Title 40 CFR Part 191 Compliance Certification Application for the Waste Isolation Pilot Plant

Appendix NUTS

M

United States Department of Energy
Waste Isolation Pilot Plant

Carlsbad Area Office
Carlsbad, New Mexico

WIPP PA User's Manual for NUTS

M

WIPP PA

User's Manual

for

NUTS (Version 2.02)

Document Version 1.00

WPO #37927

MAY 29, 1996

CONTENTS

M

LIST OF FIGURES

M

LIST OF TABLES

1.0 INTRODUCTION

This document is the User's Manual for the WIPP radioisotope-mobilization-and-decay code named NUTS (version number 2.02). It discusses the code, its execution, and its performance in the context of the 1996 Waste Isolation Pilot Project (WIPP) Compliance Certification Application (CCA) Performance Assessment (PA), and in that context only. The manual identifies the code, its authors and expert consultants (Section 1). It describes the code's WIPP-PA purposes and functions (Section 2), provides recommended user training (Section 3), outlines the code's theoretical basis and numerical methods (Section 4), its inherent capabilities and limitations (Section 5), describes user interactions (Section 6), input files (Section 7), error messages (Section 8), and output files (Section 9), and provides examples of relevant input, output, and debug files in its Appendices as well as calculations of interest (distributed throughout). Efforts have also been made to assist users in understanding how to make the required input-option decisions, to provide insights into the structure and use of the code's various input and output files, and explain the code's operation. Several examples of input files are included to help in running NUTS efficiently for cases likely to be encountered.

1.1 Software Identifier

Code Name:	NUTS, a NUclide* Transport System
WIPP Prefix:	NUT
Version Number:	2.02
Date:	23 May 1996 (CMS Build Date)
Platform:	FORTRAN 77 for OpenVMS AXP, ver. 6.1, on a DEC Alpha VAX (see Section 2.3)

1.2 Points of Contact

Sponsor: Ali A. Shinta
Applied Physics, Inc.
2201 Buena Vista Drive, SE (fourth floor)
Albuquerque NM 87106
Voice: 505 766 9629
Fax: 505 766 9125

Consultant: Palmer Vaughn
Sandia National Laboratories
Department 6849
Albuquerque NM 87185
Voice: 505 848 0678
Fax: 505 848 0705

* Throughout this document, the word "Nuclide" is intended to be synonymous with "Isotope," and never to mean "an atomic nucleus with regard to energy level."

2.0 FUNCTIONAL REQUIREMENTS

R.1 First order decay of radioactive constituents in multiple member chains.

R.2 Retardation due to sorption of the chemical constituents via a linear equilibrium isotherm.

R.3 One-dimensional convection-dispersion-decay with spatially constant velocity and dispersion coefficients.

R.4 The ability to execute to completion under a data range geometry, and dimensions similar to that anticipated in the WIPP PA calculation and resulting in a qualitatively correct mass balance.

R.5 Time-varying Source Terms, with a slab source (non-point source).

R.6 Two-dimensional transport with convective, dispersion, decay, sorption, restricted to flow fields that are aligned with the grid if dispersion is not zero (i.e, no cross-terms are required in the governing equations) and restricted to dispersivities that are consistent with first-order upwind differencing.

R.7 Non-uniform, or stretched Cartesian grids.

R.8 Spatially variable material coefficients (e.g. Saturation, Porosity, Dispersitivity, Flow-Fields).

R.9 Solubility limit with linear precipitation model and colloid preferential solubility at the inventory limit.

R.10 Implementation of Neumann boundary conditions.

R.11 Implementation of Dirichlet conditions.

NUTS has a considerable number of external-interface requirements. They are listed in Section 2.4 of the code's Requirements Document.

3.0 REQUIRED USER TRAINING AND/OR BACKGROUND

To exercise NUTS, users should have (1) basic knowledge of open VMS, (2) basic facility with Digital Command Language, and (3) an overall understanding of Sandia's CAMDAT database, which is used in virtually every WIPP code (Rechard, 1992; and Rechard et al., 1993). User's should also have (4) access to the WIPP cluster of Alpha-VAX microcomputers or their functional equivalents.

To manipulate and/or interpret the results of NUTS as it is exercised in WIPP PAs, users should have (1) a basic understanding of the chemistry of radioactive decay, solution chemistry, sorption chemistry, fluid flow and fluid transport in porous media, (2) a basic understanding of partial-differential equations and integral calculus, as they apply in the mathematical formulation of physical principles and especially of conservation of mass*, (3) a generalist's conceptual understanding of numerical methods as they are applied to the numerical solution of the boundary-value problems of mathematical physics and chemistry, and (4) a basic overview understanding of the WIPP PA process, including conceptual models, scenarios, inventories, release routes, uncertainty sampling, input-data vectors, and a general familiarity with the files and functions of the WIPP codes that run prior to exercising NUTS, principally BRAGFLO and ALGEBRA, and those that exercise immediately after NUTS, principally CCDFGF (WIPP PA Dept [5 Volumes], 1992). An annotated flow diagram of NUTS's 1996 CCA code sequence is shown in Figure 3.1.

* Strictly speaking, the transport system is not mass conserving. Alpha decay involves a loss of mass. Thus, NUTS actually conserves the number of molecules of radioactive material involved in the decay process.

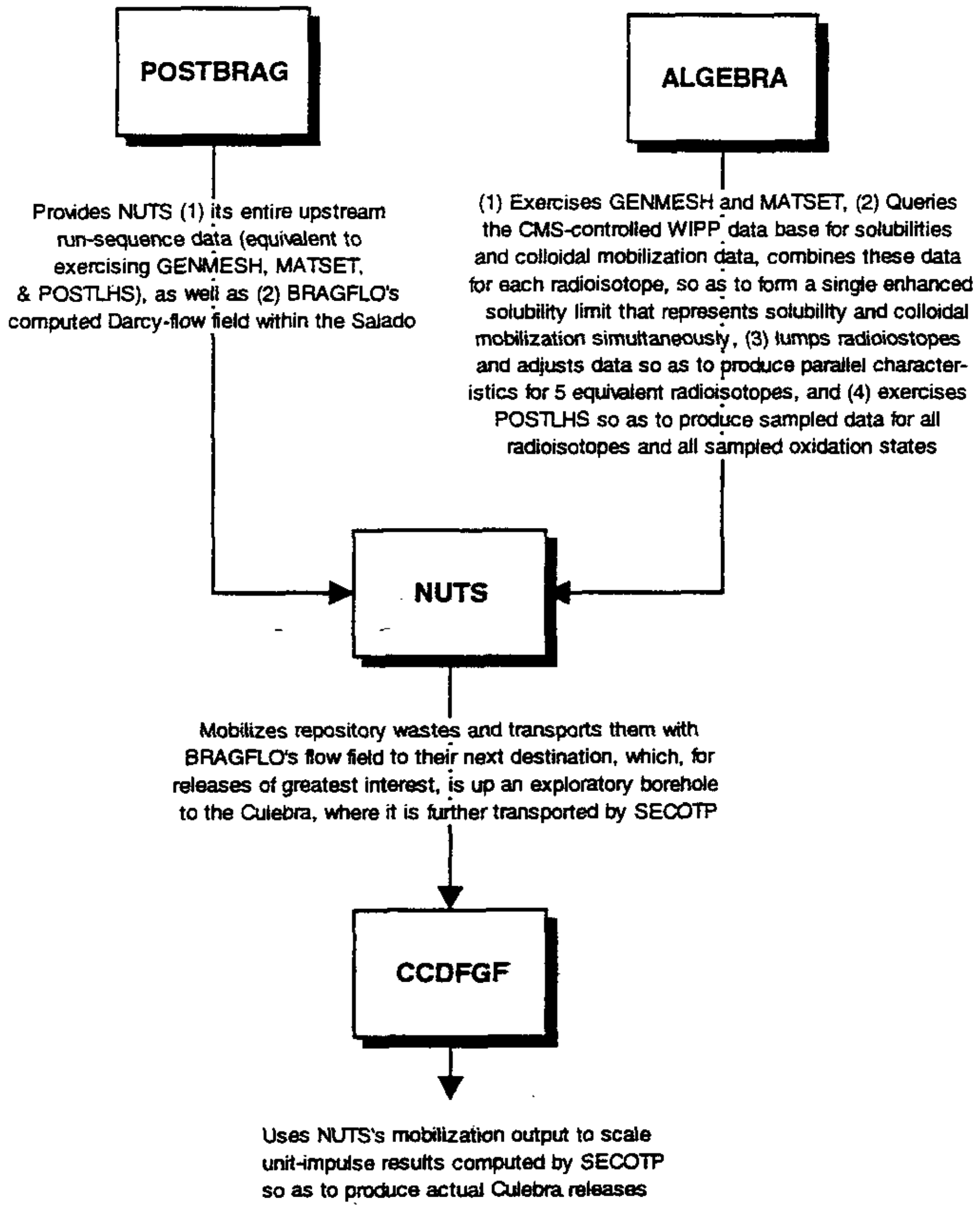
POSTBRAG
ALGEBRA
Provides NUTS (1) its entire upstream run-sequence data (equivalent to exercising GENMESH, MATSET, & POSTLHS), as well as (2) BRAGFLO's computed Darcy-flow field within the Salado
(1) Exercises GENMESH and MATSET, (2) Queries the CMS-controlled WIPP data base for solubilities and colloidal mobilization data, combines these data for each radioisotope, so as to form a single enhanced solubility limit that represents solubility and colloidal mobilization simultaneously, (3) lumps radioiostopes and adjusts data so as to produce parallel characteristics for 5 equivalent radioisotopes, and (4) exercises POSTLHS so as to produce sampled data for all radioisotopes and all sampled oxidation states
NUTS
Mobilizes repository wastes and transports them with BRAGFLO's flow field to their next destination, which, for releases of greatest interest, is up an exploratory borehole to the Culebra, where it is further transported by SECOTP
CCDFGF
Uses NUTS's mobilization output to scale unit-impulse results computed by SECOTP so as to produce actual Culebra releases

Figure 3.1: The code sequence for NUTS as it is exercised in the 1996 CCA.

