10^5 Pa.

`Tk_at_RefDiffusivity`

Temperature at the reference diffusivity (in K). If `T_at_RefDiffusivity <= 0`, the default value is 273.15 K.

`Option_gas_diffusivity_CompuMethod`

A character variable describing the method of estimation of the binary **gas** diffusivities. The following options are available:

`='Standard'`: This option involves the application of Equation (6.4), and requires non-zero multicomponent gas diffusivity values read from the standard input file.

`='Real_Gas_EOS'`: In this case, the binary gas diffusivities are computed from the cubic equation of state used to determine all the real gas properties. The diffusivities in the aqueous phase still need to be provided.

`='Constant'`: When this option is invoked, the constant multicomponent diffusivity values provided in the input file are used.

`full_multiphase_diffusion`

A logical variable describing the method of estimation of the method of estimation of multiphase diffusive fluxes. The following options are available:

`=.TRUE.`: With this option, harmonic weighting to the full multiphase effective diffusion strength is applied. This includes contributions from gas and aqueous phases, accounts for coupling of diffusion with phase partitioning effects, and can describe the most general cases of diffusion across phase boundaries.

`=.FALSE.`: In this case, harmonic weighting is performed separately for the diffusive fluxes in the mobile phases.

Records `DIFFUSION.2.1`, `DIFFUSION.2.2`, etc.

Record `DIFFUSION.2.1` is followed by `DIFFUSION.2.x` records, with `x = 1,…,NubMobPhases` (i.e., the number of mobile phases in the system under study). These records describe component diffusivities in the various phases. The same namelist is used in each one of these records. It is named

Component_Diffusivities_in_Phases, and has the following general form:

```
&Component_Diffusivities_in_Phases
     phase        = x,
     phase_number = x,
     component(1) = x,
          component_number(1)      = x,
          component_diffusivity(1) = x.xEx,
     component(2) = x,
          component_number(2)      = x,
          component_diffusivity(2) = x.xEx,
     …
     …
     …
     /
```

The parameters in the namelist Diffusion_Key_Parameters are defined as follows:

phase
: A character variable identifying the mobile phase for which the diffusivities of the various components are reported. The possible options in the **T+H** code are 'Aqueous' and 'Gas'.

phase_number
: An integer variable providing the number of the phase in the phase numbering sequence used in the code. For example, the possible values in a TOUGH+ v1.5 option involving aqueous and gas phases are:
= 2 for phase = 'Aqueous', and
= 1 for phase = 'Gas'.

component
: A character array of dimension NumCom (see Section 5.1) identifying the various mass components partioned in the phase in question (denoted by phase). The possible options the example above could be 'CH4', 'H2O' and 'NaCl' (if salinity is considered).

component_number
: An integer array providing the number of the component in the numbering sequence used in the code. The possible options in the **T+H** code are:
= 1 for component = 'CH4'
= 2 for component = 'H2O'
= 3 for component = 'NaCl' (if present)

component_diffusivity

A double precision array of dimension NumCom (see Section 5.1) describing the value of the multicomponent diffusivities D_β^κ (see Equations (2.59) and (6.4)) of the various components κ in the phase β under consideration (indentified by phase and phase_number, respectively).

NOTE: *The records* DIFFUSION.2.x *must provide data for all mobile phases and all components, even if the gas diffusivities may be overridden internally when* Option_gas_diffusivity_CompuMethod = 'Real_Gas_EOS'.

The structure of the namelists Diffusion_Key_Parameters and Component_Diffusivities_in_Phases (and their use as input formats in the data block **DIFFUSION**) are best illustrated in the example of **Figure 4.1**.

```
DIFFUSION-----*----2----*----3----*----4----*----5----*----6----*----7----*----8
&Diffusion_Key_Parameters  gas_diffusivity_equation_exponent  = 1.8d0
                           P_at_RefDiffusivity                = 1.0d5,    ! in Pa
                           Tk_at_RefDiffusivity               = 273.15d0, ! in K
                           Option_gas_diffusivity_CompuMethod = 'Real_Gas_EOS',
                           full_multiphase_diffusion          = .TRUE.
                           /
&Component_Diffusivities_in_Phases
        phase        = 'Aqueous',  phase_number = 2,
        component(1) = 'CH4',      component_number(1) = 1,
        component_diffusivity(1) = 1.0d-10,  ! (m2/s) ! Diffusivity of component 1 in phase 2
        component(2) = 'H2O',      component_number(2) = 2,
        component_diffusivity(2) = 1.0d-10,  ! (m2/s) ! Diffusivity of component 2 in phase 2
        component(3) = 'NaCl',     component_number(3) = 3,
        component_diffusivity(3) = 1.0d-10   ! (m2/s) ! Diffusivity of component 3 in phase 2
        /
&Component_Diffusivities_in_Phases
        phase        = 'Gas',      phase_number = 1,
        component(1) = 'CH4',      component_number(1) = 1,
        component_diffusivity(1) = 1.0d-05,  ! (m2/s) ! Diffusivity of component 1 in phase 1
        component(2) = 'H2O',      component_number(2) = 2,
        component_diffusivity(2) = 1.0d-05,  ! (m2/s) ! Diffusivity of component 2 in phase 1
        component(3) = 'NaCl',     component_number(3) = 3,
        component_diffusivity(3) = 0.0d-00   ! (m2/s) ! Diffusivity of component 3 in phase 1
        /
```

Figure 4.1. The **DIFFUSION** data block, with examples of the Diffusion_Key_Parameters and Component_Diffusivities_in_Phases namelists

6.4.1. User Options for Multiphase Diffusion

The treatment of full multiphase diffusion requires parameter specifications in two different data blocks. First, the user must set `binary_diffusion` to a value of `.TRUE.` in data block **`MEMORY`**. Diffusivities are input through the data block **`DIFFU`** (Section 6.4). See note in Section 6.4 regarding an alternative approach, which allows user to only specify the diffusion coefficient of salt and thus save in computation costs.

For two gaseous components ($N_G = 2$), the multicomponent diffusivities in the gas phase are the binary gas diffusivities. For $N_G > 2$ gaseous components κ_i, the multicomponent diffusivities of gas component κ_i are computed from the binary gas diffusivities by the Wilke method [*API*, 1977] as

$$D_G^{\kappa_i} = \frac{1-Y_G^{\kappa_i}}{\sum_{n=1,n\neq i}^{N_G} \frac{Y_G^{\kappa_n}}{d_G^{\kappa_i\kappa_n}}} \kappa_n , \tag{6.4}$$

where $d_G^{\kappa_i\kappa_n}$ is the binary gas diffusivity of gasesous components κ_n and κ_i.

When computing binary gas diffusivities $d_G^{\kappa_i\kappa_n}$ using the option `Option_gas_diffusivity_CompuMethod = 'Real_Gas_EOS'`, their pressure and temperature dependence is automatically accounted for in the computations. When `Option_gas_diffusivity_CompuMethod = 'Standard'`, the binary gas diffusivities of any two components κ_1 and κ_2 depend on pressure and temperature as [*Vargaftik*, 1975; *Walker et al*., 1981]

$$d_G^{\kappa_1\kappa_2}(P,T) = d_G^{\kappa_1\kappa_2}(P_0,T_0)\frac{P_0}{P}\left[\frac{T+273.15}{273.15}\right]^{\theta} \tag{6.5}$$

For example, at standard conditions of P_0 = 1 atm = 1.01325 bar, T_0 = 0 ˚C, the diffusion coefficient for vapor-air mixtures has a value of 2.13×10^{-5} m^2/s; the parameter θ for the temperature dependence is 1.80. TOUGH+ v1.5 can model a temperature dependence of gas phase diffusion coefficients according to Equation (6.5) by specifying parameter θ = `gas_diffusivity_equation_exponent` in the first record of data block **`DIFFUSION`** (see Section 6.4). Presently there are no provisions for allowing the dependence of the parameter θ on the gas phase composition.

The computation of the effective diffusion coefficient in equation (2.60) necessitates knowledge of the tortuosity. In TOUGH+ v1.5, this is computed using one of the methods described by the variable `Option_tortuosity_CompuMethod` in data block **`MEMORY`** (see Section 5.1).

6.5. Block-by-Block Permeability Modification

TOUGH+ v1.5 provides a feature that applies permeability modification (PM) coefficients for individual grid blocks according to

$$k_n \rightarrow k'_n = k_n \cdot \zeta_n \tag{6.6}$$

Here, k_n is the absolute (intrinsic or reference) permeability of grid block (cell, element) n, as specified in data block **`ROCKS`** or **`MEDIA`** for the domain to which that grid block belongs, while ζ_n is the permeability modification coefficient. The strength of capillary pressure will be automatically scaled according to *Leverett* [1941] as:

$$P_{cap,n} \rightarrow P'_{cap,n} = \frac{P_{cap,n}}{\sqrt{\xi_n}} \tag{6.7}$$

The subroutine `Initialize_Perm_Modifiers` is called after all **`MESH`** data have been processed. The `Initialize_Perm_Modifiers` routine initializes permeability modifiers and generates informative printout. To engage block-by-block permeability modification, users need to specify a (dummy) domain named "`SEED`" in block **`ROCKS`** or **`MEDIA`**. No grid blocks should be assigned to this domain; the presence of domain "`SEED`" simply serves as a flag to put permeability modification into effect, while data provided in domain "`SEED`" serve to select different options. Random (spatially uncorrelated) PM data can be internally generated in TOUGH+. Alternatively, externally defined permeability modifiers may be provided as part of the geometry data in block **`ELEME`**. This feature can be used to apply spatially correlated fields: users can run their own geostatistical package to generate whatever fields they desire, and then use a preprocessing program to place the PM-coefficients into the **`ELEME`** data block.

Note that this approach of block-by-block permeability modification affects only grid block permeabilities but not porosities. Full details on the various user options for PM-coefficients are given in informative printout that is automatically generated with each TOUGH+ run.

In addition to this approach, it is possible for the users to apply a geostatistical package of their choice to develop element-specific (block-by-block) statistically heterogeneous porosities ϕ, intrinsic permeabilities k, and phase saturations. These data can then be read as part of the **`INCON`** data block, as will be discussed in Section 8.

7.0. Geometrical Representation, Domain Discretization, and Grid Generation

The data blocks that specify the geometrical representation of the simulated system are discussed in this section, including the specification of elements (block **ELEM**), connections between elements (block **CONNE**), and the generation of grids (block **MESHM**) for radial symmetric grids, rectilinear grids, and grids containing fractured media. First, we discuss the convention used in TOUGH+ v1.5 for entering and processing geometrical data.

7.1. TOUGH+ Convention for Geometrical Data

Handling of flow geometry data in TOUGH+ is upward compatible with TOUGH2 [*Pruess et al*., 1999] and earlier TOUGH+ [*Moridis et al*., 2008] input formats and data

handling. As in other *integral finite difference* codes [*Edwards*, 1972; *Narasimhan and Witherspoon*, 1976], flow geometry is defined by means of a list of volume elements (*grid blocks*), and a list of flow connections between them (Section 3.3). This formulation can handle regular and irregular flow geometries in one, two, and three dimensions. Single- and multiple-porosity systems (porous and fractured media) can be specified, and higher order methods, such as seven- and nine-point differencing, can be implemented by means of appropriate specification of geometric data [*Pruess and Bodvarsson*, 1983].

In TOUGH+ v1.5, as in TOUGH2 [*Pruess et al*., 1999; 2012], volume elements are identified by names that consist of a string of either five or eight characters, '`12345`' or '`12345678`'. These are arbitrary, except that the last two characters (#4 and #5 in 5-character names; #7 and #8 in 8-character names) must be numbers; examples of valid 5-character and 8-character element names are '`ELE10`' and '`AB00CC23`', respectively. Flow connections are specified as ordered pairs of elements, such as '`ELE10ELE11`' (5-character names) or '`AB00CC23AB00CC24`' (8-character names). A variety of options and facilities are available for entering and processing geometric data. As in TOUGH2 [*Pruess et al*., 1999; 2012], element volumes and domain identification can be provided by means of a data block **`ELEME`** in the *standard input file*, while a data block **`CONNE`** can be used to supply connection data, including interface area, nodal distances from the interface, and orientation of the nodal line relative to the vertical (see Sections 7.2 and 7.3). These data are internally written to a disk file **`MESH`**, which in turn initializes the geometry data arrays used during the flow simulation. It is also possible to omit the **`ELEME`** and **`CONNE`** data blocks from the *standard input file*, and provide geometry data directly in the **`MESH`** file.

TOUGH+ v1.5 provides a capability for describing the flow system geometry and discretizing the domain by means of the **MeshMaker.f95** application written in FORTRAN 95/2003 (see Section 7.4). Unlike the approach in TOUGH2 [*Pruess et al*., 1999; 2012], **MeshMaker.f95** has been separate from (i.e., it is not integrated within) the core code since the first version of the TOUGH+ code [*Moridis et al*., 2008]. This application can perform a number of mesh generation and processing operations. The **MeshMaker.f95** code is written according to the tenets of Object-Oriented Programming, and has a modular structure. It can generate two-dimensional radially symmetric (*r,z*) meshes, and one-, two-, and three-dimensional rectilinear (Cartesian) grids in (*x,y,z*). Multiple-porosity processing for simulation of flow in naturally fractured reservoirs can be invoked by means of a keyword **MINC**, which stands for *Multiple INteracting Continua* (see Section 2.14). The `MINC` process operates on the data of the primary (porous medium) mesh as provided on disk file **MESH**, and generates a secondary mesh containing fracture and matrix elements with identical data formats on file **MINC**. The file **MESH** used in this process can be either directly supplied by the user, or it can have been generated from an earlier application of the **MeshMaker.f95** application.

7.2. Data Block ELEME

This block introduces element (grid block) information.

Record `ELEME.1` (for 5-character elements only)

Format (A3, I2, 2I5, A5, 6E10.4, 1X, A3)

```
ElName5C, NSEQ, NADD, MA12, elem_vol,
elem_aht, elem_pm, X, Y, Z, elem_activity
```

ElName5C

The five-character name of an element. The first three characters can be letters, numbers or blanks, and the last two characters (NE = ElName5C(4:5)) must be numbers.

NSEQ

The number of additional elements having the same volume and belonging to the same reservoir domain.

NADD

The increment between the code numbers of two successive elements. (Note: if the longest dimension in the grid is <1000, the maximum permissible code number NE + NSEQ *NADD is ≤ 999.)

MA12

A five-character material identifier corresponding to one of the reservoir domains as specified in block **ROCKS**. If the first three characters are blanks and the last two characters are numbers, then they indicate the sequence number of the domain as entered in **ROCKS**. If left blank the element is by default assigned to the first domain in block **ROCKS**.

elem_vol

The element volume (m^3).

elem_aht

The interface area (m^2) for heat exchange with semi-infinite confining beds. Internal **MESH** generation via the **MeshMaker.f95** facility will automatically generate elem_aht.

elem_pm

The permeability modifier (optional, active only when a domain "SEED" has been specified in the **ROCKS** block; see Section 6.5). Will be used as multiplicative factor for the permeability parameters from block **ROCKS**. Simultaneously, the capillary pressure strength will be scaled as elem_pm $^{-1/2}$. elem_pm = 0 will result in an impermeable block.

Random permeability modifiers can be generated internally, see detailed comments in the TOUGH+ output file. The elem_pm may be used to specify spatially correlated heterogeneous fields, but users need their own preprocessing programs for this, as TOUGH+ provides no internal capabilities for generating such fields.

X, Y, Z

Cartesian coordinates of the grid block centers. These are included in the **ELEME** data for listing in output files of the simulation results that conform to the specifications of the **TecPlot** package. Additionally, the

coordinate data *may be needed* for (a) the computation of interblock properties in cylindrical grids or (b) the initialization of a gravity-capillary equilibrium (such as the one described in Sections 6.9 and 9.10 of *Pruess et al.* [1999]), and *are needed* (c) for mechanical dispersion computations in solute transport investigations [*Moridis et al*., 1999].

`elem_activity`
: A (optional) character variable that describes the activity status of an element. An element is rendered inactive (i.e., a Dirichlet boundary) by invoking the options `elem_activity ='I__'` (indicating that the element conditions are time-invariant) or `elem_activity = 'Vxx'` (denoting several time-variable conditions, each numbered by the xx identifier). The grid block is assumed active by default for any other value of `elem_activity`, including a blank (see Section 8.4).

Record `ELEME.1` (for 8-character elements only)

Format (A8, 7X, A5, 6E10.4, 1X, A1)

```
ElName8C, MA12, elem_vol, elem_aht,
elem_pm, X, Y, Z, elem_activity
```

`ElName8C`
: The eight-character name of an element. The first five characters can be letters, numbers or blanks, and the last two characters (`NE = ElName8C(7:8)`) must be numbers.

All other variables and parameters are as previously defined in case of the five-character element names.

*Repeat record **`ELEME.1`** for the number of elements desired.*

Record `ELEME.2`

A blank record closes the **`ELEME`** data block.

NOTE: *The number of elements described in the **`ELEME`** data block cannot exceed the number* `Max_NumElem` *specified in the* **`MEMORY`** *data block (See Section 5.1). If this happens, an error message is printed and the simulation is aborted.*

7.3. Data Block CONNE

This block introduces information for the connections (interfaces) between elements.

Record CONNE.1 (for connections involving 5-character elements only)

Format (A5, A5, 4I5, 5E10.4)

```
ConxName1, ConxName2, NSEQ, NAD1, NAD2, ConxKi,
ConxD1, ConxD2, ConxArea, ConxBeta, emissivity
```

These parameters are defined as follows:

`ConxName1`
: The name of the first five-character element in the connection.

`ConxName2`
: The code name of the five-character second element in the connection.

`NSEQ`
: The number of additional connections in the sequence.

`NAD1`
: The increment of the number of the first element (as defined by the last two characters of its name, i.e., `ConxName1(4:5)`) between two successive connections.

`NAD2`
: The increment of number of the second element (as defined by the last two characters of its name, i.e., `ConxName2(4:5)`) between two successive connections.

`ConxKi`
: The permeability index of the connection. Setting it equal to 1, 2, or 3 specifies absolute permeability to be `MediaPerm(ConxKi)` for the materials in elements (`ConxName1`) and (`ConxName2`), where `MediaPerm` is read in block **`ROCKS`**. This allows assignment of different permeabilities, e.g., in the horizontal and vertical direction.

`ConxD1, ConxD2`
: The distance [m] between the common interface of a connection from the centers of the first and second element in the connection, respectively.

`ConxArea`
: The element interface area [m^2].

`ConxBeta`
: The cosine of the angle between the gravitational acceleration vector and the line between the two elements. `ConxBeta*gravity > 0 (<0)` corresponds to first element being above (below) the second element.

`emissivity`

The *radiant emittance* factor for radiative heat transfer, which for a perfectly "black" body is equal to 1. The rate of radiative heat transfer between the two grid blocks is:

$$Q_{H,rad} = \text{emissivity} * \sigma_0 * \text{ConxArea} * (T_2^4 - T_1^4)$$

where σ_0 = 5.6687e-8 J/m^2 K^4 s is the Stefan-Boltzmann constant, and T_1 and T_2 are the absolute temperatures of the two grid blocks. The term `emissivity` may be entered as a negative number, in which case the absolute value will be used, and heat conduction at the connection will be suppressed. Setting `emissivity = 0.0e0` will result in no radiative heat transfer.

Record `CONNE.1` (for connections involving 5-character elements only)

Format (A8, A8, 9X, I5, 5E10.4)

```
Conx8Name1, Conx8Name2, ConxKi,
ConxD1, ConxD2, ConxArea, ConxBeta,
emissivity
```

`Conx8Name1`

The name of the first eight-character element in the connection.

`Conx8Name2`

The code name of the second eight-character element in the connection.

All other variables and parameters are as previously defined in case of connections between five-character elements.

Repeat record `CONNE.1` for the number of connections desired.

Record `CONNE.2`

A blank record closes the **`CONNE`** data block. Alternatively, connection information may terminate on a record with `'+++ '` typed in the first five columns, followed by element cross-referencing information. The second type of termination is generated in the **`MESH`** file upon completion of a TOUGH+ run.

NOTE*: The number of connections described in the* **`CONNE`** *data block cannot exceed the number* `Max_NumConx` *specified in the* **`MEMORY`** *data block (See Section 5.1). If this happens, an error message is printed and the simulation is aborted*

7.4. The MeshMaker.f95 Facility

In this section we discuss the use of the **MeshMaker.f95** facility, and the required parameter inputs for mesh generation and processing. As indicated earlier, unlike in TOUGH2 [*Pruess et al*., 1999; 2012], **MeshMaker.f95** is not integrated into the TOUGH+ code but is an independent program that is written in FORTRAN 95/2003, has a modular structure, and an architecture based on the principles of Object-Oriented Programming. The **MeshMaker.f95** input has a modular structure, which is organized by keywords in a manner analogous to that of TOUGH+. This section provides detailed instructions for preparing the input files, illustrative examples of which are shown in **Figures 7.1** to **7.4**.