4.0 DESCRIPTION of the MODEL and METHODS

4.1 Background

4.1.1 NUTS's General Role in the CCA PA

NUTS's principal CCA PA role is to estimate the radioactive contaminant load mobilized into the brine phase of the brine/gas mixture that seeps or flows through and around the WIPP repository's decommissioned waste panels. Mobilization by any process, for example dissolution or suspension on colloids, is modeled as taking place instantaneously. The contaminants introduced into the brine are the aged radioisotopes that are assumed to reside in the repository at the time of decommissioning plus any progeny of those radioisotopes that may have been produced through natural decomposition. Thus, a detailed inventory of the radioactive isotopes stored in the repository at the time of decommissioning must be provided to NUTS before it may be exercised.

NUTS plays no role in the physics of the fluid flow in or near the repository. The time history of the detailed (Darcy) volumetric flow rate throughout the repository must be provided to NUTS by an independent hydrological-flow code. In the case of the CCA, that code is BRAGFLO.

In the CCA WIPP assessment scenarios, brine is assumed to enter the repository panels in either of two very different ways, namely: (a) by natural seepage from the surrounding Salado formation, and (b) by various sorts of unnatural flows induced by exploratory boreholes. In the case of undisturbed operation, brine can seep into the repository through the disturbed rock zone from the surrounding undisturbed halite and marker beds. In the case of a repository breached by an exploratory borehole, the brine could flow into and through the repository via the pipe-like channels created by the borehole(s). The most intense flow would occur if the borehole also penetrated a deep pressurized brine pocket, and some boreholes are assumed to do so.

The natural time scales associated with deep, tight-media, groundwater flows are typically centuries, and WIPP intrusion scenarios include temporal lapses of millennia prior to the hypothesized breaching of repository waste panels by boreholes. Moreover, EPA regulations extend to 10,000 years after decommissioning. On these time scales, radioisotopes of interest exhibit significant natural decay by which they transform to other radioactive and non-radioactive isotopes and/or compounds in well-established ways. Thus, it is required to quantify the decay process and maintain a running record of the decayed contents of the repository as well as all the products of decomposition from the time of decommissioning onward to 10,000 years. NUTS does that.

In the context of the CCA, NUTS has a single principal area of application. It is this: given the defining characteristics of the flow of uncontaminated brine into the repository's waste panels, NUTS is required to model the dissolution and sorption of radioisotopes into the brine that wets the waste panels, and to calculate, given the fluid flow rates, the rate at which each isotope that was initially present in the repository, including the radioactive progeny of those isotopes, exits the repository via the now contaminated brine flow. The exit rate of contaminants depends on (i) the outflow rate of the brine and (ii) the concentration of isotopes within the flowing brine. The first quantity is provided in advance by BRAGFLO, which analyses the 2-phase Darcy flow throughout the Castile, Salado, Rustler, and Dewey Lake Formations throughout the entire Land Withdrawal Act region. NUTS is required to calculate the second quantity (concentration of each radioisotope), and to do so by treating all the important physical processes that can affect transport via a brine carrier flowing through a porous medium.

Mobilization is assumed to take place instantaneously and to maximum capacity, i.e. to the "solubility" limit, given adequate stocks. In the CCA, the "solubility limit" is actually a maximum mobilized concentration that includes the maximum concentration of an element mobilized on four types of colloids as well as the solubility of the element. Construction of this maximum mobilized concentration is performed before the NUTS calculation using ALGEBRA.

Because of natural decay processes, it is entirely possible to find compounds in the effluent that were not present in the initial repository inventory.

M

4.1.2 Overview of NUTS's Capabilities

NUTS is a multidimensional, multicomponent, radioactive-contaminant transport code. It is designed to apply to single-porosity (SP), dual-porosity (DP), and/or dual-permeability (DPM) porous media. In two dimensions, the discretization employs five-point finite-difference methods. NUTS is designed to treat an enormously broad spectrum of transport problems. However, many of its options have been disabled in the CCA calculations for the sake of simplicity. CCA-disabled options are described briefly herein, and are clearly noted as disabled for CCA calculations.

The model simulates radioisotope transport through porous media and includes first-order radioactive decay processes. However, the simulator is not limited to radioactive materials. The transport of materials of all kinds, radioactive or not, may be simulated. In CCA applications, only radioisotopes are transported by NUTS.

In treating sorption between the waste and the media that surround it, NUTS allows for three types of sorption isotherms. They are: (1) linear, (2) Freundlich, and (3) Langumir equilibrium isotherms. **Only option (1) is tested for regulatory calculations, and none was used in the CCA calculations. A type of sorption onto colloids, however, was modeled in the ALGEBRA calculation run prior to NUTS. It is described in Section 4.5.1.**

NUTS normally models hydrodynamic dispersion under the assumption that the porous media through which the transport occurs are dispersively isotropic. However, for CCA calculations, both molecular and mechanical dispersion are taken to be zero. The effects of molecular dispersion are truly small (O(10m)). Mechanical dispersion is proportional to velocity and other physical and geometric factors (see Section 4.2.3). Ignoring transverse dispersion tends to keep solutes in the higher-velocity regions of the flow field. Ignoring lateral dispersion ignores streamwise elongation, which is reasonable given that EPA limits are defined in terms of time-integrated measures. For example, when and how a signal reaches the land-withdrawal boundary are unimportant, as long as the signal reaches the boundary in 10,000 years. Numerical dispersion and statistical dispersion (due to sampling) are both present in NUTS results, and mimic physical dispersion.

"Dissolution" of waste components into the brine carrier medium is modeled, and the possibility of their precipitation during migration is included, should solutes exceed their local "solubility" limits. Precipitates are required to undergo decay and are permitted to "redissolve" in the brine if the concentration drops below the solubility limit.

Representation of multiple radioactive sites (repositories) is also possible. In that case, the contribution from each site to the component concentration and precipitation, if any, at each computational node can be found. A similar technique may be used to treat a daughter isotope generated from the decay of different parents. **However, use of this aspect of the code is not contemplated for the WIPP CCA PA, and it is therefore not tested.**

The system of governing partial-differential equations (mass conservation for each radioisotopic constituent) is discretized and solved sequentially to determine the contribution from a decaying parent to the immediate daughter. In the sequential method, the solution proceeds progressively from the top of each radioactive chain. Therefore, the contribution to any daughter from its decaying parent is available. This approach avoids the numerical problems associated with inverting a large, sparsely-populated matrix in which the bands are not well structured.

NUTS was developed specifically to assist in performance assessments of the WIPP. **It is that application, and specifically the CCA calculation, that will be described herein.** However, in its broadest sense, NUTS can be used in other applications involving contaminant transport of all kinds through porous media. Table-1 provides a summary for the features used in the CCA calculations.

4.2 Theoretical Foundations of Transport in Fractured Porous Media having Dual-Porosity and Dual-Permeability

In the textbook approach to dual-permeability, the partial differential equations (PDEs) representing transport in a fractured, porous medium are normally written as (1) the PDEs governing transport along the fractures, and (2) the PDEs governing transport through the matrix. The two PDEs are coupled by a transfer function that represents material transport between the two media (Kazemi et al, 1976; Hill and Thomas, 1985; Litvak, 1985; Coats, 1989). That is the approach taken herein.

4.2.1 Introduction to Fractured Porous Media

A fracture can be defined as a surface of discontinuity resulting from loss of cohesion of rock materials due to rupture. Most fractures in brittle rocks of low porosity occur as a result of tectonic processes. These fractures are usually large in size and extent, and are called

Table 1. NUTS Features Used in the CCA Calculations

NUTS Features	Features in the CCA Calculations	
	Used	Disabled
Single-porosity, fracture		x
Single-porosity, matrix	x	
Double-porosity		x
Double-permeability		x
Advective transport	x	
Diffusive-dispersive transport		x
Sorption		x
Colloid transport+		x
Decay	x	
Gas phase transport		x
Temperature dependency		x
Multiple sites		x
Precipitation	x	
Solubility limit	x	
Interior sources	x	

+ Colloids are lumped with dissolved inventory (see Section 4.5.1) macrofractures. In less brittle rocks with higher intergranular porosities, the resulting fractures are usually smaller and less extensive, and are called microfractures, or fissures. Fractures generated by the state of stress in the earth's crust can be attributed to (van Golf-Racht, 1982):

1. folding and faulting;
2. rock volume shrinkage and the dehydration process;
3. volume shrinkage associated with heat loss; and
4. differential stress on the plane of weakness due to deep erosion of overburden rocks.

Thus, fractured porous media are typically conventional sedimentary materials possessing discontinuities introduced later by some sort of physical activity. Digenesis processes, such as mineral deposition (calcite), and dissolution of matrix and formation of stylolites may be associated with fracture formation. Fractures may be regular or irregular, closed and filled with cement, or open to flow. Most-fractured systems, occurring in a large variety of rock mineralogy, are described adequately (for hydrological purposes) by specifying the fracture spacing, orientation, length, and degree of cementation. These data are usually obtained from cores and outcrops coupled with rock mechanics studies, well logging, and seismic-profile distortion. Because of the inherent complexity associated with the characterization of these mineral discontinuities, porous-medium parameters such as porosity, permeability, etc., normally have a wide range of numerical uncertainty. Accordingly, the classical models of fluid flow and transport in porous media have been shown to be inadequate to represent a heterogeneous system in which the heterogeneities are due to fractures, vugs, and channels. A new and more complex model is required, the complexity deriving from the existence of two entirely different but interconnected paths for fluid flow and transport. Because natural fractures tend to be irregular in their distribution, orientation, and extension, some sort of reasonable idealization is required as a basis for a reasonable physical and mathematical description of the medium.

The basic theory of fluid flow in fractured porous media was introduced by Barenblatt and Zeltov (1960), who applied a continuum approach. Their approach is similar to the approach used in classical fluid-dynamical theories in which the physical properties of a cluster containing many molecules is allowed to persist in the limit of smallness. Barenblatt and Zeltov choose a smallest physical scale that lies between microscopic and macroscopic representations of fluid properties in regular porous media. In their approach, a representative elementary volume (REV), also called a control volume, is considered. It is much larger than pore size (i.e., it contains many pores and grains) but much smaller than the total domain considered. The REV's average porosity, permeability, pressure, saturation, concentration, and flux are ascribed to the *mathematical point residing at the center of the* REV (see Bear, 1972), *which forms the basis of* their hypothesis for a continuum. In applying a strictly continuum approach to dual-porosity or dual-permeability systems, one can assign a pair of average properties to each mathematical point; one representing fracture properties and one representing matrix properties. In other words, the medium consists of two overlapping continua that communicate one with another. In this formulation, the REV contains many matrix blocks and fractures (see Figure 4.1). In actual double-porosity conceptualizations, the fractures, because of their high permeability relative to the matrix and high intensity (small spacing), are assumed to constitute a continuous medium. The matrix blocks between them are discontinuous and form the main storage region for the fluid. In double-permeability formulations, fracture spacings are larger, and matrix-to-matrix flow is permitted in the mathematical formulation of the problem*.

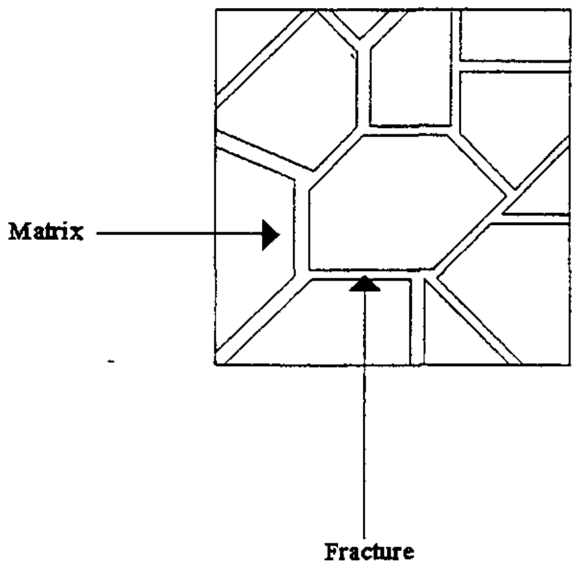

Figure 4.1 Schematic Diagram of Fractured REV.

4.2.2 Basic Dissolution Processes

When salty ground water comes into contact with repository wastes, dissolution of the soluble isotopes begins to take place and continues until equilibrium concentrations are attained, or until all undissolved supplies of radioisotopes are consumed. The capacity of WIPP brines to dissolve radioisotopes is controlled by temperature, pressure, and chemical properties of the brine and of the solutes. The main physical principal on which NUTS's transport equations are based is conservation of mass, and the principal dependant variable in which those equations are written is C, the concentration of solute (i.e., radioisotope) in the underlying fluid (i.e., brine). A pair of conservation equations arises for each radioisotope, one for the fracture domain and one for the matrix domain. The equation pairs are coupled one to another by transfer functions that

* For more information about these conceptualizations, the reader is referred to Beckner *et al.* (1991), Chen *et al.* (1987),Dean and Lo (1988), DeSwaan (1978, 1982), Duguid and Lee (1977), Dutra and Aziz (1991), Dykhuizeen (1990), Evans (1982), Gilman and Kazemi (1983, 1988), Gilman (1986), Huang-Zhang (1983), Kazemi and Merrill (1979), Kazemi *et al.* (1989), Narasimhan (1982), Pruess and Narasimhan (1985), Reiss *et al.* (1973), Rossen (1979), Saidi (1983), Shinta and Kazemi (1993), Sonier *et al.* (1986), Thomas et al (1983), Wu and Rruess (1988).

represent the flux of a given radioisotope between the matrix and fracture domains of the flow. The equation pairs are also coupled one with another by radioactive decay processes. Thus, a complex system of coupled partial differential equations is anticipated.

The concentration, C, of a given radioisotope must be non-negative everywhere in the domain. However, it also has an upper bound, equal to the maximum solubility limit for each element of the waste in whichever brine or combination of brines is present. There are two principal sources of brine, one of Salado origin and one of Castile origin (WIPP PA Dept, 1992, vol 1), and the brines are chemically different. The different brines come into play in the different scenarios. NUTS must be given the solubility limit of all soluble radioactive elements as part of its input data. It allocates the solubility limit for the isotopes according to their mole fractions. If radioisotope concentrations should exceed their solubility limits, NUTS will precipitate the excess in the grid block. NUTS subjects precipitates to decay and treats precipitation as a reversible process. If the concentration of a solute drops below the solubility limit, WIPP brines are permitted to redissolve from the precipitate a mass sufficient to restore the solubility limit of that solute, assuming sufficient precipitate is available.

4.3 Differential Equations Governing Transport in Fractured Porous Media

The mathematical formulation for the dual-continuum approach requires a mass-conservation equation for each continuum. Each such equation contains a coupling term (transfer function) that represents intercommunication (i.e., transport) between the two domains. The continuity-equation development that follows is general and therefore applies to both matrix and fracture domains, proving the coupling term is added. In porous media, the principal focus is on the transport of a solute whose intensity is measured in terms of a fluid concentration measured in Kg of solute per Kg of fluid. The mass-conservation equation (continuity equation) is derived by considering a REV shaped like a rectangular parallelepiped in the direction of the Cartesian coordinates (x, y, z), and having an infinitesimal volume of $\delta x \delta y \delta z$. The mass conservation principle can be stated in words as follows:

$$\left| \textit{Mass Flux in} - \textit{Mass Flux out} + \textit{Source / Sink} = \textit{Accumulation} \right|_{REV}$$

Boundary fluxes and internal sources derive from a myriad of physical processes, which we will describe and quantify individually in the subsections that follow.

4.3.1 Advective Transport

The mass-flux vector of the solute is represented as q^*, and its three Cartesian components as q^*_x, q^*_y, q^*_z, as shown in Figure 4.2.

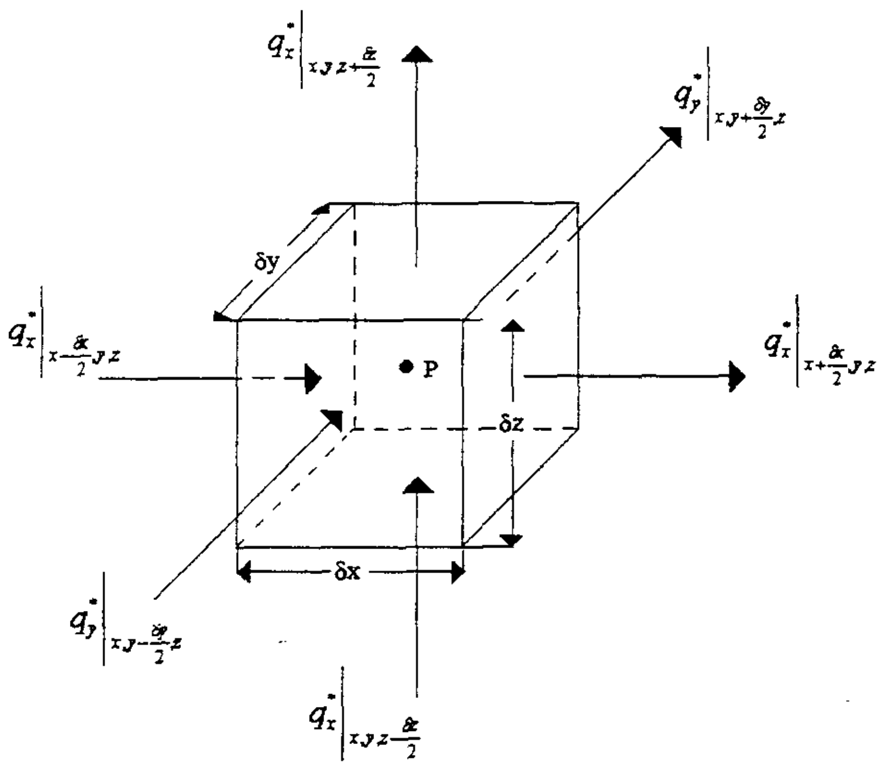
$q_z^*\big|_{x,y,z+\frac{\delta z}{2}}$
$q_y^*\big|_{x,y+\frac{\delta y}{2},z}$
δy
$q_x^*\big|_{x-\frac{\delta x}{2},y,z}$
P
$q_x^*\big|_{x+\frac{\delta x}{2},y,z}$
δz
δx
$q_y^*\big|_{x,y-\frac{\delta y}{2},z}$
$q_z^*\big|_{x,y,z-\frac{\delta z}{2}}$

Figure 4.2 Schematic Diagram Showing the REV and the Fluxes q*.