7.4.1. Inputs Related to Problem Definition and Dimensioning

These inputs occupy two records (both mandatory), and provide (a) a short description of the problem through an informative title, and (b) data to allow proper dimensioning of the work arrays and formatting of the **MESH** file to be generated. These two initial records are discussed in detail below:

Record MESHMAKER.1

The first record of the input file in any **MeshMaker.f95** application is the character variable **TITLE**, which includes a header of up to 80 characters and is read using a free format. This record is necessary for any **MeshMaker** simulation to begin.

Record MESHMAKER.2

The following variables are read in MESHMAKER.2 using a free format:

```
MaxNum_Elem, Longest, ElemNameLength,
FormatType, LengthUnits, media_by_number
```

These parameters are defined as follows:

`Max_NumElem`
: Integer denoting the maximum number of elements in the grid under construction.

`Longest`
: Integer indicating the maximum expected number of subdivisions along any of the coordinates in the grid under construction.

`ElemNameLength`
: Integer variable defining the number of characters in the element names. It may be either 5 or 8. If unequal to 8, `ElemNameLength` is internally resest to the default (`= 5`).

`FormatType`
: Character variable indicating the format of the data in the MESH file to be created. It may have one of two values:

  `= 'Old'`: This option creates a **`MESH`** file that conforms to the data format described in Sections 7.2 and 7.3, and is consistent with the TOUGH2 formats [*Pruess et al*., 1999; 2012].

  `= 'New'`: This option creates a **`MESH`** file in which the element and connection data are listed using the `NAMELIST` format facility available in FORTRAN 95/2003. This is intended for future versions of TOUGH+

  NOTE: *TOUGH+ v1.5 does not accept* `NAMELIST`-based *formats in the* **`MESH`** *file. Thus, the option* `FormatType = 'Old'` *must be used in all* **`MeshMaker.f95`** *applications*.

`LengthUnits`
: Character variable specifying the units of length of the grid to be created. The possible options are: '`m`', '`mm`', '`km`', '`ft`' and '`in`', indicating meters, millimeteres, kilometers, feet and inches, respectively. Note that all dimensions are converted internally into SI units (meters), which is the standard unit of length of the resulting mesh.

`media_by_number`
: Logical variable specifying how the various media (rocks) are to be named. The possible options are: `.T.` or `.TRUE.`, in which case the media are named by the character variable `H_RegionName` (see Section 7.4.2), and `.F.` of `.FALSE.`, indicating media identification by the number in the sequence of their listing. The `media_by_number = .T.` option is to be used only in the case of fractured media that are to be described by the Multiple

Intercative Continua (MINC) concept, and is needed for the creation of an MESH interim files. This will be further processed to create the MINC file to be used in the simulation. When the media_by_number = .TRUE. option is invoked, unfractured media are identified by a negative number, and fractured media (identified by an 'F' or 'f' as the first letter of the value of the H_RegionName character variable) are identified by a positive number.

7.4.2. Inputs Related to Domain Heterogeneity

These optional inputs provide information that allows the description of heterogeneity within the domain. The assignment of heterogeneity is based on the definition of *regions* using geometrical information that describe the location and extent of these subdomains. Thus, each of the individually defined regions is assigned the properties of a particular medium that will have to be included in the **ROCKS** data block in the input file of the subsequent TOUGH+ v1.5 simulation.

The records in this data block are discussed in detail below:

Record MESHMAKER.3 (**Optional**)

This record includes only the keyword (character variable) 'Regions', which is read using a free format. If the domain is homogeneous, then there is no need to provide any of the inputs discussed in Section 7.4.2.

Record MESHMAKER.3.1 (**Optional** – when the MESHMAKER.3 record is included)

This record is read using a free format, and includes the single integer variable Num_HetRegions (> 0) that describes the number of heterogeneous regions (subdomains) that are to be described by the ensuing data records. The minimum value that Num_HetRegions can accept is 1, corresponding to a homogeneous system. An error will occur and the execution will stop if Num_HetRegions < 1.

Because of dynamic dimensioning, there is no limit in the number of heterogeneous regions defined by Num_HetRegions. However, practical considerations may limit the size of Num_HetRegions. Thus, although it is possible to define element-by-element heterogeneity using this approach, this would be a very tedious process. Generally speaking, the most useful application

of this facility is in the description of extensive geologic units with distinctly different properties.

Record MESHMAKER.3.2 (**Optional** – when the MESHMAKER.3 record is included)

This record is read using a free format, and includes the single character variable dominant_medium that provides the name (5 character long) of the dominant (reference) porous medium in the heterogeneous domain. Note that the selection of a medium as dominant_medium is be arbitrary, as there are no restrictions on the extent of its spatial distribution for it to be designated as such.

If Num_HetRegions = 1, no more data need to be read. This case is equivalent to a homogeneous system, and the entire heterogeneity-related data block (records MESHMAKER.3 to MESHMAKER.3.2 may be omitted.

Record MESHMAKER.3.3.0 (**Optional** – when the MESHMAKER.3 record is included and Num_HetRegions > 1)

This record is read using a free format, and includes the single character variable H_RegionName that provides the name (5 character long) of the porous medium in the region (subdomain) that is about to be defined.

Record MESHMAKER.3.3.1 (**Optional** – when the MESHMAKER.3 record is included and Num_HetRegions > 1)

Here the following character variables are read using a free format:

H_RegionCoordinates, H_RegionUnits

These parameters are defined as follows:

H_RegionCoordinates

A character variable indicating the coordinate system used in the geometric definition of the region that is about to be described in the heterogeneous domain. It my have one of two values:

= 'Cartesian' : This option indicates that the region is defined geometrically in terms of Cartesian coordinates.

= 'Cylindrical' : This option indicates that the region is defined geometrically in terms of cylindrical coordinates.

H_RegionUnits

A character variable describing the units of length used in the description of the geometry of the region. The following values are acceptable options:

'mm', 'km', 'km', 'in', or 'ft',

indicating millimeters, meters, kilometers, inches and feet, respectively. Note that **MeshMaker.f95** converts all length units into meters (the length unit used in the TOUGH+ simulations) prior to producing the **MESH** file.

Record MESHMAKER.3.3.2 (**Optional** – when the MESHMAKER.3 record is included and Num_HetRegions > 1)

If H_RegionCoordinates = 'Cartesian', the following real variables are read in MESHMAKER.3.3.2 using a free format:

Xmin, Xmax, Ymin, Ymax, Zmin, Zmax

These parameters are defined as follows:

Xmin, Xmax
: Real variables indicating the range of the region along the *x*-axis of the Cartesian coordinate system.

Ymin, Ymax
: Real variables indicating the range of the region along the *y*-axis of the Cartesian coordinate system.

Zmin, Zmax
: Real variables indicating the range of the region along the *z*-axis of the Cartesian coordinate system.

If H_RegionCoordinates = 'Cylindrical', the following real variables are read in MESHMAKER.3.3.2 using a free format:

Rmin, Rmax, Zmin, Zmax

These parameters are defined as follows:

Rmin, Rmax
: Real variables indicating the range of the region along the *r*-axis of the cylindrical coordinate system.

Zmin, Zmax
: Real variables indicating the range of the region along the *z*-axis of the cylindrical coordinate system.

Repeat records MESHMAKER.3.3.0, MESHMAKER.3.3.1, MESHMAKER.3.3.3 for a total of Num_HetRegions – 1 regions. Note that the region dominant_medium does not need geometric definition.

7.4.3. Inputs Related to Description of Boundaries

These optional inputs provide information that describes the outer boundaries of the domain under discretization. The assignment of grid subdomains as boundaries is based on the geometry-based definition of the outer spatial limits of the domain under discretization. Thus, the boundaries are treated as special types of regions (see Section 7.4.2) with the properties of a particular medium that will have to be included in the **ROCKS** data block in the input file of the subsequent TOUGH+ simulation. It is possible for a region and one or more boundaries to be described by the same medium in the **ROCKS** data block.

The records in this data block are discussed in detail below:

Record `MESHMAKER.4` (**Optional**)

This record includes only the keyword (character variable) '`Boundaries`', which is read using a free format. If the domain is confined by no-flow (Newman-type) boundaries, then there is no need to provide any of the inputs discussed in Section 7.4.3.

Record `MESHMAKER.4.1` (**Optional** – when the `MESHMAKER.4` record is included)

This record is read using a free format, and includes the single integer variable `Num_Boundaries` (`> 0`) that describes the number of boundaries that are to be described.

Record `MESHMAKER.4.2.0` (**Optional** – when the `MESHMAKER.4` record is included and `Num_Boundaries > 1`)

The following character variables are read in `MESHMAKER.4.2` using a free format:

```
BoundID, BoundRegionName
```

These parameters are defined as follows:

`BoundID`

A character variable indicating the type of boundary described at the prescribed location. It my have one of two values:

= 'I': This option indicates an *inactive* boundary, the conditions and properties of which are time-invariant.

= 'V': This option indicates a time-variable boundary.

The result of this designation is reflected in the **ELEME** block, in which `elem_activity` (see Section 7.2) is set to `BoundID` in all the cells corresponding to the boundary defined here. Any other value of the `BoundID` variable causes the program to print an error message and stop execution.

`BoundRegionName`

A character variable `H_RegionNname` that provides the name (5 character long) of the porous medium in the boundary that is about to be defined.

Record `MESHMAKER.4.2.1` (**Optional** – when the `MESHMAKER.4` record is included and `Num_Boundaries > 1`)

Here the following character variables are read using a free format:

`BoundRegionCoordinates, BoundRegionUnits`

These parameters are defined as follows:

`BoundRegionCoordinates`

A character variable indicating the coordinate system used in the geometric definition of the boundary that is about to be described in the heterogeneous domain. It my have one of two values:

= 'Cartesian': This option indicates that the region is defined geometrically in terms of Cartesian coordinates.

= 'Cylindrical': This option indicates that the region is defined geometrically in terms of cylindrical coordinates.

`BoundRegionUnits`

A character variable describing the units of length used in the description of the geometry of the boundary. The following values are acceptable options:

'mm', 'km', 'km', 'in', or 'ft',

indicating millimeters, meters, kilometers, inches and feet, respectively. Note that **MeshMaker.f95** converts all length units into meters (the

length unit used in the TOUGH+ simulations) prior to producing the **MESH** file.

Record MESHMAKER.4.2.2 (**Optional** – when the MESHMAKER.4 record is included and Num_Boundaries > 1)

If BoundRegionCoordinates = 'Cartesian', the following real variables are read in MESHMAKER.3.3.2 using a free format:

```
Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
```

These parameters are defined as follows:

Xmin, Xmax
: Real variables indicating the range of the boundary region along the *x*-axis of the Cartesian coordinate system.

Ymin, Ymax
: Real variables indicating the range of the boundary region along the *y*-axis of the Cartesian coordinate system.

Zmin, Zmax
: Real variables indicating the range of the boundary region along the *z*-axis of the Cartesian coordinate system.

If BoundRegionCoordinates = 'Cylindrical', the following real variables are read in MESHMAKER.3.3.2 using a free format:

```
Rmin, Rmax, Zmin, Zmax
```

These parameters are defined as follows:

Rmin, Rmax
: Real variables indicating the range of the region boundary along the *r*-axis of the cylindrical coordinate system.

Zmin, Zmax
: Real variables indicating the range of the boundary region along the *z*-axis of the cylindrical coordinate system.

Repeat records MESHMAKER.4.2.0, MESHMAKER.4.2.1, MESHMAKER.4.2.3 for a total of Num_Boundaries boudaries.

7.4.4. Inputs for Grid Construction

There are three grid construction options available in **MeshMaker.f95.** These options are activated by appropriate keywords. Thus, the keywords 'RZ2D' or 'RZ2DL' invoke generation of a one or two-dimensional radially symmetric (*r,z*) mesh; 'XYZ' initiates generation of a one, two, or three dimensional Cartesian (*x,y,z*) mesh; and 'MINC' calls a modified version of the GMINC program [*Pruess*, 1983] to sub-partition a primary porous medium mesh into a secondary mesh for fractured media, using the method of *Multiple INteracting Continua* [*Pruess and Narasimhan*, 1982; 1985].

The meshes generated under keyword 'RZ2D' or 'XYZ' are internally written to file **MESH**. The MINC-processing operates on the data in a file **MESH** that is either pre-existing or is created directly from the inputs of **MeshMaker.f95** when the 'RZ2D', 'RZ2DL' or 'XYZ' data blocks are followed by the 'MINC' datablock. We shall now separately describe the preparation of input data for the three grid construction options.

7.4.4.1. Generation of radially symmetric grids (keyword 'RZ2D' or 'RZ2DL'). The keywords 'RZ2D' or 'RZ2DL' invoke generation of a radially symmetric mesh. Values for the radii to which the grid blocks extend can be provided by the user or can be generated internally (see below). Nodal points will be placed halfway between neighboring radial interfaces. When 'RZ2D' is specified, the mesh will be generated by columns; i.e., in the **ELEME** block we will first have the grid blocks at smallest radius for all layers, then the next largest radius for all layers, and so on.

With keyword 'RZ2DL' the mesh will be generated by layers; i.e., in the **ELEME** block we will first have all grid blocks for the first (top) layer from smallest to largest radius, then all grid blocks for the second layer, and so on. Apart from the different

ordering of elements, the two meshes for 'RZ2D' and 'RZ2DL' are identical. The reason for providing the two alternatives is as a convenience to users in implementing boundary conditions by way of inactive elements (see Section 6.4 in *Pruess et al*. [1991]). Assignment of inactive elements would be made by using a text editor on the RZ2D-generated **MESH** file, and moving groups of elements towards the end of the **ELEME** block, past a dummy element with zero volume. 'RZ2D' makes it easy to declare a vertical column inactive, facilitating assignment of boundary conditions in the vertical, such as a gravitationally equilibrated pressure gradient. 'RZ2DL' on the other hand facilitates implementation of areal (top and bottom layer) boundary conditions.

The inputs for cylindrical systems are as follows:

Data Block GRID

The first record in this data block includes only the keyword (character variable) 'RZ2D' or 'RZ2DL'. This keyword is read using a Format (A5).

Record RADII.0

RADII: A keyword that introduces data for defining a set of interfaces (grid block boundaries) in the radial direction. It is read using a Format (A5).

Record RADII.1

Format (I5)
NRAD

NRAD: Number of radii that will be read. At least one radius must be provided, indicating the inner boundary of the mesh.

Record RADII.2, RADII.3, etc.

Format (8E10.4)
RC(i), i = 1,NRAD

RC(i): The radii defining the element boundaries in ascending order [m].

Record EQUID.0

EQUIDistant

Keyword indicating that the ensuing data describe a set of equal radial increments. This keyword is read using Format (A5).

Record EQUID.1

Format (I5, 5X, E10.4)
NEQU, DR

NEQU: The number of desired radial increments.

DR: The size of the radial increment [m].

NOTE: *At least one radius must have been defined via block* RADII *before* EQUID *can be invoked*.

Record LOGAR.0

LOGARithmic — This keyword introduces data on radial increments that increase from one to the next by the same factor (i.e., $\Delta r_{n+1} = f \bullet \Delta r_n$).

Record LOGAR.1

Format(I5, 5X, 2E10.4)
NLOG, RLOG, DR

NLOG: The number of desired radial increments.

RLOG: The desired radius r_{max} of the last (largest) of these radii.

DR: The reference radial increment Δr_0: the first Δr generated will be equal to $f \bullet \Delta r_0$, with f internally determined such that the last increment will bring total radius to RLOG = r_{max}. The factor $f < 1$ for decreasing radial increments is permissible. If Δr_0 is set equal to zero, or left blank, the last increment Δr generated before the keyword **LOGAR** is invoked will be used as default.

Additional blocks RADII, EQUID, and LOGAR can be specified in arbitrary order.

NOTE: *At least one radius must have been defined before the* LOGAR *option can be invoked. If* $\Delta r_0 = 0$*, at least two radii must have been defined*.

Record LAYER.0

LAYER: This keyword introduces information on horizontal layers, and signals closure of RZ2D input data. It is read using a Format (A5).

Record LAYER.1

Format (I5)
NLAY

NLAY: The number of horizontal layers in the cylindrical grid.

Record LAYER.2

Format (8E10.4)
H(i), i = 1, NLAY

H(i): The thicknesses of the horizontal layers in the cylindrical grid, from top layer downward. By default, zero or blank entries for layer thickness will result in assignment of the last preceding non-zero entry. Assignment of a zero layer thickness, as needed for inactive layers, can be accomplished by specifying a negative value.

NOTE: *The* LAYER *data close the* RZ2D *or* RZ2DL *data block.*

Record GRID-END

Two blank records close the input data file *if* further MINC-processing of the grid is not needed. If such processing is specified, then **a single** blank record terminates the RZ2D or RZ2DL data block, and is then followed by the MINC data block (see Section 7.2.5).

7.4.4.2. Generation of rectilinear grids (keyword 'XYZ')

Data Block GRID

The first record in this data block includes only the keyword (character variable) 'XYZ' that is invokes generation of a Cartesian (rectilinear) mesh. This keyword is read using a Format (A5).

Record XYZ.1

DEG, X_ref, Y_ref, Z_ref (Free format)

DEG: The angle (in degrees) between the *y*-axis and the horizontal. If gravitational acceleration (parameter gravity in record PARAM.2, see Section 10) is positive, -90° < DEG < 90° corresponds to grid layers going from top down. Grids can be specified from the bottom layer up

by setting `gravity` or `ConxBeta` (Section 7.3) to negative values. The default (`DEG = 0.0e0`) corresponds to horizontal *y*- and vertical *z*-axis. The *x*-axis is always horizontal.

`X_ref, Y_ref, Z_ref`:
The reference *x*-, *y*-, and *z*-coordinates at the origin of the axes.

Record `XYZ.2`
Format (A2, 3X, I5, E10.4)
`NTYPE, NO, DEL`

`NTYPE`: A character variable that can assume one of the values '`NX`', '`NY`' or '`NZ`', specifying grid increments in the *x*-, *y*-, or *z*-direction, respectively.

`NO`: The number of grid increments.

`DEL`: The constant grid increment for `NO` grid blocks, if set to a non-zero value.

Record XYZ.3 (**Optional**, `DEL = 0.0e0` or blank only)
Format (8E10.4)
`DEL(i), i = 1, NO`

`DEL(i)`: A set of grid increments in the direction specified by `NTYPE` in record `XYZ.2`. Additional records with formats as `XYZ.2` and `XYZ.3` can be provided, with *x*-, *y*-, and *z*-data in arbitrary order.

Record `XYZ.4` A blank record closes the XYZ data block.