The net mass inflow associated with flow in the x-direction during the time period δt is:

$$\left[q_x^*\Big|_{x-\frac{\delta x}{2}} - q_x^*\Big|_{x+\frac{\delta x}{2}} \right]_{y,z} \delta y \delta z \delta t . \qquad 4.1$$

Using Taylor's series to expand around the central point P(x,y,z) of the REV, Equation 4.1 becomes:

$$\left[\left\{ q_x^* - \frac{\delta x}{2}\frac{\partial q_x^*}{\partial x} + \frac{(\delta x)^2}{8}\frac{\partial^2 q_x^*}{\partial x^2} - \ldots \right\} - \left\{ q_x^* + \frac{\delta x}{2}\frac{\partial q_x^*}{\partial x} + \frac{(\delta x)^2}{8}\frac{\partial^2 q_x^*}{\partial x^2} + \ldots \right\} \right] \delta y \delta z \delta t . \qquad 4.2$$

Simplifying and neglecting higher order terms (which go to zero in the limit), Equation 4.2 reduces to:

$$-\frac{\partial q_x^*}{\partial x}\delta x\delta y\delta z\delta t.$$

Repeating the same procedure in the other two directions, the net mass of solute accumulated due to flux across the boundaries during the time period δt is:

$$-\left[\frac{\partial q_x^*}{\partial x}+\frac{\partial q_y^*}{\partial y}+\frac{\partial q_z^*}{\partial z}\right]\delta x\delta y\delta z\delta t. \qquad \textbf{4.3}$$

The net mass of solute accumulated in the control volume from internal sources (and lost to sinks) during the time period δt is:

$$q^*_{s/s}C'_{i/p}\delta x\delta y\delta z\delta t, \qquad \textbf{4.4}$$

where all symbols are defined in the summary list below. The net increase in mass of solute in the control volume during time δt may also be written as the local time rate of change of solute mass in the volume times δt, that is, as:

$$\frac{\partial(\phi S\rho C')}{\partial t}\delta x\delta y\delta z\delta t. \qquad \textbf{4.5}$$

Combining the terms in Equations 4.3, 4.4 and 4.5 according to the word equation stated in Section 4.3 results in:

$$-\left[\frac{\partial q_x^*}{\partial x}+\frac{\partial q_y^*}{\partial y}+\frac{\partial q_z^*}{\partial z}\right]\delta x\delta y\delta z\delta t+q^*_{s/s}C'_{i/p}\delta x\delta y\delta z\delta t=\frac{\partial(\phi S\rho C')}{\partial t}\delta x\delta y\delta z\delta t. \qquad \textbf{4.6}$$

Dividing Equation 4.6 by $\delta x\delta y\delta z\delta t$ and taking the limit as the control volume approaches zero leads to:

$$-\nabla . q^* + q^*_{s/s}C'_{i/p}=\frac{\partial(\phi S\rho C')}{\partial t}, \qquad \textbf{4.7}$$

where the nomenclature is defined as follows:

$$q^* = \rho\upsilon C' \qquad \textbf{4.8}$$

and

$q^*_{s/s}$ = solvent sink/source mass rate per unit volume (kg/s/m^3)

ρ = solvent density (kg/ m^3)

ϕ = porosity of porous medium (dimensionless fraction)

v = solvent advective velocity (m/s)

C' = solute concentration (kg/kg)

$C'_{i/p}$= injected/produced solute concentration (kg/kg)

t = time (s)

S = saturation (dimensionless fraction)

If the entire Equation 4.7 is divided by the solvent density ρ (a constant), and q* is replaced by its equivalent value (Equation 4.8), Equation 4.7 becomes:

$$-\nabla.\mathrm{v}C + qC_{i/p} = \frac{\partial(\phi SC)}{\partial t}, \qquad \textbf{4.9}$$

where:

C = solute concentration (kg/ m^3)

q = solvent sink/source volumetric rate per unit volume ($m^3/s/m^3$)

Equation 4.9 as written applies to the fluid domain as a whole (i.e., matrix plus fracture). However, as was said, we prefer to divide the domain into two separate flow regimes, matrix and fracture, and to allow the two to intercommunicate through a transfer function. The transfer function represents mass flux from the fracture to the matrix. Therefore, the transfer function that appears as a source in the matrix equation will appear as a sink (i.e., the same term, but with its sign reversed) in the fracture equation. The continuity equation that applies to either domain is written as follows:

$$-\nabla.\mathrm{v}_f C_f + q_f C_{fi/p} - \overline{\tau}_{m/f} C_{m/f} = \frac{\partial(\phi SC)_f}{\partial t}, \text{ for the fracture, and} \qquad \textbf{4.10a}$$

$$-\nabla.\mathrm{v}_m C_m + q_m C_{mi/p} + \overline{\tau}_{m/f} C_{m/f} = \frac{\partial(\phi SC)_m}{\partial t}, \text{ for the matrix,} \qquad \textbf{4.10b}$$

where $C_{m/f}$ is the upstream concentration of solute in kg/m^3 in either the fracture or the matrix depending on the direction of the solvent flow (see Section 4.4.1 for the computation of the upstream weighting factor), and $\overline{\tau}$ is the volumetric exchange rate, m^3 /s/m^3 of solvent mass transfer between the fracture and the matrix. The + and - signs in the transfer terms in Equation 4.10 imply that the transfer acts as a source in one continuum and as a sink in the other. That is, what is lost from the matrix is gained by the fracture and vice versa.

4.3.2 Advective Transport Across Fracture/Matrix Interfaces

Transfers at fracture/matrix interfaces are dominated by a nearly steady-state Darcian flow represented by $\tau_{wm/f}$. The magnitude of these flow terms and their idealization are controlled and provided by the fluid flow code as volumetric rates.

The fracture /matrix transfer function τ for the brine could be evaluated in the fluid flow code by the following formula:

$$\tau_w = \sigma k_m \left(\frac{k_r}{\mu}\right)_{m/f} \left[P_f - P_m - \gamma_{m/f}\left(D_f - D_m\right)\right], \qquad \mathbf{4.11}$$

where

D	depth [m],
k	permeability [m^2],
k_r	relative permeability [dimensionless],
P	pressure [Pa],
γ	static pressure gradient [Pa/m],
μ	viscosity [Pa/s],
σ	shape factor [m^{-2}].

M

Subscripts

f = fracture
m = matrix
w = brine

4.3.3 Diffusive and Dispersive Transports

Advection represents the transport of solute as it is carried along by the motion of the fluid. Diffusion and dispersion represent transport of the solute due to its motion (or apparent motion) through the fluid. Molecular diffusion results from thermal motion of solute molecules. In porous media, the diffusion coefficient (sometimes called the apparent molecular diffusion coefficient) is calculated from the free-water molecular diffusion coefficient D^*_m adjusted for the tortuous path and the water content (saturation) of the transported phase of interest. Mechanical dispersion is the mixing process resulting from the fluctuation in the ground-water velocity due to the heterogeneity of the porous media. Thus, if a sharp contaminant front is introduced into a porous-medium flow, the front will rapidly thicken as though an efficient diffusion mechanism were at work, whereas an abrupt interface would be expected to persist if only advective transport was at work. Among the processes that lead to thickening and distortion of a hypothetical front are sorption-desorption processes, molecular diffusion, and mechanical mixing (Bear, 1988). The last two are usually combined into a single effect called hydrodynamic dispersion.