NOTE: *The end of the data block* `GRID` *is also marked by a blank record. Thus, when* `GRID`/`XYZ` *is used, there will be* ***two*** *blank records at the end of the input data file. Alternatively,* ***a single*** *blank record and a new record with the keyword* '`MINC`' *can be used to invoke* `MINC`*-processing for fractured media (see below).*

```
Meshmaker test: Cartesian grid
1000 20  5 'Old' 'm'  .FALSE.
XYZ
       00.
NX      10    1.0e00
NY       5       2.0
NZ      20       1.0
```

Figure 7.1. An example of a **`MeshMaker.f95`** input file for the creation of a Cartesian 3D grid. Note that no heterogeneous regions or boundaries are defined in this grid.

```
Meshmaker test: Cylindrical grid
6000  6000  5 'Old' 'm'  .FALSE.
RZ2DL
RADII
    2
    1.0e-5    1.0e-3
EQUID
 4000         2.5e-2
LOGAR
  500         2.5e+3
EQUID
    1         1.0e-3
LAYER
    1
    1.0e00
```

Figure 7.2. An example of a **`MeshMaker.f95`** input file for the creation of a single-layer (1D) cylindrical grid. Note that no heterogeneous regions or boundaries are defined in this grid

```
Input file for a large 3D cartesian grid       ! Title
500000 100 5  'Old' 'm' .FALSE.
Regions                                        ! Keyword denoting heterogeneous subdomains
6                                              ! ==> # of heterogeneous media regions
'HydrL'                                        ! Region #1: Name of dominant medium
'Aquif'                                        ! Region #2: medium name
'cartesian'     'm'                            !    coordinates, units
0.0e0 1.5e3 0.0e0 1.5e3 -6.3e1 -4.825e1        !    Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
'OverB'                                        ! Region #3: medium name
'cartesian'   'm'                              !    coordinates, units
0.0e0 1.5e3 0.0e0 1.5e3  -3.0e1  0.0e1         !    Rmin, Rmax, Zmin, Zmax
'UndrB'                                        ! Region #4: medium name
'cartesian'   'm'                              !    coordinates, units
0.0e0 1.5e3 0.0e0 1.5e3  -9.4e1 -6.3e1         !    Rmin, Rmax, Zmin, Zmax
'WellA'                                        ! Region #5: medium name
'cartesian'   'm'                              !    coordinates, units
0.0e0 5.0e-2 0.0e0 5.0e-2 -5.4e1 -3.0e1        !    Rmin, Rmax, Zmin, Zmax
'WellA'                                        ! Region #6: medium name
'cartesian'   'm'                              !    coordinates, units
4.9995e2 5.0e2 4.9995e2 5.0e2 -5.4e1 -3.0e1    !    Rmin, Rmax, Zmin, Zmax
Boundaries
2                                              ! ==> # of boundaries
'I' 'TopBB'                                    ! Boundary #1: type and medium name
'cartesian'   'm'                              !    coordinates, units
0.0e0 1.5e3 0.0e0 1.5e3 -1.0e-2  0.0e0         !    Rmin, Rmax, Zmin, Zmax
'I' 'BotBB'                                    ! Boundary #2: type and medium name
'cartesian'   'm'                              !    coordinates, units
0.0e0 1.5e3 0.0e0 1.5e3 -9.4e1 -9.3e1          !    Rmin, Rmax, Zmin, Zmax
XYZ
       00.
NX      76
    5.0e-2 2.5000e-1 2.8599e-1 3.2716e-1 3.7426e-1 4.2813e-1 4.8976e-1 5.6027e-1
 6.4092e-1 7.3319e-1 8.3873e-1 9.5947e-1 1.0976e+0 1.2556e+0 1.4363e+0 1.6431e+0
 1.8797e+0 2.1502e+0 2.4598e+0 2.8139e+0 3.2190e+0 3.6823e+0 4.2124e+0 4.8188e+0
 5.5125e+0 6.3061e+0 7.2139e+0 8.2524e+0 9.4404e+0 1.0799e+1 1.2354e+1 1.4132e+1
 1.6167e+1 1.8494e+1 2.1157e+1 2.4202e+1 2.7686e+1 3.1672e+1 3.1672e+1 2.7686e+1
 2.4202e+1 2.1157e+1 1.8494e+1 1.6167e+1 1.4132e+1 1.2354e+1 1.0799e+1 9.4404e+0
 8.2524e+0 7.2139e+0 6.3061e+0 5.5125e+0 4.8188e+0 4.2124e+0 3.6823e+0 3.2190e+0
 2.8139e+0 2.4598e+0 2.1502e+0 1.8797e+0 1.6431e+0 1.4363e+0 1.2556e+0 1.0976e+0
 9.5947e-1 8.3873e-1 7.3319e-1 6.4092e-1 5.6027e-1 4.8976e-1 4.2813e-1 3.7426e-1
 3.2716e-1 2.8599e-1 2.5000e-1    5.0e-2
NY      76
    5.0e-2 2.5000e-1 2.8599e-1 3.2716e-1 3.7426e-1 4.2813e-1 4.8976e-1 5.6027e-1
 6.4092e-1 7.3319e-1 8.3873e-1 9.5947e-1 1.0976e+0 1.2556e+0 1.4363e+0 1.6431e+0
 1.8797e+0 2.1502e+0 2.4598e+0 2.8139e+0 3.2190e+0 3.6823e+0 4.2124e+0 4.8188e+0
 5.5125e+0 6.3061e+0 7.2139e+0 8.2524e+0 9.4404e+0 1.0799e+1 1.2354e+1 1.4132e+1
 1.6167e+1 1.8494e+1 2.1157e+1 2.4202e+1 2.7686e+1 3.1672e+1 3.1672e+1 2.7686e+1
 2.4202e+1 2.1157e+1 1.8494e+1 1.6167e+1 1.4132e+1 1.2354e+1 1.0799e+1 9.4404e+0
 8.2524e+0 7.2139e+0 6.3061e+0 5.5125e+0 4.8188e+0 4.2124e+0 3.6823e+0 3.2190e+0
 2.8139e+0 2.4598e+0 2.1502e+0 1.8797e+0 1.6431e+0 1.4363e+0 1.2556e+0 1.0976e+0
 9.5947e-1 8.3873e-1 7.3319e-1 6.4092e-1 5.6027e-1 4.8976e-1 4.2813e-1 3.7426e-1
 3.2716e-1 2.8599e-1 2.5000e-1    5.0e-2
NZ      77
    1.0e-3    7.00e0    5.00e0    4.0e+0    3.2e+0    2.5e+0    2.0e00    1.6e00
   1.25e00    1.0e00    8.0e-1    6.5e-1    5.0e-1    5.0e-1    4.0e-1    4.0e-1
    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1
    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1
    4.0e-1    4.0e-1    4.5e-1    4.5e-1    4.5e-1    4.5e-1    4.5e-1    4.0e-1
    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1
    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1    4.0e-1
    4.0e-1    4.0e-1    4.0e-1    5.0e-1    5.0e-1    6.5e-1    8.0e-1    1.0e00
    1.0e00   1.25e00    1.6e00    2.0e00    2.5e00    3.2e00    3.0e00    4.00e0
    4.0e00    5.0e00    6.0e00    8.0e00    1.0e-3


====> ====> ====> ====>
```

Figure 7.3. An example of a **`MeshMaker.f95`** input file for the creation of a large Cartesian 3D grid with heterogeneous regions and defined boundaries.

```
Input file for a large cylindrical grid  ……! Title
15000 30000 5 'Old' 'm' .FALSE.
Regions                        ! Keyword denoting heterogeneous subdomains
  6                            ! => Num_HetRegions ( = number of heterogeneous regions)
'HydrL'                        !    Region #1: dominant_medium
'Aquif'                        !    Region #2: H_RegionName
'cylindrical'   'm'            !       H_RegionCoordinates, H_RegionUnits
0.0e0 1.5e3 -6.3e1 -4.825e1    !       Rmin, Rmax, Zmin, Zmax
'OverB'                        !    Region #3: H_RegionName
'cylindrical'   'm'            !       H_RegionCoordinates, H_RegionUnits
0.0e0 1.5e3 -3.0e1  0.0e1      !       Rmin, Rmax, Zmin, Zmax
'UndrB'                        !    Region #4: H_RegionName
'cylindrical'   'm'            !       H_RegionCoordinates, H_RegionUnits
0.0e0 1.5e3 -9.4e1 -6.3e1      !       Rmin, Rmax, Zmin, Zmax
'Casng'                        !    Region #5: H_RegionName
'cylindrical'   'm '           !       H_RegionCoordinates, H_RegionUnits
0.0e0 1.0e-1 -4.6e1 -3.0e1     !       Rmin, Rmax, Zmin, Zmax
'Perfo'                        !    Region #5: H_RegionName
'cylindrical'   'm '           !       H_RegionCoordinates, H_RegionUnits
0.0e0  1.0e-1 -5.225e1 -4.6e1  !       Rmin, Rmax, Zmin, Zmax
Boundaries                     ! Keyword denoting boundaries
2                              ! ==> Num_Boundaries
'I' 'TopBB'                    !    Boundary #1: BoundID, BoundRegionName
'cylindrical'   'm'            !       BoundRegionCoordinates, BoundRegionUnits
0.0e0 1.5e3 -1.0e-2  0.0e0     !       Rmin, Rmax, Zmin, Zmax
'I' 'BotBB'                    !    Boundary #2: BoundID, BoundRegionName
'cylindrical'   'm'            !       BoundRegionCoordinates, BoundRegionUnits
0.0e0 1.5e3 -9.4e1 -9.3e1      !       Rmin, Rmax, Zmin, Zmax
RZ2DL
RADII
    2
     0.001   0.10795
EQUID
    1         2.0e-2
LOGAR
  199         1.0e+3
LAYER
  68
    1.0e-3    6.50e0    5.00e0    5.0e+0    3.0e+0    3.0e+0    2.0e00    2.0e00
    1.0e00    1.0e00    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1
    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1
    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1
    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1
    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.5e-1    5.5e-1    5.5e-1    5.5e-1
    5.5e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    5.0e-1    1.0e00
    1.0e00    2.0e00    2.0e00    3.0e+0    3.0e+0    4.0e+0    4.0e+0    5.00e0
    5.0e+0    6.00e0    6.0e+0    1.0e-3


====> ====> ====> ====>
```

Figure 7.4. An example of a **`MeshMaker.f95`** input file for the creation of a large cylindrical 2D grid with multiple layers, heterogeneous regions and defined boundaries.

7.4.4.3. MINC processing for fractured media (keyword 'MINC')

Data Block GRID

The first record in this data block includes only the keyword (character variable) 'MINC' that is invokes post-processing of a primary porous medium mesh from a previously developed file **MESH**. This keyword is read using a Format (A5). The input formats in data block MINC are identical to those of the GMINC program [*Pruess*, 1983], with two enhancements: (a) there is an additional facility for specifying global matrix-matrix connections (*dual permeability* option); (b) only active elements (see Section 8.4) will be subjected to MINC-processing, the remainder of the **MESH** remaining unaltered as porous medium grid blocks. See Section 2.14 for further discussion.

NOTE: *If the application of the* ***MeshMaker.f95*** *aims to process a pre-existing external* ***MESH*** *file, the input file begins with records* ***MESHMAKER.1*** *and* ***MESHMAKER.2*** *(see Section 7.2.1) as the first two data blocks, followed by the data inputs described in detail below.*

Record MINC.1

Format (2A5, 5X, A5)
PART, TYPE, DUAL

PART: This is the first keyword following the 'MINC' keyword. It will be followed on the same line by parameters TYPE and DUAL with information on the nature of fracture distributions and matrix-matrix connections.

PART: An identifier of the data block with partitioning parameters for secondary mesh.

TYPE: A five-character variable for selecting one of the following six different proximity functions provided in MINC [*Pruess*, 1983].

= 'ONE-D': A set of plane parallel infinite fractures with matrix block thickness between neighboring fractures equal to PAR(1).

= 'TWO-D': Two sets of plane parallel infinite fractures, with arbitrary angle between them. Matrix block thickness is PAR(1) for the first set, and PAR(2) for the second set. If PAR(2) is not specified explicitly, it will be set equal to PAR(1).

= 'THRED': Three sets of plane parallel infinite fractures at right angles, with matrix block dimensions of PAR(1), PAR(2), and PAR(3), respectively. If PAR(2) and/or PAR(3) are not explicitly specified, they will be set equal to PAR(1) or PAR(2), respectively.

= 'STANA': Average proximity function for rock loading of Stanford large reservoir model [*Lam et al.,* 1988].

= 'STANB': Proximity function for the five bottom layers of Stanford large reservoir model.

= 'STANT': Proximity function for top layer of Stanford large reservoir model.

DUAL A five-character word for selecting the treatment of global matrix flow.

= ' ' (Blank – default): The global flow occurs only through the fracture continuum, while rock matrix and fractures interact locally by means of interporosity flow (*double-porosity* model).

= 'MMVER': The global matrix-matrix flow is permitted only in the vertical; otherwise like the double-porosity model; for internal consistency this choice should only be made for flow systems with one or two predominantly vertical fracture sets.

= 'MMALL': The global matrix-matrix flow in all directions; for internal consistency only two continua, representing matrix and fractures, should be specified (dual-permeability model).

NOTE: *A user wishing to employ a different proximity function other than the options provided through the* TYPE *variable in* ***MINC*** *needs to replace the function subprogram* PROX(x) *in* **MeshMaker.f95** *with a routine of the form:*

```
FUNCTION PROX(x)
PROX = (arithmetic expression in x)
RETURN
END
```

It is necessary that PROX(x) *be defined even when x exceeds the maximum possible distance from the fractures, in which case* PROX = 1. *Additionally, when the users supply their own proximity function subprogram, the parameter* TYPE *must be set equal to* 'ONE-D', 'TWO-D', *or* 'THRED', *depending on the dimensionality of the proximity function. This will assure proper definition of the innermost nodal distances* [*Pruess,* 1983].

Record PART.l

Format (2I3, A4, 7E10.4)
J, NVOL, WHERE, (PAR(i), i = 1, 7)

J: The total number of multiple interacting continua (J < 36).

NVOL: The total number of explicitly provided volume fractions (NVOL < J). If NVOL < J, the volume fractions with indices NVOL+1, ..., J will be internally generated; all being equal and chosen such as to yield proper normalization to 1.

WHERE: Character variable specifying whether the sequentially specified volume fractions begin with the fractures (WHERE = 'OUT') or in the interior of the matrix blocks (WHERE = 'IN').

PAR(i): Real array that stores the parameters describing the fracture spacing (see discussion of previous record).

Record PART.2.1, PART.2.2, etc.

Format (8E10.4)
(VOL(i), i = 1, NVOL)

VOL(i): The volume fraction (having a value between 0 and 1) of a continuum with index i (for WHERE = 'OUT') or index J+l-i (for WHERE = 'IN'). NVOL volume fractions will be read. For WHERE = 'OUT', i=1 is the fracture continuum, i=2 is the matrix continuum closest to the fractures, i=3 is the matrix continuum adjacent to i=2, etc. The sum of all volume fractions must not exceed 1.

Record GRID-END

Two blank records close the input data file.

8.0. Initial Conditions and Boundary Conditions

In this section the data blocks that allow for domain-specific initial conditions (block **INDOM**), element-specific initial conditions (**INCON**), extended capabilities for specifying initial conditions (**EXT-INCON**), including features for assigning initial conditions by a variety of methods. Following that the procedure is described for implementing initial conditions and boundary conditions in TOUGH+ v1.5.

8.1. Data Block INCON

This block introduces element-specific initial conditions.

Record INCON.1 (for 5-character element names)

Format (A5, 2I5, E15.8, 2x, A3, 36x, 3(E15.8))

```
ElName5C, NSEQ, NADD, porosity, StateIndex, (perm(i), i=1,3)
```

`ElName5C`
The 5-character name of the element that is being initialized.

`NSEQ`
The number of additional elements with the same initial conditions.

`NADD`
The increment between the code numbers of two successive elements with identical initial conditions.

`Porosity`
The porosity of the element that is being initialized. If `porosity` is zero or blank, the element porosity will be taken as specified in block **`ROCKS`** or **`MEDIA`**. This feature is necessary for assignment of element-specific properties for the description of highly or statistically heterogeneous domains.

`StateIndex`
State identifier (see Section 3.1, **Tables 3.1** and **3.2**): the initial conditions corresponding to this identifier are applied uniformly over the element.

`perm(i), i=1,3`
The intrinsic (absolute) permeabilities of the element that is being initialized along the three directions described by the `ConxKi` variable (see Section 7.3). If all `perm(i), i=1,…,3` are zero or blank, the element permeabilities will be taken as specified in block **`ROCKS`** or **`MEDIA`**. This feature is necessary for assignment of element-specific properties for the description of highly or statistically heterogeneous domains.

Record `INCON.1` (for 8-character element names)

Format (A8, 7X, E15.8, 2x, A3, 36x, 3(E15.8))

```
ElName8C, NSEQ, NADD, porosity, StateIndex, (perm(i), i=1,3)
```

`ElName8C` is the 8-character name of the element that is being initialized. All other variables remain as in the case of five-character element names.

Record `INCON.2` specifies primary variables.

Format (6E20.13)
`X(i), i = 1,NumCom+1`

The primary variables defining the state of the element specified in record `INCON.1`. **`INCON`** specifications will supersede default conditions specified in

PARAM.4 (see Section 10), and domain-specific conditions that may have been specified in data block **INDOM**. See Section 3.1 (**Tables 3.1** and **3.2**) for a detailed description of the potential sets of primary variables.

Record INCON.3

There are three ways to close the **INCON** data block.

If the simulation is not a continuation run, then a blank line or a record with '<<<' typed in the first three columns closes the **INCON** data block.

For continuation runs from a previous TOUGH+ v1.5 simulation, the element-related **INCON** data terminate with a record with '**:::**' typed in the first three columns. This is followed by a namelist that includes data describing the origin of time, the simulated time at the conclusion of the preceding TOUGH+ v1.5 simulation and the cumulative numbers of the timesteps and of the Newtonian iterations at the end of the previous run (see **Figure 8.1**). These data are recorded automatically at the end of the **SAVE** file (see Section 8.4) upon completion of the any TOUGH+ v1.5 run using a NAMELIST format. This is a very powerful format that allows maximum clarity and flexibility, accepting free formats, arbitrary ordering of variables, insertions of comments anywhere in the input fields, and providing the option of ignoring any of the NAMELIST parameters by omitting it or by not assigning a value to it. For more information, the reader is directed to a textbook on FORTRAN 95/2003.

```
.
.
.
.
FJ000           0.00000000E+00  Aqu: P,     X_m_A, X_i_A, T
 3.2500495961160E+07 0.0000000000000E+00 3.0000000000000E-02 9.3974540000000E+00
FJ106           0.00000000E+00  Aqu: P,     X_m_A, X_i_A, T
 3.3467136230360E+07 2.9244292873310E-03 3.0000000000690E-02 1.2627840000000E+01
:::
 &Data_For_Continuation_Run
       timesteps_to_this_point     =       6818,
       NR_iterations_to_this_point =      71226,
       number_of_detected_media    =          8,
       origin_of_time              =  0.00000000000000E+00,
       time_to_this_point          =  4.69736464257198E+07,
       accumulated_quantities      =  7.35454091207164E+04,  1.08644220169487E+05,
                                      1.03974248845536E+05,  6.43009774059305E+04,
                                      0.00000000000000E+00,  0.00000000000000E+00,
                                   /
<<<
```

Figure 8.1. An example of the NAMELIST-described termination data printed at the end of the **SAVE** file from a TOUGH+ v1.5 simulation. These data can be read as part of the **INCON** data block, or of the **INCON** external file. The names of the variables defined in the NAMELIST are self-explanatory. For reference, this figure lists the conditions in the last two elements (FJ000 and FJ106) of the grid in the TOUGH+ v1.5 simulation.

8.2. Data Block INDOM

This block introduces domain-specific initial conditions. These will supersede default initial conditions specified in PARAM.4 (see Section 10), and can be overwritten by element-specific initial conditions in data block **INCON** or data block **EXT-I**NCON. The option **START** is needed to use **INDOM** conditions.

Record INDOM.1

Format (A5, 2X, A3)
Rk_name, StateIndex

Rk_name
: The name of a medium (corresponding to a system subdomain), as specified in data block **ROCKS**.

StateIndex
: The state identifier describing the conditions applying to the Rk_name medium/subdomain.

Record INDOM.2

Format (6E20.13)
X(i), i = 1,NumCom+1

X(i) are the primary variables assigned to all grid blocks in the domain specified in record INDOM.1. See Section 3.1 (**Tables 3.1** and **3.2**) for description of the potential sets of primary variables.

Record INDOM.3

A blank record closes the **INDOM** data block.

Repeat records INDOM.1 and INDOM.2 for as many domains as desired. The ordering is arbitrary and need not be the same as in block **ROCKS**.

8.3. Data Block EXT-INCON

This block introduces extended capabilities for specifying initial conditions of subdomains (i.e., groups of element). The user has several options to provide element information (e.g., an element name or number list, location, element sequence, columnar structure, etc.) that defines a subdomain to be initialized with the specified conditions. The first entry is the total number of initialization entities entered within the block. Following that, each of the various input entities may be entered in an arbitrary order.

Record EXT-INCON.0

This record includes the single integer variable Total_input_num that describes the total number of input entities (representing initial conditions in particular subdomains) that will be entered. This is read using a free format.

Record EXT-INCON.1

This record includes the single character variable INTYPE that is read using a free format, and which provides a keyword defining the type of data describing the subdomain to be initialized. INTYPE can assume one of the following values: 'GEOMETRY', 'LIST', 'SEQUENCE', or 'COLUMN'. Thus, INTYPE determines what kind of variables/data will be read in record in EXT-INCON.2.

8.3.1. Data Block GEOMETRY

The data in this block are read when INTYPE = 'GEOMETRY', and define a set of elements in a subdomain that is bounded within prescribed minimum and maximum coordinates. All elements within this range will be assigned the initial conditions entered in the record EXT-INCON.3.