In practical applications of solute-transport theory, it is customary to account for both the molecular and mechanical components of hydrodynamic dispersion. NUTS utilizes Fick's law to describe the dispersive contribution to mass transfer in both the fracture and matrix components of the flow. In vector notation, it is written:

$$F_D = \phi S K \nabla C \,, \qquad \textbf{4.12}$$

where F_D is the dispersion flux. Generally (but **not** in CCA calculations), NUTS treats the material dispersivity as a 2nd-rank tensor, with the additional assumption that the porous transport medium is isotropic. Bear (1988) has shown that this assumption is equivalent to representing the dispersivity tensor as a positive-definite, symmetric, 2nd-rank tensor. The most general dispersivity tensor would be the one shown in component form by the following matrix:

$$K = \begin{vmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{vmatrix} . \qquad \textbf{4.13}$$

However, because the dispersion tensor is assumed to be symmetric, six of its coefficients are related as follows: $K_{xy} = K_{yx}$, $K_{xz} = K_{zx}$, and $K_{yz} = K_{zy}$, or in indicial notation,

$$K_{ij} = \frac{D_m^*}{\tau^*}\delta_{ij} + \frac{\alpha_T}{\phi S}|v|\delta_{ij} + \frac{v_i v_j}{\phi S|v|}(\alpha_L - \alpha_T), \qquad \textbf{4.14}$$

where nomenclature is defined in the listing below. If the chosen coordinate system happens to coincide with the principal axes of dispersion, all the off-diagonal values will become zero in the tensor matrix. In that coordinate system, the dispersion coefficient in the x-direction for the fracture flow and the matrix flow become (Bear, 1993):

$$K_{fx} = \left\{ \frac{D_m^*}{\tau_f^*} + \frac{1}{\phi_f S_f |\vec{v}_f|}\left[\alpha_{Lf} v_{fx}^2 + \alpha_{Tf}\left(v_{fy}^2 + v_{fz}^2\right)\right]\right\}, \qquad \textbf{4.15}$$

and

$$K_{mx} = \left\{ \frac{D_m^*}{\tau_m^*} + \frac{1}{\phi_m S_m |\vec{v}_m|} \quad \left[\alpha_{Lm} v_{mx}^2 + \alpha_{Tm} \left(v_{my}^2 + v_{mz}^2 \right) \right] \right\}, \tag{4.16}$$

where the terms between braces are the x-component of the dispersivity tensor within (i) the fracture flow and (ii) the matrix flow, respectively, and where,

D^*_m	= *free water molecular diffusion coefficient* [m^2/s],
τ^*	= tortuosity[+] [dimensionless],
α_L	= longitudinal dispersivity [m],
α_T	= transverse dispersivity [m],
δ_{ij}	= kronecker delta [dimensionless],

and the velocity vector $\vec{v}$ is given by:

$$|\vec{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}\,. \tag{4.17}$$

where v_x, v_y, v_z are the fluid flow velocities [m/s] in x , y , z directions, respectively. Similar formulations apply for the y and z components of the dispersivity tensor. The present version of NUTS applies the above formulation to diagonal components of the dispersivity tensor and ignores off-diagonal contributions.

Dispersion in the fracture and matrix is generally small and can be neglected in comparison to the advective part of the transport. Moreover, dispersion can always be a significant means of mass transfer between the fracture and matrix flow zones. Fick's law for the transport between the fracture and the matrix is written as:

$$F_{Dm/f} = \phi_m S_m K_{m/f} \left(C_f - C_m \right) \tag{4.18}$$

Bear (1993) gives the dispersion coefficient for fracture/matrix flow as:

$$K_{wm/f} = \sigma \left\{ \frac{D_m^*}{\tau_m^*} + \alpha_m \frac{|v_{m/f}|}{\phi_m S_m} \right\}, \tag{4.19}$$

where

[+] Throughout this document, tortuosity is defined as the ratio of the square of the tortuous path length to the square of the mean path length. This definition results in tortuosities greater than or equal to unity. However, the reader should bear in mind that some formulations found in the literature use the inverse of the definition, leading to tortuosities less than or equal unity.

$v_{m/f}$ is the velocity of liquid transfer [m/s], and
σ is a shape factor.

In Equation 4.19, it is assumed that the contribution from the fracture dispersion is negligible compared with the matrix (the fracture aperture is very small compared with the matrix block dimensions).

The shape factor σ is defined, in general, as

$$\sigma = \sum_{S} \frac{A_m}{V_m d}, \qquad \textbf{4.20}$$

M

where

S is all the faces in the matrix block subjected to flow,
A_m is the surface area of the face subjected to flow [m^2],
V_m is the volume of the matrix block [m^3], and
d is the distance between the face and the center of the matrix block [m].

Sigma plays the role of Δn in $\Delta C/\Delta n$, which is the normal gradient to an interfacial surface. For a rectangular parallelepiped matrix block, the shape factor is calculated (Kazemi et al., 1976) to be,

$$\sigma = 4\left(\frac{1}{L_x^2} + \frac{1}{L_y^2} + \frac{1}{L_z^2}\right), \qquad \textbf{4.21}$$

where L_x , L_y , and L_z are the block dimensions in x, y, and z directions.

Collecting intermediate results, if Equations 4.12 and 4.18 are added to Equations 4.10, the resulting equations will be

$$\nabla.\phi_f S_f K_f \nabla C_f - \nabla.v_f C_f + q_f C_{fi/p} - \overline{\tau}_{m/f} C_{m/f} - \phi_m S_m K_{m/f}\left(C_f - C_m\right) = \frac{\partial(\phi SC)_f}{\partial t} \qquad \textbf{4.22}$$

for the fracture, and

$$\nabla.\phi_m S_m K_m \nabla C_m - \nabla.v_m C_m + q_m C_{mi/p} + \overline{\tau}_{m/f} C_{m/f} + \phi_m S_m K_{m/f}\left(C_f - C_m\right) = \frac{\partial(\phi SC)_m}{\partial t} \qquad \textbf{4.23}$$

for the matrix.

4.3.4 Asymmetric Dispersion

The mechanical-dispersion expression introduced in the transport equation is little more than a global expression that endeavors to compensate for our lack of detailed information regarding the microscopic nature of the heterogeneities inherent in porous media. However, examination of Equations 4.15 and 4.16 suggests the effective dispersion coefficient will always be positive. Therefore, mechanical dispersion always follow and enhance the trends established by molecular diffusion. It is reasonable to assume molecular diffusion follows Fick's law because of the random thermal motion of molecules. Therefore, molecular diffusion can expand symmetrically in all directions providing only that there exists a concentration gradient. This argument, if applied to the mechanical dispersion, will lead a well-known problem in which dispersion is directed opposite to the direction of the flow. Opposite-direction dispersion is physically incorrect and is associated with *a priori* ignorance of the flow direction (i.e., v^2 is always positive). In order to overcome this problem, NUTS has a module for Asymmetric dispersion that allows the dispersion to go only in the direction of the flow regardless of method used to discretize the transport equation. Because the method used is conservative, the only effect this treatment has on the transport, is shifting the centroid of the mass further downstream. In other words, the movement of the transport front will be little faster in the case of Asymmetric dispersion, as is depicted in Figure 4.3.

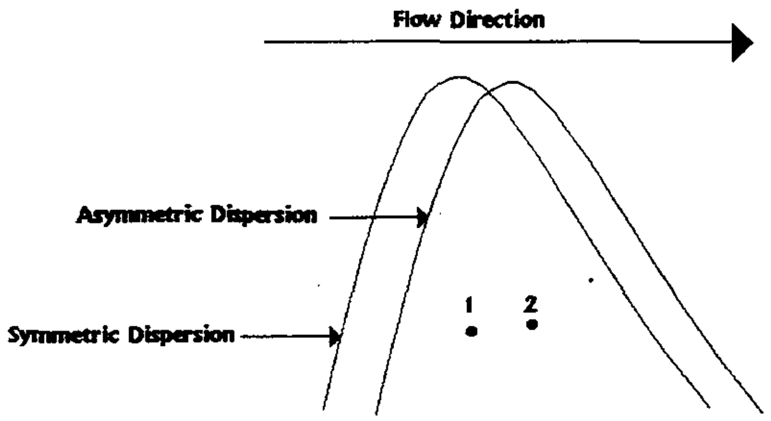

Figure 4.3 Symmetric versus Asymmetric Dispersion Fronts.

Asymmetric Dispersion is implemented numerically by computing a binary (0,1) parameter, κ as follows:

$$\kappa = \begin{vmatrix} 1 & \textit{if the concenteration gradient is in the same direction as the flow gradient} \\ 0 & \textit{otherwise} \end{vmatrix}$$

The parameter κ is then used as a multiplier only to the dispersion part of the transport equation (the molecular diffusion part is symmetric). This manipulation is performed while preparing the diagonal entries of the numerical matrix to assure mass conservation of the system.

4.3.5 Sorption-Desorption

Sorption refers to the ability of certain solids to extract substances preferentially from solution and deposit them on their surfaces. The solution can be either gaseous or liquid. Two general types of sorption are treated; physical sorption and chemical sorption. Physical sorption is a reversible process and arises on account of van der Waal intermolecular forces on the sorbing solid and the sorbed substance. In porous media, the sorbed substance may not react with the sorbing solid, but it can penetrate it via interstices, if a wettability condition is in effect. Chemical sorption is a result of chemical interaction between the sorbate and the sorbent. The chemical bond associated with this attraction is generally much greater than that found in the physical sorption. Ion exchange is sometimes classified as a kind of the chemical sorption.