Record EXT-INCON.2.0

For a Cartesian grid (coordinate_system = 'Cylindrical', see Section 5), the following real variables are read in EXT-INCON.2 using a free format:

```
Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
```

These parameters are defined as follows:

`Xmin, Xmax`
: Real variables indicating the range of the subdomain to-be-initialized along the *x*-axis of the Cartesian coordinate system.

`Ymin, Ymax`
: Real variables indicating the range of the subdomain to-be-initialized along the *y*-axis of the Cartesian coordinate system.

`Zmin, Zmax`
: Real variables indicating the range of the subdomain to-be-initialized along the *z*-axis of the Cartesian coordinate system.

For a cylindrical grid (`coordinate_system = 'Cartesian'`, see Section 5), the following real variables are read in `EXT-INCON.2` using a free format:

```
Rmin, Rmax, Zmin, Zmax
```

These parameters are defined as follows:

`Rmin, Rmax`
: Real variables indicating the range of the subdomain to-be-initialized along the *r*-axis of the cylindrical coordinate system.

`Zmin, Zmax`
: Real variables indicating the range of the subdomain to-be-initialized along the *z*-axis of the cylindrical coordinate system.

8.3.2. Data Block **SEQUENCE**

For `INTYPE = 'SEQUENCE'`

Record `EXT-INCON.2.0`

Format (*), i.e., free format

```
SequICFirstElemNum,
SequICLastElemNum,
SequICStride
```

These parameters are defined as follows:

`SequICFirstElemNum`
: Integer describing the global number of the first element in the sequence.

`SequICLastElemNum`
: Integer describing the global number of the last element in the sequence.

`SequICStride`
: Integer describing the stride in the numbering sequence

Thus, a sequence of elements is defined by the beginning and ending element number, as well as by the stride (number of elements to skip between two successive elements in the subdomain defined by the sequence). For example, a sequence with `SequICFirstElemNum = 10`, `SequICLastElemNum = 20`, and `SequICStride = 2` results in the a group (subdomain) of elements with the following global element numbers: 10, 12, 14, 16, 18, 20. If `SequICLastElemNum < SequICFirstElemNum` or if `SequICStride < 0`, an error message is printed and the simulation is aborted.

All elements within this sequence will be assigned the initial conditions entered in the record `EXT-INCON.3`.

8.3.3. Data Block **`LIST`**

For `INTYPE = 'LIST'`

Record `EXT-INCON.2.0`

Format (*), i.e., free format

```
ListICLength, N_per_row
```

These parameters are defined as follows:

`ListICLength`
: Integer denoting the length of the list (i.e., the total number of element numbers in the list) that is about to be read.

`N_per_row`
: Integer defining the number of entries (= element numbers) per row in the list that that is about to be read. The last row may have fewer than `N_per_row` entries.

Record EXT-INCON.2.1

Format (*), i.e., free format

```
ElemNum(i), i=1, ListICLength
```

NOTE: *The number of entries per row of the* `ElemNum(i)` *input data is* `N_per_row`

The element numbers that are defined through their participation in a list are read in rows (= records), each (except possibly the last one) containing `N_per_row` entries. The last row/record may have fewer than `N_per_row` entries. For example, if `ListICLength = 8` and `N_per_row = 3`, then the element numbers would be listed as follows:

```
ElemNum(1),ElemNum(2),ElemNum(3),  ! 1st record
ElemNum(4),ElemNum(5),ElemNum(6),  ! 2nd record
ElemNum(7),ElemNum(8)              ! 3rd record
```

All elements in this list will be assigned the initial conditions entered in the following record (`EXT-INCON.3`).

8.3.4. Remaining Data Blocks in **`EXT-INCON`**

For `INTYPE = 'GEOMETRY'`, `'SEQUENCE'`, or `'LIST'`

Record EXT-INCON.3

Format (*), i.e., free format

```
StateIndex, X0(i),i=1,NumCom+1
```

These parameters are defined as follows:

`StateIndex`
: A character variable indicating the state index (see Section 3) corresponding to the initial conditions that are to be assigned to the subdomain defined by `INTYPE`.

`X0(i),i=1,NumCom+1`
: A real array that includes the primary variables (corresponding to `StateInd`) describing the conditions to which the respective subdomain is initialized. See Section 3.1 for a thorough description of the potential sets of primary variables.

Repeat records `EXT-INCON.3`, `EXT-INCON.2` and `EXT-INCON.3` for a total of `Total_input_num` subdomain-based initializations.

8.3.5. Data Block **COLUMN**

This data block is somewhat different from the previous ones, and is very useful in applying initial conditions in problems that involve initialization after achieving gravity equilibration in a single column of the domain, which then serves as the reference column. Thus, when `INTYPE = 'COLUMN'`, the following records and data are read:

Record `EXT-INCON.2.0`

This record includes the single integer variable `ColmICSize` that is read using a free format, and which describes the column length, i.e., the number of elements in the column.

Record `EXT-INCON.2.1`

Format (*), i.e., free format

```
ColmICFirstElemNum,
ColmICNumElemInCol,
ColmICstride
```

These parameters are defined as follows:

`ColmICFirstElemNum`	An integer variable indicating the global element number of the first element in the columnar structure to be initialized.
`ColmICNumElemInCol`	An integer variable describing the total number of elements to be initialized using the columnar structure. Thus, the total number of columns to be initialized using the data provided in this block is: `ColmICNumElemInCol/ColmICSize`
`ColmICstride`	An integer variable describing the stride in the numbering sequence. This number is the difference between the global numbers of two elements that two successive locations in the same column.

For example, if `ColmICSize=10`, `ColmICFirstElemNum=1`, `ColmICNumElemInCol=80`, and `ColmICstride=1`, then initialization using the `COLUMN` data block will assign the initial conditions (obtained from the 10 elements of the reference column) to columns composed of elements with the

following global numbers: Column #1, elements 1 to 10; Column #1, elements 11 to 20; … Column #8, elements 71 to 80. In this case, elements 1, 11, 21,…,71 have the same initial conditions (equal to those of the first entry in the reference column). Similarly, elements 2, 12, 22, …, 72 have all the same initial conditions (equal to those of the second entry in the reference column).

Conversely, if `ColmICstride=1`, then elements 1 to 8 have all the same initial conditions (equal to those of the first entry in the reference column), 2 to 16 have all the same initial conditions (equal to those of the second entry in the reference column), etc.

Record `EXT-INCON.3.x, x=1,…,ColmICSize`

Format (*), i.e., free format

```
StateIndex, (X0(i),i=1,NumCom+1)
```

A total of `ColmICSize` records are read, each providing the state index and primary variables of the elements of the reference column. The variables in these records are as previously defined (see Section 8.3.4).

8.4. Implementing Initial Conditions

Flow systems are initialized by assigning a complete set of primary thermodynamic variables to all grid blocks into which the flow domain is discretized. Various options are available in a hierarchical system, as follows. During the initialization of a TOUGH+ run, all grid blocks are first assigned to default thermodynamic conditions specified in data block **`PARAM`**. The defaults can be overwritten for selected reservoir domains by assigning domain-specific conditions in data blocks **`INDOM`** or **`EXT_INCON`**. These in turn may be superseded by thermodynamic conditions assigned to individual grid blocks in data block **`INCON`**. A disk file **`INCON`** written to the same specifications as data block **`INCON`** may also be used.

The possible sets of primary variables are discussed in Section 3.1 (**Tables 3.1** and **3.2**), with the actual primary variables depending on the fluid/solid phase composition. During phase change primary variables will be automatically switched from one set to another. In multiphase flow systems, therefore, different grid blocks will in general have different sets of primary variables, and must be initialized accordingly.

For many applications, special initial conditions are needed, such as gravity-capillary equilibrium, or steady state corresponding to certain mass and heat flows. This can be realized by performing a series of TOUGH+ runs, in which thermodynamic conditions obtained in one run, and written to disk file **SAVE**, are used as initial conditions in a subsequent continuation run. For example, in a hydrate accumulation simulation, a first run may be made to obtain hydrostatic pressure conditions. These may subsequently be used as boundary conditions in a second run segment to simulate undisturbed natural state conditions with through-flow of mass and heat. This could be followed by a third run segment with fluid production and injection.

Restarting of a TOUGH+ run is accomplished by renaming the file **SAVE** generated in a previous run as file **INCON** for initialization. Usually additional (often minor) adjustments will be made for a restart. For example, different specifications for the number of time steps and desired printout times may be made. Some editing of the **MESH** file may be needed to make certain grid blocks inactive, so that previously calculated pressures can serve as boundary conditions (see below). In a continuation run, simulation time and time step counters may be continuously incremented, or they may be reset to zero. For example, the latter option will be used when simulating production and injection

operations following preparation of a natural initial state, which may correspond to a large simulation time.

As far as the internal workings of the code is concerned, there is no difference between a fresh start of a simulation and a restart. The only feature that makes a simulation a continuation run is that the **INCON** data were generated by a previous TOUGH2 or TOUGH+ run, rather than having them explicitly provided by the user.

The file **SAVE** originating from TOUGH2 simulations [*Pruess et al*., 1991; 2012] always ends with a data record with '+++' in the first three columns, followed by one record with restart information (time step and iteration counters, simulation time). The file **SAVE** originating from a TOUGH+ simulation terminates with continuation data written using the `NAMELIST` format shown in **Figure 8.1**. In either case, to reset all counters and continuation data to zero when using **SAVE** as file **INCON** for another TOUGH+ run, users can simply replace all records below the conditions of the last element with a single blank record.

8.5. Implementing Boundary Conditions

8.5.1. General

Boundary conditions can be of two basic types. Dirichlet conditions prescribe thermodynamic conditions, such as pressure, temperature, etc. on the boundary, while Neumann conditions prescribe fluxes of mass or heat crossing boundary surfaces. A special case of Neumann boundary conditions is *no flux*, which is the default in the integral finite difference framework when no flow connections are specified across the

boundary. More general flux conditions are prescribed by introducing sinks or sources of appropriate strength into the elements adjacent to the boundary.

In TOUGH2 [*Pruess et al*., 1999; 2012], Dirichlet conditions could be implemented by assigning very large volumes (e.g., $V = 10^{50}$ m^3, as described by the `elem_vol` variable in Section 7.2) to grid blocks adjacent to the boundary, so that their thermodynamic conditions do not change at all from fluid or heat exchange with finite-size blocks in the flow domain. In addition, a small value (such as 10^{-9} m) should be specified for the nodal distance (`ConxD1` or `ConxD2`, see Section 7.3) of such blocks, so that boundary conditions are in fact maintained in close proximity to the surface where they are desired, and not at some distance from it. It is possible to specify a nodal distance that is outright zero; however, this may interfere with options for the computation of interface mobilities that are intended for modeling fracture-matrix interactions. Therefore, assigning zero nodal distances (`ConxD1` or `ConxD2`) should be used with caution.

For time-independent Dirichlet boundary conditions, TOUGH2 [*Pruess et al*., 1999; 2012] offered an alternative implementation, which provided savings in computational work along with added user conveniences in running simulation problems. This was accomplished by defining *active* and *inactive* elements. By convention, elements encountered in data block **`ELEME`** (or in geometry files **`MESH`** or **`MINC`**) were taken to be active until the first element entry with a zero or negative volume was encountered. The first element with volume `elem_vol ≤ 0.0E0`, and all subsequent elements, were by convention taken to be inactive. The easiest way to declare selected grid blocks as inactive was to use a text editor to move them to the end of the **`ELEME`** data block, and then insert a dummy grid block of zero volume in front of them.

TOUGH+ v1.5 maintains these two older options available in TOUGH2, but provides an additional one that is much simpler as it does not require any editing of the element volume or physically moving any portions of the element list. In TOUGH+ v1.5, elements can be designated as inactive if the parameter `element_activity` in record `ELEME.1` (see Section 7.2) is set to `'I'` (when its conditions are time-invariant) or `'Vxx'` (when its conditions and properties vary over time). Then, these elements, as well as all elements designated as inactive by the older TOUGH2 options, are treated as inactive.

For the inactive elements no mass or energy balance equations are set up, their primary thermodynamic variables are not included in the list of unknowns, and their thermodynamic conditions remain unchanged during the course of the simulation. Inactive elements can appear in flow connections and initial condition specifications like all other elements. The computational overhead of inactive elements is small because they do not increase the number of equations to be solved in a flow problem.

8.5.2. Data Block **BOUNDARIES**

This data block provides time-variable boundary conditions that are applicable to gridblocks with an activity indicator `elem_activity` = '`Vxx`' (xx is a number, see Section 7.2). When this data block is present in the input data file, the time-variable boundary conditions are read in a tabular form.

The data in this record are read using a `NAMELIST` format, and may occupy one or more lines (a choice left to the user). As already discussed (Section 8.1), `NAMELIST`-based formats are a feature of FORTRAN 95/2003 and provide unique power and

flexibility, allowing (a) assignment of updated values to any subset of the parameters included in the NAMELIST definition, (b) arbitrary order, (c) free formats of individual parameter values, (d) inclusion of comments, etc. Future versions of TOUGH+ will involve NAMELIST-based formats to read most input data.

The following records and data are read in the 'BOUNDARIES' data block:

Record BOUNDARIES.1

The namelist in this record is named Transient_Boundaries, has the following general form

```
&Transient_Boundaries  number_of_defined_boundaries = x,
                       num_table_defined_boundaries = x /
```

and includes the following variables:

number_of_defined_boundaries:
An integer variable describing the number of time-variable boundaries that are to be described

num_table_defined_boundaries:
An integer variable describing the number of tables that are to be read, which describe the time-variable behavior of these boundaries.

The variable number_of_defined_boundaries can be equal or smaller than num_table_defined_boundaries. However, the current TOUGH+ v1.5 version can only accommodate tabular data in the BOUNDARIES.1 block, requiring that:

number_of_defined_boundaries = num_table_defined_boundaries

Record BOUNDARIES.2

This record includes general data defining the boundary. A total of num_table_defined_boundaries such records need to be provided. The namelist in this record is named Transient_Boundary_Definition, has the form

```
&Transient_Boundary_Definition boundary_number = x,
                               boundary_name = 'x',
                               num_PV_to_read = x,
                               PV_numbers = x,
                               data_form = 'x'/
```

and includes the following parameters:

`boundary_number:`
: Integer parameter describing the number of the boundary (more than one boundaries can be defined).

`boundary_name:`
: The name of of the boundary corresponding to `boundary_number`.

`num_PV_to_read:`
: Integer parameter describing the number of the primary variables to be used from among the data read from the table. More than one primary variables can be used to define a boundary, and the maximum number is `num_PV_to_read = NumEqu`.

`PV_numbers:`
: The number of the primary variables (as listed in the order of the primary variables in the EOS of the User's Manuals of the individual TOUGH+ application options) corresponding to each column of the tabular data.

`data_form:`
: This character parameter (5 characters long) describes the type of the data defining the boundary. In the current version of TOUGH+, the only option is `data_form = 'Table'`.

The structure of the `Transient_Boundary_Definition` namelist (and its use as an input format in the data block **`BOUNDARIES`**) is best illustrated in the example of **Figure 8.2**.

Record `BOUNDARIES.3`

This record includes specific data describing the tabular data of the conditions at the boundary. A total of `num_table_defined_boundaries` such records need to be provided. The namelist in this record is named `Tabular_Data_Definition`, has the form

```
&Tabular_Data_Definition read_by_row_or_by_col = x,
                         number_of_rows        = 'x',
                         tot_number_of_columns = x,
                         RowColNum_to_PrimVarNum = x,…,x,
                         units_of_PrimVar = 'x',…,'x',
                         read_format      = 'x'/
```

and includes the following parameters:

`read_by_row_or_by_col:`
: Character parameter of length 3 describing if the tabular data are to be read by row or by column. The following self-explanatory options are available:

='Row': The data are read by row. Each row includes the time and the corresponding primary variable value(s); each column corresponds to a single time.

='Col': The data are read by column. Each column lists the entire series of time or of one or more primary variables.

number_of_rows:
Integer parameter defining the number of data points.

tot_number_of_columns:
Integer parameter describing the number of the number of columns to be read, including both the time and the boundary primary variable columns (must be > 1).

units_of_PrimVar:
A character array of length 3 and of dimension NumEqu+1 that describes the units of time and of the primary variables in the table.

read_format:
This character parameter (up to 120 characters long) describes the format to read the tabular data.

The structure and use of the namelists in the data block **BOUNDARIES** is best illustrated by the example of **Figure 8.2**.

Record BOUNDARIES.4
This record includes a comment (a character variable of 200 characters) that provides some information on the table to follow.

Record BOUNDARIES.5.1, 5.2, 5.3, …, number_of_rows
Each one of those records includes tot_number_of_columns individual data points of the table that are read using the format specified by read_format.

The structure and use of the data block **BOUNDARIES** is best illustrated by the example of **Figure 8.2**.

```
BOUNDARIES
&Transient_Boundaries   number_of_defined_boundaries = 1,
                        num_table_defined_boundaries = 1
                        /
  &Transient_Boundary_Definition   boundary_number = 1,
                                   boundary_name   = 'V01'',
                                   num_PV_to_read  = 2,
                                   PV_numbers      = 1,3,
                                   data_form = 'Table'
                                   /
    &Tabular_Data_Definition  read_by_row_or_by_col    = 'row',
                              number_of_rows           = 3,
                              tot_number_of_columns    = 4,
                              RowColNum_to_PrimVarNum  = 3,-1,0,1,
                              units_of_Primvar(0) ='sec',
                              units_of_Primvar(1) = 'Pa',
                              units_of_Primvar(3) = 'C',
                              read_format         = '*'
                              /
'The PV are in the sequence: Temperature, dummy, time, dummy'
     12.5e0  -1222.34     0.0e0  9.80e6
     14.0e0  -1.48e03     3.0e3  8.80e6
     17.0e0  -1.77e03     9.0e3  8.00e6
```

Figure 8.2. An example of the namelist structure of the BOUNDARIES data block. The time-variable data are provided and read in tabular form.

9.0. Sources and Sinks

9.1. Data Block GENER

This block introduces sinks and/or sources to the system

Record GENER.1 (for 5-character element names)

Format (A5, A5, 4I5, 5X,A4, A1, 3E10.4 ,A4, 6x, 3(E10.4))

```
ElName5C, SS_name, NSEQ, NADD, NADS, LTAB,
SS_Type, ITAB, GX, EX, HX,
WellResponse, PresLimits,
RateStepChange, RateLimit
```

These parameters are defined as follows:

ElName5C:
The code name of the 5-character element containing the sink/source.

SS_name:
The name of the sink/source. The first three characters are arbitrary, the last two characters must be numbers.

NSEQ: The number of additional sinks/sources with the same injection/production rate (not applicable for SS_Type = 'DELV').

NADD: The increment between the code numbers of two successive elements with identical sink/source.

NADS: The increment between the code numbers of two successive sinks/sources.

LTAB: The number of points in table of generation rate versus time. Set 0 or 1 for constant generation rate. For wells on deliverability, LTAB denotes the number of open layers, to be specified only for the bottommost layer.

SS_Type:
The type of source or sink. This variable specifies different options for fluid or heat production and injection. For example, different fluid components may be injected, the nature of which depends on the EOS module being used. Different options for considering wellbore flow effects may also be specified. The following options are available:

- ='HEAT': Introduces a heat sink/source. This option is to be used for injection only.
- ='COM1': Indicates mass component #1 (usually water). This option is to be used for injection only.
- ='WATE': Indicates water injection.
- ='COM2': Indicates mass component #2. This option is to be used for injection only.
- ='COM3': Indicates mass component #3. This option is to be used for injection only.
- ='COMn': Indicates mass component #n. This option is to be used for injection only.
- ='MASS': Specified a mass production rate, i.e., the mass rate of all the fluids withdrawn from a system.

ITAB: Unless left blank, a table of specific enthalpies will be read (LTAB > 1 only).

GX: The constant generation rate. GX is positive for injection and negative for production. GX describes a mass rate [kg/sec] for generation types COM1, COM2, COM3, etc., and MASS; it describes an energy rate [W] for a HEAT sink/source. For wells on deliverability, GX is the productivity index PI [m^3] – see Equation (9.2).

EX: The fixed specific enthalpy [J/kg] of the fluid for mass injection (GX>0). For wells on deliverability against fixed bottomhole pressure, EX is the

bottomhole pressure P_{wb} [Pa] at the center of the topmost producing layer in which the well is open.

`HG`: The thickness of layer [m]. This is to be used only in cases of wells on deliverability with specified bottomhole pressure.