Desorption is the process in which the solid phase releases some of the sorbed material back into solution. Sorption-desorption as a process depends on, pressure, temperature, and the chemical composition of the sorbate and the sorbent. It is usually described by an empirical relationship relating the concentration of the solute and the concentration of sorbate at a constant temperature, and is called an "isotherm." If a first-order kinetic model is used to describe the sorption-desorption process, one can write the following reaction-rate equation,

$$\frac{\partial R}{\partial t} = k_1 C - k_2 C_s , \qquad 4.24$$

where R is the net rate of reaction, k_1 is the sorption reaction constant, k_2 is the desorption reaction constant, C is the concentration of sorbent in kg per kg of fluid, C_s is the concentration of sorbate in kg/kg dry porous media. In the limit of sorptive equilibrium,

$$\frac{\partial R}{\partial t} \to 0 \qquad \text{as } t \to \infty\text{, and}$$

$$\frac{k_1}{k_2} = \chi = \frac{C_s}{C} \Rightarrow C_s = \chi C , \qquad 4.25$$

where χ is a parameter that dependents on the shape of the equilibrium isotherm and the equilibrium conditions. For a linear isotherm $\chi = k_d$, the linear sorption equilibrium constant. The mass rate of sorption per unit volume, F_S, in kg/s/m^3 can therefore be determined from

$$F_S = \frac{\partial}{\partial t}(1-\phi)\rho_s C_s, \qquad \textbf{4.26}$$

where

ρ_s = rock grain density kg/m^3.

4.3.6 Matrix/Fracture Boundary Sorption

Many dissolved WIPP chemicals are subject to sorption onto the solid surfaces that form the boundaries in the rock matrix through which they flow. This phenomenon takes place because of the locally irregular atomic and molecular properties of the surface skins of virtually all materials. The mass transfer between a solute and the solid surface that bounds it is controlled by the chemical properties of the solid surface, the liquid transport medium, and the solute. Boundary sorption manifests itself as a retarding factor and is expressed in NUTS through the variables C_{sm} and C_{sf}, which represent the sorbed concentration of any specific radionuclide per unit mass of rock. It is conventional to relate C_{sm} and C_{sf} to the solute concentration by any of various empirical relations called "isotherms." As the name suggests, these isotherms are families of empirical graphs of sorbate concentration versus solute concentration at constant temperature. NUTS accommodates three equilibrium isotherms. They are the:

1. Linear sorption isotherm in which adsorbate concentration is related to solute concentration by:

$$C_s^{n+1} = k_d\, C_w^{n+1}, \qquad \textbf{4.27}$$

where

n+1 represents the implicitness of the concentration (current time step), and
k_d is the equilibrium partition coefficient.

2. Freundlich sorption isotherm in which sorbate concentration is related to solute concentration by:

$$C_s^{n+1} = x_1\left(C_w^{\frac{1}{x_2}}\right)^{n+1} = x_1\left(C_w^{\frac{1-x_2}{x_2}}\right)^{n} C_w^{n+1} = \chi\, C_w^{n+1} \qquad \textbf{4.28}$$

where n refers to the former time step, $\chi = x_1\left(C_w^{\frac{1-x_2}{x_2}}\right)^n$, and x_1 and x_2 are the Freundlich distribution coefficient and the Freundlich coefficient, respectively.

3. Langumir sorption isotherm, which, together with the linear isotherm, is one of the most common isotherms used in the literature today, and is defined by:

$$C_s^{n+1} = \chi_1 C_w^{n+1} \quad where\ \chi_1 = x_1\left(1 - \frac{x_2 C_w^n}{1 + x_2 C_w^n}\right) \qquad 4.29$$

where the empirical parameters x_1 and x_2 are the Langumir distribution coefficient and the Langumir coefficient, respectively.

NUTS permits sorption in the fractures and the matrix, but it has the user-controlled ability to bypass sorption in either flow field. Moreover, it is possible to assign any specified isotope as sorbable or nonsorbable, regardless of the flow regime.

Matrix/fracture boundary sorption and desorption are **not** included in the CCA calculation. However, sorption onto colloidal materials suspended within the flow is included, and is treated exactly as dissolution is treated, i.e., the effective maximum solubility coefficients are increased so as to represent the sum of all maximum allowable concentrations associated with (i) dissolution and (ii) all allowable forms of colloidal mobilization. See Section 4.5.1.

4.3.7 Decay

Radioisotopes are naturally unstable and spontaneously emit clusters of particles equivalent to helium atoms until, eventually, they arrive at stable configurations. Thus, the mass of daughter products is always lower than that of the parent radioisotope. Rates of disintegration vary over a wide range. Some are so slow that the parent radioisotopes are regarded as stable for all practical purposes. The mode of the observed disintegration can be classified into two categories: 1) disintegration associated with a change in atomic mass, such as alpha particle emission and spontaneous fission, and 2) disintegration without a change in atomic mass, such as the emission of positive or negative beta particles and the capture by the nucleus of an electron in the n=1 shell. Regardless of the mode of disintegration, the rate of the process is governed by a first-order rate law. Thus, the differential equation governing the mass of radioisotope residing at the top of a decay chain is of the form

$$\frac{\partial N}{\partial t} = -\lambda N, \qquad 4.30$$

M

where λ, the proportionality constant, is called the decay constant, and N is the number of atoms present at a given time, t . The integration of Equations like 4.30 leads to solutions of the form

$$N = N_0 e^{-\lambda t}, \qquad \textbf{4.31}$$

where N is the number of the atoms at any time, t , and N_0 is the number of the atoms at zero time. The decay constant is related to the half life by the following relationship:

$$\lambda = \frac{\ln 2}{t_{1/2}}, \qquad \textbf{4.32}$$

where $t_{1/2}$ is the half life. The decay constant and, therefore, the half life are fixed physical characteristics of each unstable radioisotope and are independent of its state of chemical combination, the presence of electric and magnetic fields, the temperature, and the pressure. Because there are Avogadro's number of atoms in each mole of a substance and a mass equivalent to the molecular weight in each mole, Equations 4.30 and 4.31 can be written in terms of mass M as:

$$\frac{\partial M}{\partial t} = -\lambda M, \qquad \textbf{4.33}$$

and

$$M = M_0 e^{-\lambda t}, \qquad \textbf{4.34}$$

M

where
M = mass of the solute at any time t (kg) , and

M_0 = mass of the solute at the initial time t_0 (kg) .

In porous media, the total mass of a radioisotope is distributed between the dissolved mass and the adsorbed mass. Therefore, from Equation 4.33, the rate at which mass is transported by decay, F_{decay}, in kg/s/m^3, can be written in terms of concentration as

$$F_{Decay} = -\lambda\left(\phi SC + (1-\phi)\rho_s C_s\right). \qquad \textbf{4.35}$$

Equation 4.35 represents a sink (i.e., loss of mass). If there is a parent(s) for the substance in question, the decay of the parent(s) will act as a source in the transport equation and have a value equal to the rate of growth of the daughter. The rate of growth due to parent(s) decay, F_g in kg/s/m^3 is

$$F_g = \sum_J \left(\phi SC + (1-\phi)\rho_s C_s\right)_{pj} \lambda_{pj}, \tag{4.36}$$

where J is the total number of parent substances and pj is a parent index.

4.3.8 The Overall Mass-Transport Equation

Amassing all the source/sink effects associated with sorption and radioactive decay (Equations 4.26, 4.35, and 4.36) within the continuity equation thus far derived (Equations 4.22 and 4.23), the overall mass-transport equation applicable to either hydrological regime is:

$$\begin{aligned}&\nabla.\phi SK\nabla C - \nabla.\mathrm{v}C + C^* q \pm \bar{\tau} C_{m/f} \pm \phi_m S_m K_{m/f}\left(C_f - C_m\right) = \\ &\frac{\partial}{\partial t}\left(\phi SC + (1-\phi)\rho_s C_s\right) + \left(\phi SC + (1-\phi)\rho_s C_s\right)\lambda - \\ &\sum_{j=1}^{J} \left(\phi SC + (1-\phi)\rho_s C_s\right)_{pj} \lambda_{pj} C_{pj}\end{aligned} \tag{4.37}$$

In fractured porous media, if the total porosity, ϕ_t is defined as $\phi_t = \phi_m + \phi_f$, where ϕ_m is the matrix porosity and ϕ_f is the fracture porosity, then the partial differential equations representing the transport in the two geophysical domains (fractured, and porous media) can be written as follows:

Fracture Equation:

$$\begin{aligned}&\nabla.\phi_{wf} S_{wf} K_{wf} \nabla C_{wf} - \nabla.\mathrm{v}_{wf} C_{wf} + C^*_{wf} q_{wf} - \bar{\tau} C_{wm/f} - \phi_{wm} S_{wm} K_{wm/f}\left(C_{wf} - C_{wm}\right) = \\ &\frac{\partial}{\partial t}\left(\phi_{wf} S_{wf} C_{wf} + (1-\phi_t)\rho_s C_{sf}\right) + \left(\phi_{wf} S_{wf} C_{wf} + (1-\phi_t)\rho_s C_{sf}\right)\lambda - \\ &\sum_{j=1}^{J} \left(\phi_{wf} S_{wf} C_{wf} + (1-\phi_t)\rho_s C_{sf}\right)_{pj} \lambda_{pj} C_{wfpj}\end{aligned} \tag{4.38}$$