`WellResponse:`
A character variable that describes how the source/sink is to be treated if pressure limits (as described by the variable `PresLimits`) at the corresponding cell are violated. The following options are available:

`='STOP'`: The simulation is halted.
`='ZERO'`: The source/sink rate `GX` is reset to zero, and the simulation continues.
`='ADJU'`: The source/sink rate `GX` is adjusted, and the simulation continues. Note that a simulation involving the `WellResponse = 'ADJU'` option may involve several successive `GX` adjustments.

`PresLimits:`
A real variable describing the pressure limit [Pa] that acts as a trigger for `WellResponse` to be enacted. For injection, pressure in the element containing the source has to exceed `PresLimits` for changes to be implemented. For production, pressure in the element containing the sink has to fall below `PresLimits` for changes to be implemented.

`RateStepChange:`
A real variable that is used only when `WellResponse = 'ADJU'`. By convention, `RateStepChange` is always a positive number. It represents the absolute value of the fraction by which `GX` is to decrease when the `PresLimits` criterion is violated. For obvious reasons, $0 \leq$ `RateStepChange` ≤ 1.

`RateLimit:`
A real variable describing the minimum rate limit, described as the lowest fraction of the original `GX` to which the rate is allowed to decline. The simulation stops when `RateLimit` is reached.

Record `GENER.1` (for 8-character element names)

Format (A8, A5, 12X, I5, 5X,A4, A1, 3E10.4 ,A4, 6x, 3(E10.4))

```
ElName8C, SS_name, LTAB,
SS_Type, ITAB, GX, EX, HX,
WellResponse, PresLimits,
```

```
RateStepChange, RateLimit
```

Here `ElName8C` is the name of the 8-character element containing the sink/source. All other variables and parameters are as in the case of the 5-character elements.

Record `GENER.1.1` (**Optional**, `LTAB > 1` only)

Format (4E14.7)
`F1(k), k=1, LTAB`

`F1(k)`: Generation times in the table of the time variable source/sink data.

Record `GENER.1.2` (**Optional**, `LTAB > 1` only)

Format (4E14.7)
`F2(k), k=1, LTAB`

`F2(k)`: Generation rates in the table of the time variable source/sink data.

Record `GENER.1.3` (**Optional**, `LTAB > 1` and `ITAB` non-blank only)

Format (4E14.7)
`F3(k), k=1, LTAB`

`F3(k)`: Specific enthalpy of produced or injected fluid in the table of the time variable source/sink data.

Repeat records `GENER.1`, `GENER.1.1`, `GENER.1.2`, and `GENER.1.3` for the desired number of sinks/sources.

Record `GENER.2`

A blank record closes the **`GENER`** data block. Alternatively, generation information may terminate on a record with '+++' typed in the first three columns, followed by data describing the numbers of the elements where the sources/sinks are located.

9.2. Discussion on sinks and sources

Sinks and sources are introduced through data block **GENER** in the input file. Several options are available for specifying the production ($q < 0$) or injection ($q > 0$) of fluids and heat. Any of the mass components may be injected in an element at a constant rate, or at time-dependent rates that may be prescribed through user-defined tables. The user has the option of specifying the specific enthalpy of the injected fluid as either a constant or time dependent value. Heat sources/sinks (with no mass injection) may be either constant or time-dependent.

Fluid production from an element may be handled by prescribing a constant or time-dependent mass rate. In this case, the phase composition of the produced fluid may be determined by the relative phase mobilities in the source element. Alternatively, the produced phase composition may be specified to be the same as the phase composition in the producing element. In either case, the mass fractions of the components in the produced phases are determined by the corresponding component mass fractions in the producing element. Different options are available for interpolating time-dependent rates from user-supplied tabular data; these may be selected through parameter `MOP(12)` – see Section 10.

PAGE LEFT INTENTIONALLY BLANK

10. Computational Parameters

In this section the data blocks (**PARAM** and **SOLVR**) that specify computational parameters are described and followed by a discussion on linear equation solvers.

10.1. Data Block PARAM

This block describes an assortment of computation parameters.

Record PARAM.1

Format (2I2, 3I4, 24I1, 3E10.4, 2I5)

```
Max_NumNRIterations, OutputOption,
Max_NumTimeSteps, iCPU_MaxTime,
PRINT_frequency, (MOP(i),i=1,24),
BaseDiffusionCoef, DiffusionExpon,
DiffusionStrength,
SAVE_frequency, TimeSeries_frequency
```

These parameters are defined as follows:

`Max_NumNRIterations`:
Integer specifying the maximum number of Newtonian iterations per time step (default is 8)

`OutputOption:`
Integer defining an option that controls the amount of printout in the standard TOUGH+ output (the default is 1). The following options are available:

`= 0,1:` Print a selection of the most important variables.

`= 2:` In addition, print the mass and heat fluxes and the flow velocities.

`= 3:` In addition, print the primary variables and their changes.

If the above values for `OutputOption` are increased by 10, printout will occur after each Newton-Raphson iteration (not just after convergence).

`Max_NumTimeSteps`:
An integer defining the maximum number of time steps allowed during the simulation.

NOTE: *If* `Max_NumTimeSteps < 0`, *then the maximum number of timesteps becomes* `1000*ABS(Max_NumTimeSteps)`

`iCPU_MaxTime`:
An integer describing the maximum duration, in CPU seconds, of the simulation (default is infinite).

`PRINT_frequency`:
An integer describing the printout frequency. This, printout will occur for every multiple of `PRINT_frequency` steps (the default is `1`).

NOTE: *If* `PRINT_frequency < 0`, *then the printout frequency becomes* `1000*ABS(PRINT_frequency)`

`MOP(i), i=1,24`:
An integer array that allows choice of various computational options. These are described in detail below, and are documented in the printed standard output from a TOUGH+ v1.5 run.

`MOP(1):`
If `MOP(1)` ≠ `0`, a short printout for non-convergent iterations will be generated at the end of each Newton-Raphson iteration.

`MOP(2)` to `MOP(6)`:
If ≠ `0`, these options generate additional printout in various subroutines at the end of each Newton-raphson iteration. This feature is not needed in

normal applications, but can be convenient in the development of new capabilities as they can be used to identify coding errors. The amount of printout increases with the value of MOP(i). The user is encouraged to consult the source code listings for details. Below we list the subroutines corresponding to MOP(2) to MOP(6):

MOP(2):
Simulation_Cycle (executive subroutine that advances time and controls the flow of data during the simulation)

MOP(3):
JACOBIAN_SetUp (routine computing the flow and accumulation terms in the mass and energy balance equations).

MOP(4):
SourceSink_Equation_Terms (subroutine determining the contribution of sinks/sources to the mass and energy balance equations).

MOP(5):
Equation_Of_State (routine describing the equation of state of the system under investigation and computing all thermophysical properties).

MOP(6):
Solve_Jacobian_Matrix_Equation (routine that solves the linear equations of the Jacobian matrix).

MOP(7):
If MOP(7) $\neq$ 0, a printout of the input data is provided in the standard output file.

MOP(8):
It determines how relative permeability and capillary pressure are estimated in the presence of solid phases (see discussion of Section 2.12). The following options are available:

=0: Based on the OPM model; capillary pressure scaling based on EPM #1.

=1: Based on EPM #1 model; capillary pressure scaling based on EPM #1.

=2: Based on EPM #2 model; capillary pressure scaling based on EPM #2.

=3: Based on EPM #1 model, no capillary pressure scaling.

=4: Based on EPM #2 model, no capillary pressure scaling.

=9: Based on OPM model, no capillary pressure scaling.

MOP(9):

It determines the composition of produced fluid with the MASS option - see discussion on the data block **GENER** in Section 9. The relative amounts of phases are determined as follows:

= 0: according to relative mobilities in the source element.

= 1: the produced source fluid has the same phase composition as the producing element.

MOP(10):

It controls the selection of the interpolation formula for the composite heat conductivity as a function of the various phase saturations. The following options are available

= 0: $\bar{k}_\theta = k_{\theta d} + \sqrt{S_A}\left(k_{\theta w} - k_{\theta d}\right) + \phi S_I k_{\theta I}$

= 1: $\bar{k}_\theta = k_{\theta d} + S_A\left(k_{\theta w} - k_{\theta d}\right) + \phi S_I k_{\theta I}$

= 2: $\bar{k}_\theta = k_{\theta d} + \phi \sum_{\substack{\beta=1,\ldots,N_\beta \\ \beta \neq G}} S_A k_{\theta\beta}$

= 3: $\bar{k}_\theta = k_{\theta d} + \phi \sum_{\beta=1,\ldots,N_\beta} S_A k_{\theta\beta}$

Options MOP(10)=0 and MOP(10)=1 are based on extensions of an earlier model of *Somerton et al*. [2003; 2004] and are applicable to two-phase systems, in which S_A represents the saturation of the wetting phase (not necessarily the aqueous phase). It is not known under what conditions (if any) the linear model of *Bejan* [1984] (invoked for MOP(10)= 2 and 3, indicating ignoring and accounting for the gas contribution) is applicable, but it is included for completeness. The option MOP(10)= 3 is discouraged because of (a) doubts about the validity of the *Bejan* [1984] linear model, (b) the very demanding computations for the estimation of the gas thermal conductivity from the real gas property package in TOUGH+ v1.5 is demanding, and (c) the small overall contribution to the composite thermal conductivity.

MOP(11):

It provides alternative options for the evaluation of mobility and permeability at interfaces. These are:

= 0: The mobilities are upstream weighted according to the W_upstream factor (see discussion in PARAM.3), and the permeability is upstream-weighted.

= 1: The mobilities are averaged between adjacent elements, and the permeability is upstream-weighted.

= 2: The mobilities are upstream weighted, and the permeability is harmonic-weighted.

= 3: The mobilities are averaged between adjacent elements, and the permeability is harmonic-weighted.

= 4: The mobility and permeability are both harmonic weighted.

For multiphase flow simulations in which the upstream element is not known *a priori*, MOP(11) = 0 or MOP(11) = 2 are the recommended options. The user is **strongly** cautioned against using other options.

MOP(12):

It determines the interpolation procedure of the tabular data (involving times, flow rates and enthalpies, see Section 8) in time-dependent sources and sinks. The following options are available:

= 0: triple linear interpolation; tabular data are used to obtain interpolated rates and enthalpies for the beginning and end of the time step; the average of these values is then used.

= 1: step function option; rates and enthalpies are taken as averages of the table values corresponding to the beginning and end of the time step.

= 2: rigorous step rate capability for time dependent generation data.

A set of times t_i and generation rates q_i provided in data block **GENER** is interpreted to mean that sink/source rates are piecewise constant and change in discontinuous fashion at table points. Specifically, generation is assumed to occur at constant rate q_i during the time interval $[t_i, t_{i+1})$, and changes to q_{i+1} at t_{i+1}. The actual rate used during a time step that ends at time t, with $t_i \le t \le t_{i+1}$, is automatically adjusted in such a way that total cumulative exchanged mass at time t

$$Q(t) = \int_0^t q\,dt = \sum_{j=1}^{i-1} q_j (t_{j+1} - t_j) + q_i (t - t_i)$$

is rigorously conserved. If tabular data for enthalpies are also provided, an analogous adjustment is made to fluid enthalpy to preserve $\int q_h\,dt$.

MOP(13):

Option used in processes involving mechanical dispersion.

MOP(14):

It specifies the handling of gas solubility in liquid phases according to one of the following options:

= 0: The gas solubility is computed using appropriate equations of Henry's dissolution parameters (*T*-dependent).

> 0: The gas solubility is computed using fugacities (*T*- and *P*-dependent)

For low *P* and *T* variations, it is possible to use the MOP(14)=0 option, which leads to the use of a *P*- and *T*-invariant Henry's constants for the estimation of solubility. When MOP(14)=0, solubility is computed from fast parametric equations that describe the effect of temperature. For MOP(14)>0, gas solubility is computed from fugacities and activity coefficients. This option is not activated for all TOUGH+ v1.5 application options; in several application options, it is not possible to enable the computationally intensive MOP(14)>0 option because it is excessive and unnecessary.

MOP(15):

A flag indicating whether conductive heat exchange with impermeable confining layers (see Section 7.4) is to be considered.

= 0: The heat exchange is not considered.

= 1: The heat exchange is activated (for grid blocks that have a non-zero heat transfer area; see data block **ELEME** in Section 7.2).

MOP(16):

It provides automatic time step control. Time step size will be doubled if convergence occurs within ITER ≤ MOP(16) Newton-Raphson iterations. It is recommended to set MOP(16) in the range of 2 - 5. The default value is 4.

MOP(17):

It specifies the handling of binary gas diffusivities according to one of the following options:

= 0: The binary gas diffusivity is computed from the method of *Fuller et al.* [1969].

= 7: The basic estimate of the binary gas diffusivity is computed from the method of *Fuller et al.* [1969], and is then adjusted for high pressures using the method of *Riazi and Whitson* [1993].

MOP(18):

A flag determining the method for estimating interface density.

= 0: Perform upstream weighting for interface density.

> 0: Use the average interface density between the two grid blocks. However, when one of the two phase saturations is zero, upstream weighting is to be performed.

MOP(19):

This is the parameter that controls the simulation output. When MOP(19)<8, a standard ASCII file output (as described in general terms in all the User's Manuals of the individual TOUGH+ v1.5 application options) is produced. Depending on the value of the OutputOption parameter in block PARAM.1, this output can includes pressure, temperature and saturation distribution of the various phases, concentrations, thermophysical properties, and primary and secondary variables.

When MOP(19)=8, an additional file containing the most important properties are also printed in a format that conforms to the requirements of the TecPlot package [*TecPlot*, 2003], and is suitable for most other plotting and graphing packages. The name of this file is Plot_Data_Elem, and it stores the element-specific properties and parameters for plotting and graphing. For MOP(19)=9, the plotting files and a truncated standard output file are produced (listing only mass balances at the prescribed printout times).

MOP(20):

Flag determining whether the validity of the initial conditions is to be checked. The options are:

< 9: The initial conditions are checked to ensure physically meaningful and non-contradictory state indices and the corresponding primary variable values (default).

= 9: No checking of initial conditions is performed.

This option may be useful in continuation runs involving large grids, in which case checking of the initial conditions (as provided by the **SAVE** file) is both time consuming (in terms of computation time) and generally unnecessary. In general, the user is discouraged from bypassing the checking process.

MOP(21):

A computational parameter to selects the linear equation solver (see Section 10.2) from among the following options:

= 0: defaults to MOP(21) = 3.

= 1: LUBAND, banded direct solver using LU decomposition.

= 2: DSLUBC, bi-conjugate gradient solver with preconditioner.

= 3: DSLUCS (default), Lanczos-type preconditioned bi-conjugate gradient solver with preconditioner.

= 4: DSLUGM, generalized minimum residual preconditioned conjugate gradient solver with preconditioner.

= 5: DLUSTB, stabilized bi-conjugate gradient solver with preconditioner.

All conjugate gradient solvers use incomplete LU-factorization as a default preconditioner. Other preconditioners may be chosen by means of the data block **SOLVR** (see Section 10.2).

MOP(22), MOP(23), MOP(24):
The function of these variables are described in the individual User's Manuals of the TOUGH+ v1.5 application options (EOS) in which they are active/enabled.

BaseDiffusionCoef:
The base gas diffusion coefficient [m^2]

DiffusionExpon:
Parameter (exponent) describing the temperature dependence of gas phase diffusion coefficient – see Equation (6.4).

DiffusionStrength:
Parameter (optional) describing the effective strength of enhanced vapor diffusion; if set to a non-zero value, it will replace the parameter group $\phi\tau_0\tau_\beta$ for vapor diffusion – see Equations (2.15) and (2.17), and Section 6.4.1.

SAVE_frequency:
Frequency of writing and saving the **SAVE** file. This feature avoids data loss if the simulation is interrupted. A value between 100 and 500 is recommended. When SAVE_frequency = 0, the **SAVE** file is written and stored only once at the conclusion of the simulation.

TimeSeries_frequency:
Frequency of writing and saving the data in the various time-series output files tracking subdomains, interfaces and/or source-sink (well) groups (see Sections 10.3 to 10.5). This feature avoids the creation of very large time series files in long simulations that involve many thousands of time steps. Note that mass-balance related calculations related to some of the parameters included in the time series output files are conducted at the conclusion of each time step, but the results are printed only when specified by this parameter. The default value of TimeSeries_frequency = 1.

Record PARAM.2

Format (4E10.4, A5, 5X, 3E10.4)

```
TimeOrigin, SimulationTimeEnd,
InitialTimeStep, MaxTimeStep, TrackElemName,
gravity, Dt_reducer, scale
```

These parameters are defined as follows:

`TimeOrigin`:
A real variable indicating the origin of time (starting time) in the simulation [sec]. The default is `TimeOrigin = 0.0E0`.

`SimulationTimeEnd`:
A real variable indicating the time [sec] at which simulation should stop. The default is infinite.

`InitialTimeStep`:
A real variable specifying the initial time step size [sec]. If `InitialTimeStep<0`, then the program proceeds to read `NumDts = INT(ABS(InitialTimeStep))` records with time step information.

`MaxTimeStep`:
A real variable defining the upper limit for time step size [sec]. The default is infinite.

`TrackElemName`:
A character variable providing the name of an element, the behavior of which is to be tracked over time by printing a short printout of the evolution of its key conditions and properties after each time step.

`Gravity`:
A real variable specifying the magnitude [m/sec^2] of the gravitational acceleration vector. Blank or zero gives "no gravity" calculation.

`Dt_reducer`:
A real variable defining the factor by which time step is reduced in case of convergence failure or other problems. The default value is `4`.

`Scale:`
The scale factor (a real variable) by which the size of the mesh is adjusted (default = 1.0).

Record PARAM.3

Format (7E10.4, 2X, A3, 3E10.4)

```
rel_convergence_crit, abs_convergence_crit,
U_p, W_upstream, W_NRIteration,
derivative_increment, W_implicitness,
DefaultStateIndex, P_overshoot, T_overshoot,
S_overshoot
```

These parameters are defined as follows:

`rel_convergence_crit`:
Convergence criterion for relative error (real variable, parameter ε_1, see Equation (3.8), default = 10^{-5}).

`abs_convergence_crit`:
Convergence criterion for absolute error (real variable, parameter ε_2, see discussion of Equation (3.9), default = 1).

`U_p`: Not used in TOUGH+ v1.5; maintained only to ensure compatibility with older TOUGH2 [Pruess et al., 1999] input files.

`W_upstream`:
The upstream weighting factor (real variable) for computing mobilities and enthalpies at interfaces. The default `W_upstream` = 1.0 is strongly recommended for multi-phase flows ($0 \le$ `W_upstream` ≤ 1).

`W_NRIteration`:
A weighting factor $0 <$ `W_NRIteration` ≤ 1 (real variable) determining the level of updating of the solutions based on the results of the Newton/Raphson iteration. The default `W_NRIteration` = `1.0E0` is recommended.

`derivative_increment`:
The increment factor (a real variable) for numerically computing derivatives. The default value is `derivative_increment` = $10^{-m/2}$, where m, evaluated internally, is the number of significant digits of the floating point processor used. For 64-bit arithmetic, `derivative_increment` $\approx 10^{-8}$.

`W_implicitness`:
A weighting factor $0 <$ `W_implicitness` ≤ 1 (a real variable) describing the level of implicitness in the solutions. The default `W_implicitness` = `1.0` is recommended.

`DefaultStateIndex`:
The default state identifier (a character variable) of the general initial conditions that apply uniformly over the entire if not amended by the data blocks/files describing initial conditions in the domain (see Section 8).

`P_overshoot`:

A real variable specifying the level of *overshoot* (defined as a fraction) allowed in the computation of pressure if *P* is used as a criterion for triggering phase and state changes. When `P_overshoot = 0.0e0` (default), the *P*-triggered phase and state changes are at their most accurate and sensitive (*hair-trigger*).

This variable is introduced to alleviate potential problems caused by narrow oscillations about phase equilibrium lines that are possible under certain conditions. In most cases, a value of $10^{-6} \leq$ `P_overshoot` $\leq 10^{-4}$ is sufficient if a hair trigger causes problems. If `P_overshoot` < 0 or `P_overshoot` $\geq 5.0 \times 10^{-2}$, it is rest internally to its default value (=0).

`T_overshoot`:

A real variable specifying the level of *overshoot* (defined as a fraction) allowed in the computation of temperature if *T* is used as a criterion for triggering phase and state changes. The definitions, defaults, limits and application are entirely analogous to those of `P_overshoot`.

`S_overshoot`:

A real variable specifying the level of *overshoot* (defined as a fraction) allowed in the computation of saturations if thearese used criteria for triggering phase and state changes. The definitions, defaults, limits and application are entirely analogous to those of `P_overshoot`.

Record `PARAM.4`

This record introduces a set of primary variables that are used as default initial conditions for all grid blocks that are not assigned by means of data blocks **`INDOM`**, **`INCON`** or **`EXT-INCON`**. The format and date read here are:

Format (6E20.13)

```
default_initial_cond(i), i=1,NumCom+1
```

As is self evident, the real variables `default_initial_cond(i)` describe the initial conditions of the state defined by `StateIndex`, as defined by the corresponding primary variables. When more than six primary variables are needed, more than one line (record) must be provided. See Section 3.1 for description of potential sets of state indices and primary variables.

10.2. Modification of Computational Parameters During the Course of a TOUGH+ Simulation

It is possible to modify the computational parameters described in Section 10.1 in the course of a TOUGH+ v1.5 simulation without having to interrupt the execution. This feature is particularly useful in simulations involving large grids and a large number of timesteps (`Max_NumTimeSteps`) when the user observe solution convergence and time advancement that can be improved by varying some of these computational parameters.

The process is controlled by the parameter `SAVE_frequency` (see Section 10.1, record `PARAM.1`) that determines the frequency of updating the **`SAVE`** file. At the time of updating the **`SAVE`** file, TOUGH+ also interrogates the directory of execution for the presence of a file called **`Parameter_Update_File`**. If no such file exists, there is no updating of the computational parameters.

If a file **`Parameter_Update_File`** exists, then it is opened and the following data are read:

Record `UPDATE.1`

This record includes the single character variable `UpdateHeader` that is read using a free format. The TOUGH+ computational parameters are updated only when `UpdateHeader = 'Update_Simulation_Parameters'`. Otherwise, the **`Parameter_Update_File`** is closed and the simulation continues without any parameter updating.

Record `UPDATE.2`

This record includes a set of real variables (computational parameters) that are to be updated. The data in this record are read using a `NAMELIST` format, and may occupy one or more lines (a choice left to the user). As already discussed (Section 8.1), `NAMELIST`-based formats are a feature of FORTRAN 95/2003 and provide unique power and flexibility, allowing (a) assignment of updated values to any subset of the parameters included in the `NAMELIST` definition, (b) arbitrary order, (c) free formats of individual parameter values, (d) inclusion of comments, etc.

Future versions of TOUGH+ will involve NAMELIST-based formats to read most input data.

The namelist in this record is named Real_Parameters_To_Update, and includes the following real parameters:

(1) SimulationTimeEnd
(2) MaxTimeStep
(3) rel_convergence_crit
(4) abs_convergence_crit
(5) P_overshoot
(6) T_overshoot
(7) S_overshoot

The corresponding TOUGH+ computational parameters will be updated if values are provided for any of these computational parameters. The structure of the Real_Parameters_To_Update namelist (and its use as an input format) is best illustrated in the example of **Figure 10.1**.

Record UPDATE.3

This record includes a set of integer computational parameters that are to be updated. The data in this record are read using a NAMELIST format. This namelist is named Integer_Parameters_To_Update, and includes the following integer parameters:

(1) Max_NumTimeSteps
(2) Max_NumNRIterations
(3) MOP_16
(4) SAVE_frequency
(5) TimeSeries_frequency

The corresponding TOUGH+ integer parameters will be updated if values are provided for any of these computational parameters. It is possible to stop a simulation by providing a Max_NumTimeSteps value that is smaller than its current value in the code. Then, the simulation will be halted upon reading the smaller Max_NumTimeSteps value while at the same time preserving the data in the **SAVE** file, which would be lost if the execution is interrupted.

The structure of the Integer_Parameters_To_Update namelist (and its use as an input format) is best illustrated in the example of **Figure 10.1**.

Upon reading the contents of the **Parameter_Update_File** and updating the computational parameters, TOUGH+ v1.5 (a) prints a prominent message in the standard output file that provides all the new parameter values, and (b) replaces the

`UpdateHeader = 'Update_Simulation_Parameters'` value with the value `UpdateHeader = '==> Completed Update # n'`, where `n` is the number of the update. Because `UpdateHeader` has no longer the value that will cause TOUGH+ to read the subsequent data, this substitution prevents multiple readings of the same **`Parameter_Update_File`** while keeping track of the number of updates and preserving the evolution of the updated parameters in the input file. Note that several updates are possible in the course of a long simulation. To accomplish this, the data for the next update are simply added to the top of the **`Parameter_Update_File`** file without erasing the updating history up to this point. The process is clearly illustrated in the example of **Figure 10.2**.

```
'Update_Simulation_Parameters'      ! UpdateHeader
    &Real_Parameters_To_Update    ! Namelist #1
        SimulationTimeEnd    = 5.0d6  ,
        MaxTimeStep          = 3600.  ,
        rel_convergence_crit = 1.5d-5 ,
        P_overshoot          = 1.0e-4 ,
        S_overshoot          = 1.0e-5
        /                      ! Not updated: abs_convergence_crit, T_overshoot
    &Integer_Parameters_To_Update   ! Namelist #2
        Max_NumTimeSteps    =  500   ,
        Max_NumNRIterations =  10000 ,
        MOP_16              =  4     ,
        SAVE_frequency      =  100   ,
        TimeSeries_frequency = 5
        /
```

Figure 10.1. An example of a **`Parameter_Update_File`** for parameter updating in the course of a TOUGH+ v1.5 simulation. Within the namelists (**`Real_Parameters_To_Update`** and **`Integer_Parameters_To_Update`**), parameters can be entered in any order, data are read using any kind of appropriate format, only the needed parameters are included, and comments can be added.

```
Update_Simulation_Parameters
    &Real_Parameters_To_Update
        SimulationTimeEnd    = 3.0d7
        /
    &Integer_Parameters_To_Update
        Max_NumTimeSteps    =  2000
        /
==> Completed Update #  3             ! 3rd Update
    &Real_Parameters_To_Update
        SimulationTimeEnd    = 2.0d7  ,
        MaxTimeStep          = 8.64e4 ,
        rel_convergence_crit = 2.5d-5 ,
        /
    &Integer_Parameters_To_Update
        Max_NumTimeSteps    =  1500  ,
        Max_NumNRIterations =  20000
        /
==> Completed Update #  2             ! 2nd Update
    &Real_Parameters_To_Update
        SimulationTimeEnd    = 8.0d6  ,
        MaxTimeStep          = 7200.  ,
        rel_convergence_crit = 2.0d-5 ,
        P_overshoot          = 1.0e-5
        /
    &Integer_Parameters_To_Update
        Max_NumTimeSteps    =  1000  ,
        MOP_16              =  5     ,
        SAVE_frequency      =  200   ,
        TimeSeries_frequency = 5
        /
==> Completed Update #  1             ! 1st Update
    &Real_Parameters_To_Update
        SimulationTimeEnd    = 5.0d6  ,
        MaxTimeStep          = 3600.  ,
        rel_convergence_crit = 1.5d-5 ,
        P_overshoot          = 1.0e-4 ,
        S_overshoot          = 1.0e-5
        /
    &Integer_Parameters_To_Update
        Max_NumTimeSteps    =  500   ,
        Max_NumNRIterations =  10000 ,
        MOP_16              =  4     ,
        SAVE_frequency      =  100   ,
        TimeSeries_frequency = 5
        /
```

Figure 10.2. An example of a **`Parameter_Update_File`** indicating three completed parameter updates, in addition to another one (at the top of the file) that has not yet been executed.

10.3. Data Block SOLVR

This (optional) block specifies parameters used by linear equation solvers.

Record SOLVR.1

Format(I1,2X,A2,3X,A2,2E10.4)

```
MatrixSolver, Z_preprocessing,
O_preprocessing, Max_CGIterationRatio,
CG_convergence_crit
```

These parameters are defined as follows:

MatrixSolver:
This integer variable selects the linear equation solver from among the following options:
= 1: LUBAND
= 2: DSLUBC
= 3: DSLUCS
= 4: DSLUGM
= 5: DLUSTB

Z_preprocessing:
A character variable that determines the type of Z-preconditioning [*Moridis and Pruess*, 1998]. Regardless of user specifications, Z-preprocessing will only be performed when iterative solvers are used ($2 \leq$ MatrixSolver ≤ 5), and if there are zeros on the main diagonal of the Jacobian matrix. The following options are available:

='Z0': No Z-preprocessing; default for NumEqu=1 and MatrixSolver=1

='Z1': Replace zeros on the main diagonal by a small near-zero constant (1.0E-25; default for NumEqu>1 and for 1<MatrixSolver≤5

='Z2': Make linear combinations of equations for each grid block to produce non-zero main diagonal entries

='Z3': Normalize equations, followed by Z2

='Z4': Same as in O_preprocessing='O4'

O_preprocessing:
A character variable that determines the type of O-preconditioning [*Moridis and Pruess*, 1998]. It can take the following possible values:

='O0': No O-preprocessing; default for NumEqu=1 and MatrixSolver=1

='O1': Elimination of lower half of the main-diagonal submatrix with center pivoting

='O2': O1 + Elimination of upper half of the main-diagonal submatrix with center pivoting

='O3': O2 + Normalization; results in unit main-diagonal submatrices

='O4': Pre-processing which results in unit main-diagonal submatrices without center pivoting

Max_CGIterationRatio:
An integer variable that specifies the maximum number of CG iterations as a fraction of the total number of equations
(0 < Max_CGIterationRatio ≤ 1; default is Max_CGIterationRatio = 0.1)

CG_convergence_crit:
A real variable that specifies the convergence criterion for the CG iterations
(1.0E-12≤CG_convergence_crit≤1.0E-6; the default CG_convergence_crit=1.0E-6)

The solver DLUSTB implements the BiCGSTAB(m) algorithm [*Sleijpen and Fokkema*, 1993], an extension of the BiCGSTAB algorithm of *van der Vorst* (1992). DLUSTB provides improved convergence behavior when iterations are started close to the solution, i.e., near steady state. The preconditoning algorithms can cope with difficult problems in which many of the Jacobian matrix elements on the main diagonal are zero. An example was given in *Pruess et al.* [1999; 2012] in "*two-waters*" problems in which typically 2/3 of the elements in the main diagonal are zero. Our tests show that this type of problem can be solved by means of Z2 or Z3 preconditioning [*Moridis and Pruess*, 1998].

10.4. Discussion on Linear Equation Solvers

The computational work to be performed in the course of a TOUGH+ v1.5 simulation includes evaluation of thermophysical properties for all grid blocks, assembly of the vector of residuals and the Jacobian matrix, and solution of the linear equation system for each Newton-Raphson iteration step. Except for small problems with just a few grid blocks, the most computationally intensive of these different tasks is the solution of the linear equation system. TOUGH+ v1.5 offers a choice of direct or iterative methods for linear equation solution; technical details of the methods and their performance can be found in *Moridis and Pruess* [1998].

The most reliable linear equation solvers are based on direct methods, while the performance of iterative techniques tends to be problem-specific and lacks the predictability of direct solvers. The robustness of direct solvers comes at the expense of large storage requirements and execution times that typically increase with problem size N (= number of equations solved) proportional to N^3. In contrast, iterative solvers have much lower memory requirements, and their computational work will increase much less rapidly with problem size, approximately proportional to N^{ω}, with $\omega \approx 1.4$ - 1.6 [*Moridis and Pruess*, 1995]. For large problems and especially 3-D problems with several thousand grid blocks or more, iterative conjugate gradient (CG) type solvers are therefore the method of choice.

The default linear equation solution technique in TOUGH+ uses DSLUCS, a Lanczos-type bi-conjugate gradient solver, with incomplete LU-factorization as preconditioner. Users need to beware that iterative methods can be quite “fickle” and may fail for matrices with special features, such as many zeros on the main diagonal, large

numerical range of matrix elements, and nearly linearly dependent rows or columns. Depending on features of the problem at hand, appropriate matrix preconditioning may be essential to achieve convergence. Poor accuracy of the linear equation solution will lead to deteriorated convergence rates for the Newtonian iteration, and an increase in the number of iterations for a given time step. In severe cases, time steps may fail with residuals either stagnating or wildly fluctuating. Information on the convergence of the linear equation solution is written to disk file **LINEQ**, which may be examined with any text editor. Users experiencing difficulties with the default settings are encouraged to experiment with the various solvers and preconditioners included in the TOUGH+ package.

PAGE LEFT INTENTIONALLY BLANK

11. Output Specifications

In this section, the various primary and secondary variables that may be provided as outputs from TOUGH+ v1.5 are discussed. In addition, data blocks are described for specifying output data, such as the times at which data are printed (data block **TIMES**), and for defining subdomains, interfaces and groups of sinks and sources (optional data blocks **SUBDOMAINS**, **INTERFACES** and **SS_GROUPS**, respectively) where important variables are to be monitored and tracked by printing time-series data in external output files. With the exception of data block **TIMES** that was available in the TOUGH2 family of codes [*Pruess et al*., 1991; 2012], these are new TOUGH+ v1.5 output capabilities unavailable to any earlier version of the code. This section also includes a few comments on error messages and warnings in the TOUGH+ v1.5 family of codes.

11.1 Output of Primary and Secondary Variables

The TOUGH+ v1.5 code can provide the following output:

1. The pressure, temperatures, saturations, and equilibrium pressure distributions.
2. The mass fractions of the various components in the various phases.
3. Flows and velocities of the phases across the gridblock interfaces (connections) of the domain.
4. The primary variables and their changes in the elements of the domain.
5. Capillary pressures and relative permeabilities.
6. Densities, viscosities (when mobile), and enthalpies of the various phases.
7. Dissociation reaction rates and the corresponding heat of dissociation.
8. Volume and mass balances of the phases and components in the domain.
9. Production rates and production composition at wells.
10. Time series of the evolution of the most important variables at user-specified elements, connections and sources/sinks.

Of those possible outputs, (1), (8), (9), and (10) are always printed in the standard TOUGH+ v1.5 output. The amount of the output is controlled by the parameter `OutputOption` in the data block **`PARAM`**. In keeping with the TOUGH2 convention, if the `OutputOption` values are increased by 10, printouts will occur after each iteration (not just after convergence). The specific outputs are discussed in detail in the User's Manuals of the various TOUGH+ application options.

11.2. Data Block `TIMES`

This optional block permits the user to obtain printout at specified times. This printout will occur in addition to printout specified in record **`PARAM.1`**.

Record `TIMES.1`

Format (2I5, 2E10.4)

```
NumPrintTimes, Max_NumPrintTimes,
TimeStepMax_PostPrint, PrintTimeIncrement
```

These parameters are defined as follows:

`NumPrintTimes`:
The number of times provided on records `TIMES.2`, `TIMES.3`, etc., (see below; restriction: `NumPrintTimes` ≤ `100`).

`Max_NumPrintTimes`:
The total number of times at which an output is desired (`NumPrintTimes` ≤ `Max_NumPrintTimes` ≤ `100`; default is `NumPrintTimes = Max_NumPrintTimes`).

`TimeStepMax_PostPrint`:
The maximum time step size after any of the prescribed times have been reached (default is infinite).

`PrintTimeIncrement` :
Time increment for times with indices `NumPrintTimes`, `NumPrintTimes+1`, ..., `Max_NumPrintTimes`.

Record `TIMES.2, TIMES.3`, etc.
Format (8E10.4)
`(PrintTime(i), i = 1, NumPrintTimes)`

`PrintTime(i)`:
A list of times (in ascending order) at which printout is desired. Note that, if any `PrintTime(i+1)< PrintTime(i)`, the resulting negative timestep forces the simulation to stop.

11.3. Data Block `SUBDOMAINS`

This optional data block is a new TOUGH+ v1.5 feature that is unavailable to any earlier version of the code. It allows monitoring of the evolution of the pore volume-averaged and properties and conditions in a set of subdomains (subsets) of the global grid (domain). The time series results are written in a number of separate output files that are equal to the number of the subdomains (a file for each subdomain), and are named according to the convention *`SubdomName`***`_Time_Series`**, where '*`SubdomName`*' is the name of the

subdomain under observation. If the keyword 'SUBDOMAINS' is present in the input file, then the following data are read using NAMELIST formats:

Record SUBDOMAIN.1

This record includes general data describing the number of subdomains to be monitored. The namelist in this record is named Subdomain_General_Info and has the general form:

```
&Subdomain_General_Info  number_of_subdomains = x /,
```

i.e., it contains only the following single parameter:

number_of_subdomains:
: An integer parameter describing the number of subdomains on which time series data are to be obtained. There is no upper limit in the number of subdomains that can be defined.

  NOTE: *The* number_of_subdomains *parameter must be set to a positive value for the simulation to continue.*

The structure of the Subdomain_General_Info namelist and its use as the input format in the data block **SUBDOMAINS** is best illustrated in the examples of **Figure 11.1**.

Record SUBDOMAIN.2

The namelist in this record provides general information on each of the subdomains. A total of number_of_subdomains such records/namelists must be provided. This namelist is named Individual_Subdomain_Specifics, has the general form

```
&Individual_Subdomain_Specifics subdomain_name    = x,
                                number_of_regions = x /
```

and includes the following parameters:

subdomain_name:
: The subdomain name is described by this character variable of a maximum of 8 characters. Upon reading the variable, the TOUGH+ code creates an output file named ***SubdomName*_Time_Series**, where '*SubdomName*' is the value of subdomain_name. If this variable is omitted in the input, then the TOUGH+ code assigns internally the name *SubdomName* = 'SubDomNN', where 'NN' is the 2-digit subdomain number in the subdomain definition sequence.

`number_of_regions:`

An integer parameter specifying the number of regions of which the subdomain is composed. **NOTE**: *The* `number_of_regions` *parameter must be set to a positive value for the simulation to continue.*

The structure of the `Individual_Subdomain_Specifics` namelist and its use as an input format in the data block **`SUBDOMAINS`** is best illustrated in the examples of **Figure 11.1**.

Record `SUBDOMAIN.2.1`

The namelist in this record is named `Region_Specifics`. It provides information that allows the definition of a region within a subdomain and identification of the elements belonging to it. A total of `number_of_regions` such records must by provided in order to fully define the subdomain. In its most complete form it has the following structure:

```
&Region_Specifics  definition_mode         = 'x',
                   number_of_elements      =  x,
                   first_element_number    =  x,
                   first_element_name      = 'xxxxx',
                   element_sequence_stride =  x,
                   format_to_read_data     = 'x',
                   region_shape            = 'x',
                   Xmin                    =  x.xEx,
                   Xmax                    =  x.xEx,
                   Ymin                    =  x.xEx,
                   Ymax                    =  x.xEx,
                   Zmin                    =  x.xEx,
                   Zmax                    =  x.xEx,
                   Rmin                    =  x.xEx,
                   Rmax                    =  x.xEx,
                   top_cylinder_center_xyz =  x.xEx,
                   bot_cylinder_center_xyz =  x.xEx,
                   sphere_center_xyz       =  x.xEx
                   /
```

The great advantage of the use of namelists is that only their necessary components may be used and be assigned values; the rest can be omitted or commented-out. Additionally, the namelist components can be listed in any order. The parameters in the `Region_Specifics` namelist are defined as follows:

`definition_mode:`

This character variable (up to 10 characters in length) describes the method by which the region is defined. It can take the following possible values:

`='Geometry'`: The region is defined by its geometric boundaries

`='Sequence'`: The region is defined by a sequence of element numbers

='NumberList': The region is defined by a list of element numbers

='NameList': The region is defined by a list of element names

number_of_elements:
This integer parameter is needed as an input when definition_mode ≠'Geometry' and describes the number of elements in the region.

format_to_read_data:
This character variable (up to 50 characters long) is needed as an input when definition_mode ='NumberList' or 'NameList', and describes the format to read the list of number_of_elements element numbers or names. The number_of_elements elements to be read according to the format format_to_read_data are listed immediately after the end of the namelist (see examples of **Figure 11.1)**.

NOTE: *no other parameters of those in the namelist are needed if* definition_mode ='NumberList' or 'NumberList'.

first_element_number, first_element_name, element_sequence_stride:
These integer parameters are needed as inputs when definition_mode ='Sequence'. Using the provided first_element_number or determining it from the first_element_name, a total of number_of_elements element numbers are determining using the element_sequence_stride.

NOTE: *no other parameters of those in the namelist are needed if* definition_mode ='Sequence'.

region_shape:
This character variable (up to 9 characters in length) is needed as an input when definition_mode ='Geometry' and describes the shape of the region that is about to be defined. The following self-explanatory options are available:

='Rectangle': This option can only be used in Cartesian grids, i.e., if coordinate_system = 'Cartesian' (see Section 5.1).

='Cylinder': This option can only be used for either Cartesian or cylindrical grids.

='Sphere': This option can only be used for either Cartesian or cylindrical grids.