Matrix Equation:

$$\begin{aligned}&\nabla.\phi_{wm} S_{wm} K_{wm} \nabla C_{wm} - \nabla.\mathrm{v}_{wm} C_{wm} + C^*_{wm} q_{wm} + \bar{\tau} C_{wm/f} + \phi_{wm} S_{wm} K_{wm/f}\left(C_{wf} - C_{wm}\right) = \\ &\frac{\partial}{\partial t}\left(\phi_{wm} S_{wm} C_{wm} + (1-\phi_t)\rho_s C_{sm}\right) + \left(\phi_{wm} S_{wm} C_{wm} + (1-\phi_t)\rho_s C_{sm}\right)\lambda - \\ &\sum_{j=1}^{J} \left(\phi_{wm} S_{wm} C_{wm} + (1-\phi_t)\rho_s C_{sm}\right)_{pj} \lambda_{pj} C_{wmpj}\end{aligned} \tag{4.39}$$

where *f* =fracture, *m*=matrix, *w* = brine, *s*=solid, *t*=total, *m/f* = fracture/matrix, and $C^* = C_{i,p}$

4.3.9 Initial and Boundary Conditions

The system of transport equations (fracture and matrix) given above is part of a boundary-value problem. Therefore, some sort of (i) initialization condition is required throughout the domain, and (ii) boundary conditions for all time are required for the dependent variable $C(x,y,z,t)$. The initial condition of the matrix equation in the three-dimensional domain (0,X; 0,Y; and 0,Z) is

$$C_{wm}(x,y,z,0) = f_m(x,y,z) \qquad 0 \le x \le X,\ 0 \le y \le Y,\ 0 \le z \le Z. \tag{4.40}$$

Similarly, the fracture is initialized by

$$C_{wf}(x,y,z,0) = f_f(x,y,z) \qquad 0 \le x \le X,\ 0 \le y \le Y,\ 0 \le z \le Z. \tag{4.41}$$

To assure that the system is in initial equilibrium, there must be no pressure or concentration gradients between the two continua, which leads to:

$$f_f(x,y,z) = f_m(x,y,z), \tag{4.42}$$

which must hold along their common boundary. The boundary conditions in the fracture and/or the matrix may be either Dirichlet or Neumann boundary conditions of the first kind. If they are Dirichlet conditions, the dependent variable itself is specified at the boundaries, as follows:

$$\begin{aligned}
C(0,y,z,t) &= g_{1x}(t) & t \ge 0,\\
C(X,y,z,t) &= g_{1X}(t) & t \ge 0,\\
C(x,0,z,t) &= g_{1y}(t) & t \ge 0,\\
C(x,Y,z,t) &= g_{1Y}(t) & t \ge 0,\\
C(x,y,0,t) &= g_{1z}(t) & t \ge 0,\\
C(x,y,Z,t) &= g_{1Z}(t) & t \ge 0.
\end{aligned} \tag{4.43}$$

If they are Neumann boundary conditions, the normal derivatives of the dependent variable are specified at the boundaries, as follows

$$
\begin{aligned}
&\frac{\partial}{\partial x}\left[C(0,y,z,t)\right] = g_{2x}(t) \qquad && t \geq 0, \\
&\frac{\partial}{\partial x}\left[C(X,y,z,t)\right] = g_{2X}(t) \qquad && t \geq 0, \\
&\frac{\partial}{\partial y}\left[C(x,0,z,t)\right] = g_{2y}(t) \qquad && t \geq 0, \\
&\frac{\partial}{\partial y}\left[C(x,Y,z,t)\right] = g_{2Y}(t) \qquad && t \geq 0, \\
&\frac{\partial}{\partial z}\left[C(x,y,0,t)\right] = g_{2z}(t) \qquad && t \geq 0, \\
&\frac{\partial}{\partial z}\left[C(x,y,Z,t)\right] = g_{2Z}(t) \qquad && t \geq 0.
\end{aligned}
\qquad 4.44
$$

These conditions are sometimes specified internally in the domain and called point source/sink.

In NUTS applications, Dirichlet boundary conditions can be used at both the outer boundaries or internally at sources. Neumann boundary conditions, on the other hand are used to specify no-transport at the outer boundaries of the simulation domain, i.e., $\frac{\partial C}{\partial x} = 0$, $\frac{\partial C}{\partial y} = 0$, and $\frac{\partial C}{\partial z} = 0$

4.4 Discretization Methods

The partial differential equations and boundary/initial conditions derived in Section 4.3 and representing transport in a dual-porosity, dual-permeability medium can be solved analytically for only the very simplest of problems, problems that bear virtually no relationship to real-world ground-water flows like those of the WIPP. To treat real-world problems, it is necessary to turn to numerical integration techniques, of which several are available.

The most common numerical methods used in modern hydrology are the finite-difference and finite-element (variational) methods, and particularly the Galerkin method. NUTS uses the finite-difference method to transform the time-space continuum problem into a solvable discrete mathematical model. An *algebraic* equation approximating the partial differential equation at each point of the domain of interest is derived. The derivational methods are described in this Section.

The first important step of the derivation is to define a convenient mesh (grid) over the entire spatial domain of interest.

For simplicity, consider the discrete representation of a continuous independent variable, x, between 0 and 1, as shown in Figure 4.4.

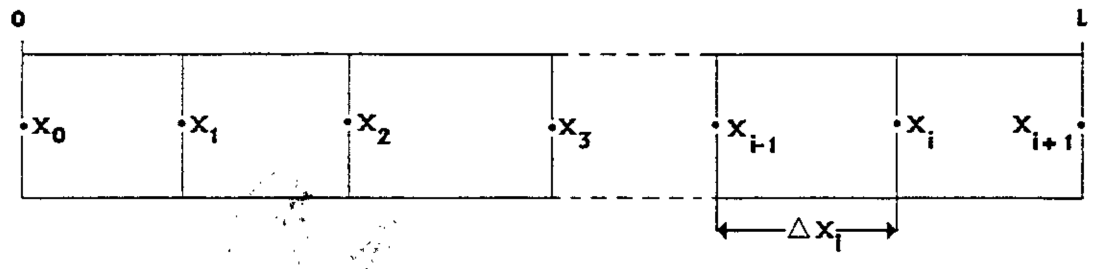

Figure 4.4 Point-Center Grid in Finite-Difference Approximation.

As is shown, the continuous domain is replaced by a set of discrete points whose spacing may be equal or unequal, and the discrete variable x is defined at those points. Clearly, the smaller the spacing the greater the resolution, but the more complex the resulting algebraic problem. Optimum spacings normally depend on the type of problem, the solution requirements, and the available resources. In Figure 4.4, the discrete values of x are denoted by x_i, $i = 0, 1, 2, ...$ The algebra is a little easier if the intervals are of equal width Δx, where Δx is called the finite difference.

$$x_i = i\Delta x . \tag{4.45}$$

In that case, neighboring points are related to one another by

$$x_{i+1} = x_i + \Delta x , \tag{4.46}$$

and Δx is the distance between successive grid points.

$$x_{i-1} = x_i - \Delta x . \tag{4.47}$$

The function C(x), which we think of as one of the unknown dependent variables of our problem, is normally defined on the continuous interval 0,1 . We will replace it by its finite-difference approximation C_i , which is defined only at the points $x = x_i$, i = 0, 1, ..., I+1 , and is represented by

$$C(x_i) = C_i . \tag{4.48}$$

Since our governing equations are partial differential equations, it is essential to have finite-difference expressions for the derivatives of an unknown function, as well as the function itself.

The derivative of the function C_i can be determined by Taylor's expansion C_i in the neighborhood of the point x. Thus for the function $C(x+\Delta x)$,

$$C(x+\Delta x)=C(x)+\frac{\partial C}{\partial x}\Delta x+\frac{\partial^2 C}{\partial x^2}\frac{\Delta x^2}{2!}+\frac{\partial^3 C}{\partial x^3}\frac{\Delta x^3}{3!}+\cdots+R_n, \qquad \textbf{4.49}$$

where the derivatives are evaluated at x and the remainder term R_n is given by

$$R_n=\frac{1}{n!}\left(\Delta x\frac{\partial}{\partial x}\right)^n C(x+\zeta\Delta x), \qquad \textbf{4.50}$$

and ζ lies between 0 and 1, so the argument of C lies between x and $x+\Delta x$. Equation 4.50 is sometimes written in the form

$$R_n=O\left[|\Delta x|^n\right], \qquad \textbf{4.51}$$

in which $|\Delta x|^n$ refers to the order of the truncation error in the Taylor series, and means there exists a positive constant M such that

$$|R_n|\le M\left(|\Delta x|^n\right) \text{ as } \Delta x\to 0. \qquad \textbf{4.52}$$

If we use a more compact notation, Equation 4.49 may be written as

$$C_{i+1}=C_i+\Delta x C_{ix}+\frac{(\Delta x)^2}{2!}C_{ixx}+\frac{(\Delta x)^3}{3!}C_{ixxx}+\cdots+R_n \qquad \textbf{4.53}$$

Similarly, the function $C(x-\Delta x)$ can be written as

$$C_{i-1}=C_i-\Delta x C_{ix}+\frac{(\Delta x)^2}{2!}C_{ixx}-\frac{(\Delta x)^3}{3!}C_{ixxx}+\cdots+R_n \qquad \textbf{4.54}$$

where $C_x=\frac{\partial C}{\partial x}, C_{xx}=\frac{\partial^2 C}{\partial x^2}$, etc., and all derivatives are evaluated at the point *i*. By adding and then subtracting Equations 4.53 and 4.54, one can arrive at finite-difference approximations for the first two derivatives of C_i, as follows:

$$C_{ix}=\frac{C_{i+1}-C_i}{\Delta x}+O(\Delta x), \qquad \textbf{4.55}$$

$$C_{ix} = \frac{C_i - C_{i-1}}{\Delta x} + O(\Delta x), \qquad 4.56$$

$$C_{ix} = \frac{C_{i+1} - C_{i-1}}{2\Delta x} + O\left[(\Delta x)^2\right], \qquad 4.57$$

$$C_{ixx} = \frac{C_{i+1} - 2C_i + C_{i-1}}{(\Delta x)^2} + O\left[(\Delta x)^2\right] \qquad 4.58$$

Equations 4.55, 4.56, 4.57 are known, respectively, as forward-, backward-, and central-difference formulae for the first derivative of C_i . Equation 4.58 gives a finite-difference approximation for the second derivative of C_i . All these formulae arise from truncated infinite series. Thus, truncation errors are involved. Truncation errors vary in these formulations, being first-order in the forward and backward difference formulae, and second-order in the central-difference and the second derivative.