If `region_shape = 'Rectangle'` and `coordinate_system = 'Cartesian'`, then the following parameters must be included in the namelist:

```
Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
```

These real parameters indicate the range (minimum and maximum) of the region along the *x*-, *y*- and *z*-axis of the Cartesian coordinate system, respectively.

If `region_shape = 'Cylinder'` and `coordinate_system = 'Cartesian'`, then the following parameters must be included in the namelist:

```
Rmin, Rmax, top_cylinder_center_xyz,
      bot_cylinder_center_xyz
```

The real parameters `Rmin` and `Rmax` are the minimum and maximum radii of the cylindrical region. The 1D arrays `top_cylinder_center_xyz` and `bot_cylinder_center_xyz` are of size 3. These are the (*x,y,z*) coordinates of the centers of the top and bottom circular surfaces of the cylinder, respectively.

If `region_shape = 'Cylinder'` and `coordinate_system = 'Cylindrical'`, then the following parameters must be included in the namelist:

```
Rmin, Rmax, Zmin, Zmax
```

These real parameters indicate the range (minimum and maximum) of the region along the *r*- and *z*-axis of the Cartesiancylindrical coordinate system, respectively.

If `region_shape = 'Sphere'` and `coordinate_system = 'Cartesian'` or `coordinate_system = 'Cylindrical'`, then the following parameters must be included in the namelist:

```
Rmin, Rmax, sphere_center_xyz
```

The real parameters `Rmin` and `Rmax` are the minimum and maximum radius of the sphere. The (*x,y,z*) coordinates of the center of the sphere are stored in the elements of the real array `top_cylinder_center_xyz` of size 3. Of those three values, only the 3rd (corresponding to the *z*-coordinate of the center of the sphere) one is used when `coordinate_system = 'Cylindrical'`.

NOTE: *no other parameters of those in the namelist are needed if* `definition_mode ='Geometry'`.

The use of these inputs in the **SUBDOMAINS** datablock is best illustrated in the examples of **Figure 11.1**.

NOTE #1: It is possible to combine regions of very different geometries to create subdomains that are very irregular in shape. Each one of these regions can be defined independently, i.e., by using different `definition_mode` values.

NOTE #2: *All the various parameter combinations and permutations in the user-supplied namelist* `Region_Specifics` *must be defined for the simulation to continue.*

NOTE #3: *All the regions within the subdomain must be defined. A total of* `number_of_regions` *records (one for each region of the subdomain) must be provided in* `SUBDOMAIN.2.1`. *Thus, if we define two subdomains, the first with 3 regions and the second with 4, then we need to provide the following data: For the 1st Subdomain: One* `SUBDOMAIN.2` *record, and 3* `SUBDOMAIN.2.1` *records. For the 2nd Subdomain: One* `SUBDOMAIN.2` *record, and 4* `SUBDOMAIN.2.1` *records.*

11.4. Data Block INTERFACES

This optional data block is a new TOUGH+ v1.5 feature that is unavailable to any earlier version of the code. It allows monitoring the flow – instantaneous and cumulative – through user-defined interfaces (composed of smalled individual surfaces). The time series results are written in a number of separate output files that are equal to the number of the interfaces (a file for each interface), and are named according to the convention *InterfName*`_Time_Series`, where '*InterfName*' is the name of the interface under observation. The *InterfName*`_Time_Series` files are important in continuatin runs: information from the last TOUGH+ simulation is gleaned from the older files and is used to seamlessly continue the computations of the cumulative mass flows through the interfaces.

If the keyword `'INTERFACES'` is present in the input file, then the following data are read using `NAMELIST` formats:

```
SUBDOMAINS
&Subdomain_General_Info  number_of_subdomains = 1 /
   &Individual_Subdomain_Specifics  subdomain_name    = 'Zone1',
                                    number_of_regions = 1 /
      &Region_Specifics  definition_mode = 'Geometry',
                         region_shape    = 'Rectangle',
                         Xmin =  2.0e-1, Xmax = 4.0e-1,
                         Ymin = -1.0e8,  Ymax = 1.0e8,
                         Zmin = -1.0e8,  Zmax = 1.0e8,
                       ! Rmin, Rmax,                 ! Not used – commented out
                       ! top_cylinder_center_xyz,    ! Not used – commented out
                       ! bot_cylinder_center_xyz,    ! Not used – commented out
                         /


SUBDOMAINS
&Subdomain_General_Info  number_of_subdomains = 1 /
   &Individual_Subdomain_Specifics  subdomain_name = 'Zone8',
                                    number_of_regions = 2
                                    /
      &Region_Specifics  definition_mode = 'Sequence',
                         number_of_elements      = 5,
                         first_element_number    = 8,
                         element_sequence_stride = 1
                         /
      &Region_Specifics  definition_mode = 'Sequence',
                         number_of_elements      =  5,
                         first_element_name      = 'A0010',
                         element_sequence_stride = 1
                     /



&Subdomain_General_Info  number_of_subdomains = 1 /
   &Individual_Subdomain_Specifics  subdomain_name = 'Zone5',
                                    number_of_regions = 3 /
      &Region_Specifics  definition_mode = 'NumberList',
                         number_of_elements = 5,
                         format_to_read_data = '*',
                         /
8 9 10 11 12
      &Region_Specifics  definition_mode = 'NameList',
                         number_of_elements = 5,
                         format_to_read_data = '*',
                         /
'A0012'  'A0013'  'A0014'  'A0015'  'A0016'
        &Region_Specifics  definition_mode = 'Geometry',
                         region_shape    = 'Cylinder'
                         Rmin =  2.0e-1, Rmax = 1.0e00,
                       !  Ymin = -1.0e8,  Ymax = 1.0e8,    ! Not used – commented out
                         Zmin = -1.0e8,  Zmax = 1.0e8,
                       !  top_cylinder_center_xyz,          ! Not used – commented out
                       !  bot_cylinder_center_xyz           ! Not used – commented out
                        /
```

Figure 11.1. Examples of the **SUBDOMAINS** data block for tracking the evolution of volume-averaged properties and conditions in specified subdomains. This data block uses namelist-based formats for data inputs.

Record INTERFACE.1

This record includes general data describing the number of interfaces to be monitored. The namelist in this record is named Interface_General_Info and has the general form:

```
&Interface_General_Info  number_of_interfaces = x /,
```

i.e., it contains only the following single parameter:

number_of_interfaces:
: An integer parameter describing the number of interfaces at which time series data are to be obtained. There is no upper limit in the number of interfaces that can be defined.

  NOTE: *The* number_of_interfaces *parameter must be set to a positive value for the simulation to continue.*

The structure of the Interface_General_Info namelist and its use as the input format in the data block **INTERFACES** is best illustrated in the examples of **Figure 11.2**.

Record INTERFACE.2

The namelist in this record provides general information on each of the interfaces. A total of number_of_interfaces such records/namelists must be provided. This namelist is named Individual_Interface_Specifics, has the general form

```
&Individual_Interface_Specifics interface_name         ='x',
                                number_of_surfaces     = x,
                                sign_of_flow_direction ='x'/
```

and includes the following parameters:

interface_name:
: The interface name is described by this character variable of a maximum of 8 characters. Upon reading the variable, the TOUGH+ v1.5 code creates an output file named ***InterfName*_Time_Series**, in which '*InterfName*' is the value of interface_name. If this variable is omitted in the input, then the TOUGH+ code assigns internally the name *InterfName* = 'IntrFcNN', where 'NN' is the 2-digit interface number in the interface definition sequence.

`number_of_surfaces:`

An integer parameter specifying the number of individual surfaces of which the interface is composed. **NOTE**: *The* `number_of_surfaces` *parameter must be set to a positive value for the simulation to continue.*

`sign_of_flow_direction:`

A character parameter of length 3 that defines how the sign of the flow (positive or negative) is to be treated in reporting the information. It can take the following possible values:

`='ABS'`: The absolute value of the flows at the surface connections is used. This is necessary in cases where the arbitrariness in the order of the elements in a connection (a TOUGH+ v1.5 feature) may produce erroneous results if the arithmetic sum of the flows through individual connections is used. For example, in computing flow toward a well in a 3D Cartesian grid, the direction of flow is well known but the signs of the individual flows through each connection in the surface may differ because of the element orientation. In such a case, the `'ABS'` value is known to accurately describe flows.

`='DIR'`: The internal signs of the individual flows at the surface connections are maintained in the flow summations. This option is used when the order of elements in the definition of the connections is known to be monotonic (in terms of direction), i.e., when the grid numbering is predictably consistent and the flow through a surface is known to be variable in direction (e.g., the net recharge of an aquifer from an aquitard with an operating injection well – water escapes into int overlying aquifer near the injection well, but recharge from the aquitard occurs away from the well).

The structure of the `Individual_Interface_Specifics` namelist and its use as an input format in the data block **`INTERFACES`** is best illustrated in the examples of **Figure 11.2**.

Record `INTERFACE.2.1`

The namelist in this record is named `Surface_Specifics`. It provides information that allows the definition of a surface within an interface and the identification of the connections belonging to it. A total of `number_of_surfaces` such records must by provided in order to fully define the interface. In its most complete form it has the following structure:

```
&Surface_Specifics   definition_mode         = 'x',
                     number_of_connections   =  x,
                     format_to_read_data     = 'x',
                     surface_shape           = 'x',
                     Xmin                    =  x.xEx,
                     Xmax                    =  x.xEx,
                     Ymin                    =  x.xEx,
                     Ymax                    =  x.xEx,
                     Zmin                    =  x.xEx,
                     Zmax                    =  x.xEx,
                     surface_location        =  x.xEx,
                     surface_coord_number    =  x.xEx,
                     cylinder_radius         =  x.xEx,
                     top_cylinder_center_xyz =  x.xEx,
                     bot_cylinder_center_xyz =  x.xEx,
                     inner_circle_radius     =  x.xEx
                     outer_circle_radius     =  x.xEx
                     sphere_center_xyz       =  x.xEx
                     sphere_radius           =  x.xEx
                     /
```

The great advantage of the use of namelists is that only their necessary components may be used and assigned values; the rest can be completely omitted or commented-out. Additionally, the components of the namelist can be listed in any order. The parameters in the `Surface_Specifics` namelist are defined as follows:

`definition_mode:`

This character variable (up to 10 characters in length) describes the method by which the region is defined. It can take the following possible values:

`='Geometry'`: The surface is defined by its geometric attributes

`='NumberList'`: The region is defined by a list of connection numbers

`='NameList'`: The region is defined by a list of connection names

`number_of_connections:`

This integer parameter is needed as an input when `definition_mode` `≠'Geometry'` and describes the number of connections in the surface.

`format_to_read_data:`

This character variable (up to 50 characters long) is needed as an input when `definition_mode ='NumberList'` or `'NameList'`, and describes the format to read the list of `number_of_connections` connection numbers or names. The `number_of_connections` elements to be read according to the format `format_to_read_data` are listed immediately after the end of the namelist (see examples of **Figure 11.2)**.

NOTE: *No other parameters of those in the namelist are needed if* `definition_mode ='NumberList'` or `'NumberList'`.

`surface_shape:`

This character variable (up to 9 characters in length) is needed as an input when `definition_mode ='Geometry'` and describes the shape of the surface that is about to be defined. The following self-explanatory options are available:

`='Rectangle'`: This option can only be used in Cartesian grids, i.e., if `coordinate_system = 'Cartesian'`(see Section 5.1).

`='Cylinder'`: This option can only be used for either Cartesian or cylindrical grids.

`='Circle'`: This option can only be used for either Cartesian or cylindrical grids. It is used to compute flows through a circular surface, e.g., at the top a bottom permeable boundaries of an aquifer with a well at its center.

`='Sphere'`: This option can only be used for either Cartesian or cylindrical grids.

If `surface_shape = 'Rectangle'` and `'coordinate_system = 'Cartesian'`, then the following parameters *must* be included in the namelist:

```
Xmin, Xmax, Ymin, Ymax, Zmin, Zmax
```

The first four real parameters in the list indicate the range (minimum and maximum) of the surface along the *x*-, *y*- and *z*-axis of the Cartesian coordinate system, respectively. Obviously, the max and min values along one (only) direction have to be identical, thus defining the orientation of the planar surface. For example, a planar surface perpendicular to the *z*-axis is defined when `Zmin = -20m` and `Zmax = -20m`. If the max and min coordinate values match in more than one direction, or if there is no match in any direction, an error message is printed and the simulation is aborted because the orientation of the planar surface cannot be determined. *No other parameters of those in the namelist are needed to define the planar surface.*

If `surface_shape = 'Cylinder'` and `coordinate_system = 'Cartesian'` or `coordinate_system = 'Cylindrical'`, then the following parameters *must* be included in the namelist:

```
top_cylinder_center_xyz,
bot_cylinder_center_xyz,
cylinder_radius
```

The function of the `cylinder_radius` real parameter is obvious. The real 1D arrays `top_cylinder_center_xyz` and `bot_cylinder_center_xyz` are of size 3. These are the (*x,y,z*) coordinates of the centers of the top and bottom circular surfaces of the cylinder, respectively. When `coordinate_system = 'Cylindrical'`, only the *z*-coordinates in the two arrays are important because the axis of the cylindrical surface is automatically aligned with the *z*-axis. *No other parameters of those in the namelist are needed to define the cylindrical surface.*

If `surface_shape = 'Circle'` and `coordinate_system = 'Cylindrical'` or `coordinate_system = 'Cylindrical'`, then flow is perpendicular to the circular surface and the following parameters must be included in the namelist:

```
inner_circle_radius, outer_circle_radius,
surface_location, surface_coord_number
```

The function of first two real parameters is self-explanatort. The integer parameter `surface_coord_number` identifies is the direction of flow through the surface (i.e., it is perpendicular to the circular surface). Thus, for flow along the *x*-direction, `surface_coord_number = 1`; for flow along the *z*-direction, `surface_coord_number = 3`. The real parameter `surface_location` identifies the location (coordinate) of the circular surface on the axis specified by `surface_coord_number`. Thus, if `surface_coord_number = 3` and `surface_location=-40 m`, then the circular surface is perpendicular to the z-axis at this elevation. When `coordinate_system = 'Cylindrical'`, only the `surface_location` is important (i.e., the `surface_coord_number` parameter may be omitted from the namelist) because the circular surface is taken to be perpendicular to the *z*-axis. *No other parameters of those in the namelist are needed to define the circular surface.*

If `surface_shape = 'Sphere'` and `coordinate_system = 'Cylindrical'` or `coordinate_system = 'Cylindrical'`, then the following parameters must be included in the namelist:

```
sphere_center_xyz, sphere_radius
```

The (*x,y,z*) coordinates of the center of the sphere are stored in the elements of the real, 1D array `top_cylinder_center_xyz` of size 3.

Of those three values, only the 3rd (corresponding to the *z*-coordinate of the center of the sphere) one is used when `coordinate_system = 'Cylindrical'` because in this case the center of the spherical surface is assumed to be at a radius $r = 0$. *No other parameters of those in the namelist are needed to define the spherical surface.*

`format_to_read_data:`

This character parameter (up to 50 characters long) is needed as an input when `definition_mode ='NumberList'` or `'NumberList'`, and describes the format to read the list of `number_of_connections` connection numbers or names. The `number_of_connections` elements to be read according to the format `format_to_read_data` are listed immediately after the end of the namelist (see examples in **Figure 11.2)**.

NOTE: *no other parameters of those in the namelist are needed if* `definition_mode ='NumberList'` or `'NumberList'`.

The use of these inputs in the **`INTERFACES`** namelist is best illustrated in the examples of **Figure 11.2**.

NOTE #1: It is possible to combine surfaces of very different geometries to create interfaces that are very irregular in shape. Each one of these surfaces can be defined independently, i.e., using different `definition_mode` values for each.

NOTE #2: *All the various parameter combinations and permutations in the user-supplied namelist* `Interface_Specifics` *must be defined for the simulation to continue.*

NOTE #3: *All the surfaces within the interface must be defined. A total of* `number_of_surfaces` *records (one for each surface of the interface) must be provided in* `INTERFACE.2.1`. *Thus, if we define two interfaces, the first with 3 surfaces and the second with 4, then we need to provide the following data: For the 1st Interface: One* `INTERFACE.2` *record, and 3* `SUBDOMAIN.2.1` *records. For the 2nd Interface: One* `INTERFACE.2` *record, and 4* `INTERFACE.2.1` *records.*

The structure and use of the namelists in the data block **`INTERFACES`** is best illustrated by the examples of **Figure 11.2**.

```
INTERFACES

&Interface_General_Info  number_of_interfaces = 1 /
   &Individual_Interface_Specifics  interface_name         = 'Int03',
                                    number_of_surfaces     = 1,
                                    sign_of_flow_direction = 'DIR'
                                    /
      &Surface_Specifics  definition_mode       = 'NameList',
                          number_of_connections = 5,
                          format_to_read_data = '*',
                          /
'A00 4A00 5'
'A00 3A00 4'
'A00 5A00 6'
'A00 6A00 7'
'A00 7A00 8'


INTERFACES
&Interface_General_Info  number_of_interfaces = 2 /
   &Individual_Interface_Specifics  interface_name         = 'LBase',
                                    number_of_surfaces     = 1,
                                    sign_of_flow_direction = 'DIR' /
      &Surface_Specifics  definition_mode      = 'Geometry',
                          surface_shape        = 'Rectangle'
                          Xmin = -2.0e3,  Xmax = 5.0e03,
                          Ymin = -2.0e3,  Ymax = 5.0e03,
                          Zmin = -1.1d2,  Zmax = -1.1d2,
                          /
   &Individual_Interface_Specifics  interface_name         = 'MBase',
                                    number_of_surfaces     = 1,
                                    sign_of_flow_direction = 'DIR' /
      &Surface_Specifics  definition_mode      = 'Geometry',
                          surface_shape        = 'Rectangle'
                          Xmin = -2.0e3,  Xmax = 5.0e03,
                          Ymin = -2.0e3,  Ymax = 5.0e03,
                          surface_location     = -3.0d1
                          surface_coord_number = 3,
                          /


INTERFACES
&Interface_General_Info  number_of_interfaces = 1 /
   &Individual_Interface_Specifics  interface_name         = 'Int03',
                                    number_of_surfaces     = 1,
                                    sign_of_flow_direction = 'ABS'/
      &Surface_Specifics  definition_mode         = 'Geometry',
                          surface_shape           = 'Cylinder'
                          top_cylinder_center_xyz = 0.0d0, 0.0d0, -2.0d1,
                          bot_cylinder_center_xyz = 0.0d0, 0.0d0, -6.0d1,
                          cylinder_radius = 1.0d1,
                          /


INTERFACES
&Interface_General_Info  number_of_interfaces = 1 /
   &Individual_Interface_Specifics  interface_name         = 'Int04',
                                    number_of_surfaces     = 1,
                                    sign_of_flow_direction = 'ABS'/
      &Surface_Specifics  definition_mode         = 'Geometry',
                          surface_shape           = 'Sphere'
                          sphere_center_xyz = 0.0d0, 0.0d0, -5.0d1,
                          sphare_radius     = 2.0d1
                          /
```

Figure 11.2. Examples of the **INTERFACES** data block for tracking flows at specified interfaces. This data block uses namelist-based formats for data inputs.

11.5. Data Block SS_GROUPS

This optional data block is a new TOUGH+ v1.5 feature that is unavailable to any earlier version of the code. It allows monitoring the flow – instantaneous and cumulative – through user-defined groups of sources and sinks (hereafter referred to as SSG). The time series results are written in a number of separate output files that are equal to the number of the SSG's (one for each SSG), and are named according to the convention ***SSGName*_Time_Series**, where '*SSGName*' is the name of the SSG under observation. The ***SSGName*_Time_Series** files are important in continuatin runs: information from the last TOUGH+ simulation is gleaned from the older files and is used to seamlessly continue the computations of the cumulative mass flows through the SSG's.

If the keyword '**SS_GROUPS**' is present in the input file, then the following data are read using NAMELIST formats:

Record SS_GROUP.1

This record includes general data describing the number of interfaces to be monitored. The namelist in this record is named SSGroup_General_Info and has the general form:

```
&SSGroup_General_Info  number_of_SSGroups = x /,
```

i.e., it contains only the following single parameter:

number_of_SSGroups:
: An integer parameter describing the number of SSG's at which time series data are to be obtained. There is no upper limit in the number of SSG's that can be defined.

  NOTE: *The* number_of_SSGroups *parameter must be set to a positive value for the simulation to continue.*

The structure of the SSGroup_General_Info namelist and its use as the input format in the data block **SS_GROUPS** is best illustrated in the example of **Figure 11.3**.

Record SS_GROUP.2

The namelist in this record provides general information on each of the interfaces. A total of number_of_SSGroups such records/namelists must be provided. This namelist is named Individual_SSGroup_Specifics, has the general form

```
&Individual_SSGroup_Specifics   SSGroup_name        ='x',
                                definition_mode     = x,
                                number_of_SS        ='x',
                                format_to_read_data ='x',/
```

and includes the following parameters:

SSGroup_name:
The interface name is described by this character variable of a maximum of 9 characters. Upon reading the variable, the TOUGH+ v1.5 code creates an output file named ***SSGroupName*_Time_Series**, where '***SSGroupName***' is the value of interface_name. If this variable is omitted in the input, then the TOUGH+ code assigns internally the name ***SSGroupName*** = 'SSGroupNN', where 'NN' is the 2-digit interface number in the interface definition sequence.

definition_mode:
This character variable (up to 10 characters in length) describes the method by which the region is defined. It can take the following possible values:

='NumberList': The region is defined by a list of connection numbers

='NameList': The region is defined by a list of connection names

number_of_SS:
An integer parameter specifying the number of individual sources and sinks in the SS group. *The* number_of_SS *parameter must be set to a positive value for the simulation to continue.*

format_to_read_data:
This character variable (up to 50 characters long) is needed as an input when definition_mode ='NumberList' or 'NameList', and describes the format to read the list of number_of_SS numbers or names of the sources/sinks in the group. The number_of_SS elements to be read according to the format format_to_read_data are listed immediately after the end of the namelist (see example of **Figure 11.3)**.

The structure and use of the namelists in the data block **SS_GROUPS** is best illustrated by the example of **Figure 11.3**.

11.8. Warning Output and Error Messages

If inputs indicate conflicting conditions and/or parameter values are outside realistic ranges, TOUGH+ is designed to respond according to the severity of the violation. Non-critical conflicts result in a warning or clarifying message, internal adjustment of the corresponding conditions and/or parameters, and continuation of the simulation. Serious violations (e.g., initial conditions that violate fundamentals of physics and thermodynamics) result in an error message identifying the problem and the interruption of the simulation.

```
SS_GROUPS
&SSGroup_General_Info  number_of_SSGroups = 1 /
   &Individual_SSGroup_Specifics  SSGroup_name        = '     ',
                                  definition_mode     = 'NameList',
                                  number_of_SS        = 3,
                                  format_to_read_data = '*'
                                  /
'Pro01'
'Pro02'
'Wel06'
```

Figure 11.3. Example of the **SS_GROUPS** data block for tracking flows through specified groups of sources/sinks. This data block uses namelist-based formats for data inputs.

PAGE LEFT INTENTIONALLY BLANK

Acknowledgements

This work was supported by the Assistant Secretary for Fossil Energy, Office of Natural Gas and Petroleum Technology, through the National Energy Technology Laboratory, under the U.S. Department of Energy, Contract No DE-AC02-05CH11231.

PAGE LEFT INTENTIONALLY BLANK

References

Americal Petroleum Institute (API), Division of Refining, *Technical Data Book, Petroleum Refining*, American Petroleum Institute, Refining Department, Washington, 1977.

Barree R.D., and M.W. Conway, Multiphase non-Darcy flow in proppant packs, Paper SPE 109561, 2007 Annual Technical Conference and Exhibition, Anaheim, CA, 11–14 Nov 2007.

Bejan, A., *Convection Heat Transfer*, John Wiley & Sons, New York, 1984

Bird, R.B., W.E. Stewart, and E.N. Lightfoot, *Transport Phenomena*, New York: John Wiley & Sons, Inc., 2007.

Brooks, R.H. and A.T. Corey, Hydraulic Properties of Porous Medium, Hydrology Paper No. 3, Colorado State University, Fort Collins, Colorado, March 1964.

Cass, A., G.S. Campbell and T.L. Jones, Enhancement of Thermal Water Vapor Diffusion in Soil. *Soil Sci. Soc. Am. J.,* **48**(1), 25 - 32, 1984.

Chung, T.H., M. Ajlan, L.L. Lee and K.E. Starling, Generalized multiparameter correlation for nonpolar and polar fluid transport properties, *Ind. Eng. Chem. Res*., **27**(4), 671-679, 1988 (doi: 10.1021/ie00076a024)

Corey, A.T., The Interrelation Between Gas and Oil Relative Permeabilities, *Producers Monthly*, 38-41, November 1954.

de Marsily, G., *Quantitative Hydrogeology*, Academic Press, Orlando, FL, 1986.

Doughty, C., Investigation of conceptual and numerical approaches for evaluating moisture, gas, chemical, and heat transport in fractured unsaturated rock, *J. Contam.t Hydrol.*, **38**(1-3), 69-106, 1999

Edwards, A.L., TRUMP: A Computer Program for Transient and Steady State Temperature Distributions in Multidimensional Systems, National Technical Information Service, National Bureau of Standards, Springfield, VA, 1972.

Falta, R.W., K. Pruess, I. Javandel and P.A. Witherspoon, Density-Driven Flow of Gas in the Unsaturated Zone Due to the Evaporation of Volatile Organic Compounds, *Water. Resour. Res.*, **25**(10), 2159 - 2169, 1989.

Fatt, I. and W.A. Klikoff, Effect of Fractional Wettability on Multiphase Flow Through Porous Media, *AIME Transactions*, 216, 246, 1959.

Finsterle, S., iTOUGH2 User's Guide, Report LBNL-40040, Lawrence Berkeley National Laboratory, Berkeley, California, 1999.

Finsterle, S., Implementation of the Forchheimer Equation in iTOUGH2, Project Report, Lawrence Berkeley National Laboratory, Berkeley, Calif., 2001.

Forchheimer, P., Wasserbewewegung durch Boden, *Zeit.Ver. Dtsch. Ing.* **45**, 1781, 1901.

Freeman, C.M., G.J. Moridis, and T.A. Blasingame, A Numerical Study of Microscale Flow Behavior in Tight Gas and Shale Gas Reservoir Systems, *Transp. in Porous Med., 90*(1): 253-268, 2011 (doi: 10.1007/ s11242-011-9761-6)

Fuller, E. N., K. Ensley, and J. C. Giddings, Diffusion of Halogenated Hydrocarbons in Helium: The Effect of Structure on Collision Cross Sections, *J. Phys. Chem.,* **73,** 3679-3685, 1969.

Grant, M.A., Permeability Reduction Factors at Wairakei, paper 77-HT-52, presented at AICHE-ASME Heat Transfer Conference, Salt Lake City, Utah, August 1977.

International Association for the Properties of Water and Steam (IAPWS), *Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam*. Erlangen, Germany, 1997.

International Association for the Properties of Water and Steam (IAPWS). *Revised Release on the IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam*, Lucerne, Switzerland, 2007.

International Association for the Properties of Water and Steam (IAPWS). *Release on the IAPWS Formulation 2008 for the Viscosity of Ordinary Water Substance*. Berlin, Germany, 2008.

International Association for the Properties of Water and Steam (IAPWS). *Revised Release on the Equation of State 2006 for H2O Ice Ih*. Doorwerth, The Netherlands, 2009.

International Association for the Properties of Water and Steam (IAPWS). *Release on the IAPWS Formulation 2011 for the Thermal Conductivity of Ordinary Water Substance*. Plzeň, Czech Republic, 2011a.

International Association for the Properties of Water and Steam (IAPWS). *Revised Release on the Pressure along the Melting and Sublimation Curves of Ordinary Water Substance*. Plzeň, Czech Republic, 2011b.

International Association for the Properties of Water and Steam (IAPWS). *Guideline on a Low-Temperature Extension of the IAPWS-95 Formulation for Water Vapor*. Boulder, Colorado, USA, 2012.

International Formulation Committee, *A Formulation of the Thermodynamic Properties of Ordinary Water Substance*, IFC Secretariat, Düsseldorf, Germany, 1967.

Itasca Consulting Group. *FLAC3D: Fast Lagrangian Analysis of Continua in 3 Dimensions*, Minneapolis, Minnesota, 2002.

Jones, S. C., A rapid accurate unsteady-state Klinkenberg parameter, *SPE Journal* 383–397, 1972.

Jury, W.A., W.F. Spencer and W.J. Farmer, Behavior Assessment Model for Trace Organics in Soil: I. Model Description, *J. Environ. Qual.,* **12**(4), 558 - 564, 1983.

Kim, J., and G.J. Moridis, Development of the T+M coupled flow-geomechanical simulator to describe fracture propagation and coupled flow-thermal-geomechanical processes in tight/shale gas systems, *Computers & Geosciences, 60*, 184-198, 2013 (doi: 10.1016/j.cageo.2013.04.023)**.**

Klinkenberg, L.J., The Permeability of Porous Media to Liquids and Gases, in *API Drilling and Production Practice,* 200–213, 1941.

Lam, S.T., A. Hunsbedt, P. Kruger and K. Pruess, Analysis of the Stanford Geothermal Reservoir Model Experiments Using the LBL Reservoir Simulator, Geothermics, **17**(4), 595- 605, LBL-25957, 1988.

Leverett, M.C., Capillary Behavior in Porous Solids, *Trans. Soc. Pet. Eng. AIME*, **142**, 152-169, 1941.

Mason, E.A. and A.P. Malinauskas, *Gas Transport in Porous Media: The Dusty Gas Model,* Elsevier, Amsterdam, The Netherlands, 1983.

Millington, R.J. and J.P. Quirk, Permeability of Porous Solids, *Trans. Faraday Soc.*, **57**, 1200-1207, 1961.

Milly, P.C.D., Moisture and Heat Transport in Hysteretic, Inhomogeneous Porous Media: A Matric-Head Based Formulation and a Numerical Model, *Water Resour. Res.*, **18**(3), 489 - 498, 1982.

Moridis, G.J., User's Manual for the HYDRATE v1.5 option of TOUGH+ v1.5: A Code for the Simulation of System Behavior in Hydrate-Bearing Geologic Media, Lawrence Berkeley National Laboratory Report LBNL-6871E, August 2014.

Moridis, G.J. and C.M. Freeman, User's Manual for the REALGASBRINE v1.0 option of TOUGH+: A Code for the Simulation of System Behavior in Gas-Bearing Geologic Media, Lawrence Berkeley National Laboratory Report LBNL-6870E, August 2014.

Moridis, G. and K. Pruess, TOUGH Simulations of Updegraff's Set of Fluid and Heat Flow Problems, Lawrence Berkeley Laboratory Report LBL-32611, Berkeley, CA, November 1992.

Moridis, G. and K. Pruess, Flow and Transport Simulations Using T2CG1, a Package of Conjugate Gradient Solvers for the TOUGH2 Family of Codes, Lawrence Berkeley Laboratory Report LBL-36235, Berkeley, CA, 1995.

Moridis, G. and K. Pruess, T2SOLV: An Enhanced Package of Solvers for the TOUGH2 Family of Reservoir Simulation Codes, *Geothermics*, **27**(4), 415 - 444, 1998.

Moridis, G.J., M.B. Kowalsky and K. Pruess, TOUGH+HYDRATE v1.0 User's Manual: A Code for the Simulation of System Behavior in Hydrate-Bearing Geologic Media, Report LBNL-0149E, Lawrence Berkeley National Laboratory, Berkeley, CA (2008).

Moridis, G.J., M.B. Kowalsky and K. Pruess, TOUGH+HYDRATE v1.1 User's Manual: A Code for the Simulation of System Behavior in Hydrate-Bearing Geologic Media, Report LBNL-0149E, Lawrence Berkeley National Laboratory, Berkeley, CA (2009).

Moridis, G.J., M.B. Kowalsky and K. Pruess, TOUGH+HYDRATE v1.2 User's Manual: A Code for the Simulation of System Behavior in Hydrate-Bearing Geologic Media, Report LBNL-0149E, Lawrence Berkeley National Laboratory, Berkeley, CA (2012).

Morrow, C., D. Lockner, D. Moore and J. Byerlee, Permeability of Granite in a Temperature Gradient, *Journal of Geophysical Research*, **86**(84), 3002-3008, April 1981.

Mualem, Y., A New Model for Predicting the Hydraulic Conductivity of Unsaturated Porous Media, *Water Resour. Res.*, **12**(3), 513 - 522, 1976.

Narasimhan, T.N. and P.A. Witherspoon, An Integrated Finite Difference Method for Analyzing Fluid Flow in Porous Media, *Water Resour. Res.*, **12**(1), 57 – 64, 1976.

Narasimhan, T.N., P.A. Witherspoon and A.L. Edwards, Numerical Model for Saturated-Unsaturated Flow in Deformable Porous Media, Part 2: The Algorithm, *Water Resour. Res.*, **14**(2), 255-261, 1978.

Pape, H., C. Clauser and J. Iffland, Permeability Prediction Based on Fractal Pore-Space Geometry, *Geophysics,* **64**(5), 1447 - 1460, 1999.

Parker J.C., Lenhard R.J. and T. Kuppusamy, A Parametric Model for Constitutive Properties Governing Multiphase Flow in Porous Media, *Water Resour. Res.,* **23**(4), 618- 624, 1987.

Peaceman, D.W., *Fundamentals of Numerical Reservoir Simulation*, Elsevier, Amsterdam, The Netherlands, 1977.

Peng, D.Y., and D.B. Robinson, A New Two-Constant Equation of State, *Indust. and Engr. Chemistry: Fundamentals* **15**, 59-64, 1976.

Phillips, O.M., *Flow and Reactions in Permeable Rocks,* Cambridge University Press, Cambridge, New York, Melbourne, 1991.

Pickens, J.F., R.W. Gillham and D.R. Cameron, Finite Element Analysis of the Transport of Water and Solutes in Tile-Drained Soils, *J. of Hydrology*, **40**, 243-264, 1979.

Pruess, K., GMINC - A Mesh Generator for Flow Simulations in Fractured Reservoirs, Lawrence Berkeley Laboratory Report LBL-15227, Berkeley, CA, March 1983.

Pruess, K., TOUGH2 - A General Purpose Numerical Simulator for Multiphase Fluid and Heat Flow, Lawrence Berkeley Laboratory Report LBL-29400, Berkeley, CA, 1991.

Pruess, K., and T.N. Narasimhan, On Fluid Reserves and the Production of Superheated Steam from Fractured, Vapor-Dominated Geothermal Reservoirs, *J. Geophys. Res.*, **87**(B11), 9329 - 9339, 1982.

Pruess, K. and G.S. Bodvarsson, A Seven-Point Finite Difference Method for Improved Grid Orientation Performance in Pattern Steam Floods, *Proceedings*, Seventh Society of Petroleum Engineers Symposium on Reservoir Simulation, Paper SPE-12252, 175 - 184, San Francisco, CA, 1983.

Pruess, K. and T.N. Narasimhan, A Practical Method for Modeling Fluid and Heat Flow in Fractured Porous Media, *Soc. Pet. Eng. J.*, **25**(1), 14-26, February 1985.

Pruess, K., C. Oldenburg, and G. Moridis, TOUGH2 User's Guide, Version 2.0, Report LBNL-43134, Lawrence Berkeley National Laboratory, Berkeley, Calif., 1999.

Pruess, K., C. Oldenburg, and G. Moridis, TOUGH2 User's Guide, Version 2.1, Report LBNL-43134, Lawrence Berkeley National Laboratory, Berkeley, Calif., 2012.

Redlich, O. and J.N.S. Kwong, On The Thermodynamics of Solutions. *Chem. Rev.* **44**(1),

233–244, 1949. doi:10.1021/cr60137a013.

Riazi, M. R., and C. H. Whitson, Estimating Diffusion Coefficients of Dense Fluids, *Ind. Eng. Chem. Res.,* **32**, 3081-3088, 1993.

Rutqvist J. and C.-F Tsang, A Study of Caprock Hydromechanical Changes Associated with CO_2 Injection into a Brine Aquifer. *Environmental Geology*, **42**, 296-305, 2002.

Scheidegger, A. E., *The Physics of Flow Through Porous Media*, University of Toronto Press, Toronto and Buffalo, Third Edition, 1974.

Sleijpen, G.L.G. and D. Fokkema, BiCGSTAB(m) for Linear Equations Involving Unsymmetric Matrices with Complex Spectrum, *Electronic Transactions on Numerical Analysis*, **1**, 11 - 32, 1993.

Soave, G., Equilibrium Constants from a Modified Redlich–Kwong Equation of State, *Chem. Eng. Sci.*, **27**, 1197-1203, 1972.

Somerton, W.H., et al., Thermal Behavior of Unconsolidated Oil Sands, Paper SPE-4506, 48th Annual Fall Meeting of the Society of Petroleum Engineers, Las Vegas, NV, 1973.

Somerton, W.H., et al., High Temperature Behavior of Rocks Associated With Geothermal Type Reservoirs, Paper SPE-4897, 44th Annual California Regional Meeting of the Society of Petroleum Engineers, San Francisco, CA, 1974.

Stone, H.L., Probability Model for Estimating Three-Phase Relative Permeability, *Trans. SPE of AIME,* **249**, 214-218, 1970.

TecPlot, Inc., *Tecplot 10 User's Manual*, Bellevue, Washington, 2003.

Udell, K.S. and J.S. Fitch, Heat and Mass Transfer in Capillary Porous Media Considering Evaporation, Condensation, and Non-Condensible Gas Effects, 23rd ASME/AIChE National Heat Transfer Conference, Denver, CO, 1985.

van der Vorst, H.A., Bi-CGSTAB: A Fast and Smoothly Converging Variant of BiCG in the Presence of Rounding Errors, *SIAM J. Sci. Statist. Comput.*, **13**, 631 - 644, 1992.

vanGenuchten, M.Th., A Closed-Form Equation for Predicting the Hydraulic Conductivity of Unsaturated Soils, *Soil Sci. Soc.*, **44**, 892 - 898, 1980.

Vargaftik, N.B., *Tables on the Thermophysical Properties of Liquids and Gases*, 2nd Ed., John Wiley & Sons, New York, NY, 1975.

Vaughan, P.J., Analysis of Permeability Reduction During Flow of Heated, Aqueous Fluid Through Westerly Granite, in C.F. Tsang (ed.), *Coupled Processes Associated with Nuclear Waste Repositories*, 529 - 539, Academic Press, New York, 1987.

Verma, A.K., K. Pruess, C.F. Tsang and P.A. Witherspoon, A Study of Two-Phase Concurrent Flow of Steam and Water in an Unconsolidated Porous Medium, Proc. 23rd National Heat Transfer Conference, Am. Society of Mechanical Engineers, Denver, CO, 135–143, 1985.

Verma, A. and K. Pruess, Thermohydrologic Conditions and Silica Redistribution Near High-Level Nuclear Wastes Emplaced in Saturated Geological Formations, *J. of Geophys. Res.*, **93**(B2), 1159-1173, 1988.

Wagner W. et al., The IAPWS Industrial Formulation 1997 for the Thermodynamic Properties of Water and Steam, *ASME J. Eng. Gas Turbines and Power*, **122**, 150-182, 2000.

Walker, W.R., J.D. Sabey, and D.R. Hampton, Studies of Heat Transfer and Water Migration in Soils, Final Report, Department of Agricultural and Chemical Engineering, Colorado State University, Fort Collins, CO, 80523, April 1981.

Warren, J.E. and P.J. Root, The Behavior of Naturally Fractured Reservoirs, *Soc. Pet. Eng. J., Transactions, AIME,* **228**, 245-255, September 1963.

Webb, S.W., Gas-Phase Diffusion in Porous Media - Evaluation of an Advective-Dispersive Formulation and the Dusty Gas Model for Binary Mixtures, *J. Por. Media,* **1**(2), 187 - 199, 1998.

Webb, S.W. and C.K. Ho, Enhanced Vapor Diffusion in Porous Media - LDRD Final Report, Sandia National Laboratories Report SAND98-2772, Albuquerque, NM, 1998b.

Wilke, C. R., A Viscosity Equation for Gas Mixtures, *J. Chem. Phys.,* **18,** 517-519,1950.

Wu, Y., Pruess, K., and P. Persoff, Gas Flow in Porous Media with Klinkenberg Effects. *Transp*. *Porous Media*, **32**, 117-137, 1988.

Xu, T., Y. Ontoy, P. Molling, N. Spycher, M. Parini and K. Pruess, Reactive Transport Modeling of Injection Well Scaling and Acidizing at Tiwi Field, Philippines, *Geothermics,* **33**(4), 477 - 491, 2004.

Yaws, C., *Chemical Properties Handbook*, McGraw-Hill Education, 1999

PAGE LEFT INTENTIONALLY BLANK

APPENDIX

A Sample Input File

PAGE LEFT INTENTIONALLY BLANK