In NUTS's discretization of the transport equation, a block-centered grid is used, which is consistent with BRAGFLO's discretization. With a block-centered grid, the location of the advective component of the transport is evaluated not at grid points, but at the interior points mid way between grid points. This formulation leads to introduction of new points at which the discretized variable is defined, namely, at $C_{i+1/2}$ and $C_{i-1/2}$, which is in addition to the discrete points i where the concentration C_i was defined earlier (see Figure 4.5)

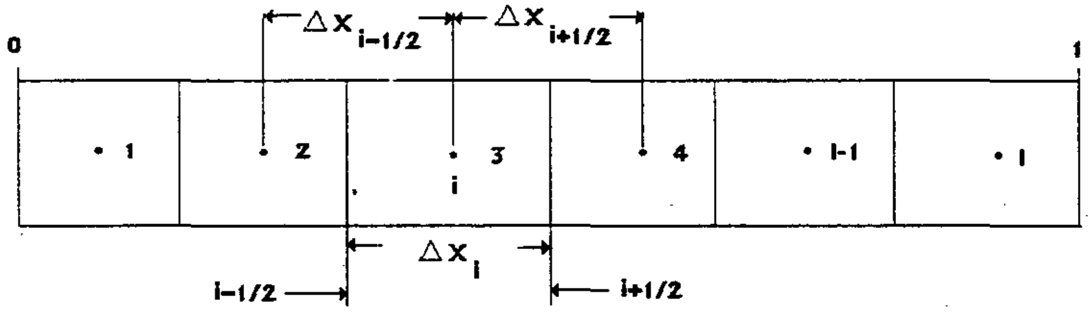

Figure 4.5 Block-Center Grid in Finite-Difference Approximation.

where $\Delta x_{i+\frac{1}{2}} = \frac{\Delta x_i + \Delta x_{i+1}}{2}$ is the distance between the neighboring points i and $i+1$. Various methods are used to approximate the value of the function at these interfacial points, each method having a characteristic accuracy. Among the most common methods are:

1. Mid-point weighting approximation. In this second-order accurate method, the concentration at the interface is approximated as follows:

$$C_{i-\frac{1}{2}} = (0.5+\mu_1)C_i + (0.5-\mu_1)C_{i-1}, \qquad \textbf{4.59}$$

and

$$C_{i+\frac{1}{2}} = (0.5+\mu_2)C_{i+1} + (0.5-\mu_2)C_i, \qquad \textbf{4.60}$$

where

$$\mu_1 = \frac{1}{4}\frac{\Delta x_{i-1} - \Delta x_i}{\Delta x_{i-1} + \Delta x_i}, \text{ and } \mu_2 = \frac{1}{4}\frac{\Delta x_i - \Delta x_{i+1}}{\Delta x_i + \Delta x_{i+1}}. \qquad \textbf{4.61}$$

From Equation 4.61, it is obvious that μ_1 and μ_2 are both zero for a uniform grid, wherein $C_{i+½}$ and $C_{i-½}$ become simple averages of neighboring values.

2. One-point upstream weighting. The interfacial concentration in this first-order accurate method is approximated by the concentration of the upstream block as follows:

$$C_{i-\frac{1}{2}} = \begin{Bmatrix} C_i & \textit{if the flow from i to i - 1} \\ C_{i-1} & \textit{if the flow from i - 1 to i} \end{Bmatrix}. \qquad \textbf{4.62}$$

Similarly

$$C_{i+\frac{1}{2}} = \begin{Bmatrix} C_i & \textit{if the flow from i to i + 1} \\ C_{i+1} & \textit{if the flow from i + 1 to i} \end{Bmatrix}. \qquad \textbf{4.63}$$

3. Two-point upstream weighting. This method is second-order accurate. The interfacial concentration in the two-point upstream weighting is approximated by the concentration of the two grid blocks upstream of the interface as follows:

$$C_{i-\frac{1}{2}} = \begin{Bmatrix} (1+\mu_1)C_i - \mu_1 C_{i+1} & \textit{if the flow from i to i - 1} \\ (1+\mu_2)C_{i-1} - \mu_2 C_{i-2} & \textit{if the flow from i - 1 to i} \\ \textit{where } \mu_1 = \dfrac{\Delta x_i}{\Delta x_i + \Delta x_{i+1}} \textit{ and } \mu_2 = \dfrac{\Delta x_{i-1}}{\Delta x_{i-1} + \Delta x_{i-2}} & \end{Bmatrix}, \qquad \textbf{4.64}$$

and

$$C_{i+\frac{1}{2}} = \begin{Bmatrix} (1+\mu_1)C_i \; - \mu_1 C_{i-1} & \textit{if the flow from } i \textit{ to } i+1 \\ (1+\mu_2)C_{i+1} \; - \mu_2 C_{i+2} & \textit{if the flow from } i+1 \textit{ to } i \\ \textit{where } \mu_1 = \dfrac{\Delta x_i}{\Delta x_i + \Delta x_{i-1}} \textit{ and } \mu_2 = \dfrac{\Delta x_{i+1}}{\Delta x_{i+1} + \Delta x_{i+2}} & \end{Bmatrix}. \qquad \textbf{4.65}$$

4. Leonard method. This third-order accurate method uses two grid blocks upstream and one grid block downstream. In this method and for a uniform grid, if the flow in Figure 4.5 is from the left to the right, then

$$C_{i-\frac{1}{2}} = \frac{2C_i + 5C_{i-1} - C_{i-2}}{6}, \qquad \textbf{4.66}$$

and

$$C_{i+\frac{1}{2}} = \frac{2C_{i+1} + 5C_i - C_{i-1}}{6}. \qquad \textbf{4.67}$$

5. Total-variation-diminishing (TVD) flux-limiter methods. In this class of methods, the flux, $F = vC$, and not the concentration is used as the independent variable. It starts with the average of F,

$$F_{i+\frac{1}{2}} = f_i + \frac{\Delta x}{2} \frac{f_{i+1} - f_i}{\Delta x} \qquad \textbf{4.68}$$

which is written as the first two terms of a Taylor expansion. Then, the slope-term contribution is multiplied by a flux-limiter function, φ, so that

$$F_{i+\frac{1}{2}} = f_i + \varphi(r_i) A_{i+\frac{1}{2}} \qquad \textbf{4.69}$$

where $A_{i+\frac{1}{2}}$ is the slope-term contribution and $\varphi(r)$, the flux-limiter term, is chosen in a way that gives a value of approximately 1.0 in smooth regions of the concentration profile. In regions where oscillations could occur due to the unrestrained slope-term contributions, $\varphi(r)$ is allowed to vary so as to eliminate spurious solutions. The variable r is defined as the ratio of successive slope terms, namely:

$$r_i = \frac{A_{i-\frac{1}{2}}}{A_{i+\frac{1}{2}}} \qquad \textbf{4.70}$$

Many satisfactory flux-limiter functions are available in the literature. A widely used example is the van Leer limiter, which is defined as:

$$\varphi(r_i) = \frac{|r_i| + r_i}{1 + |r_i|} \qquad \textbf{4.71}$$

The flux limiter should be calculated to be TVD, which means the total variation in the concentration is non increasing. If the total variation in the concentration at a certain time is defined by:

$$TV(C^n) = \sum (C_{i+1}^n - C_i^n) \qquad \textbf{4.72}$$

then in the TVD,

$$TV(C^{n+1}) \le TV(C^n) \qquad \textbf{4.73}$$

where *n+1* refers to the next time step. The formulation for $F_{i-\frac{1}{2}}$ is parallel. Using Equation 4.69, it is clear that TVD is a mid-point scheme, when $\varphi(r)=1$; a one-point upstream weighting scheme, when $\varphi(r)=0$; and a two-point upstream weighting scheme, when $\varphi(r)=r$. Other constraints apply to the flux-limiter function, depending on the degree of implicitness in the solution, but they are beyond the scope of this manual.

Thus far, the discretized values of the function and its spatial derivatives have been introduced. For time dependent problems, which arise in WIPP PAs, the dependent variables, generically denoted C_i are also functions of time $C_i = C_i(t)$. Therefore, at each grid point x_i, (i) a continuous or (ii) a discrete solution in time must also be calculated. If the time derivative is not discretized, the system of governing equations can, in principle, be solved by any of the methods traditionally used to solve ordinary differential equations. Euler's method, and the Runge-Kutta method are popular choices. However, the more common approach is to discretize temporal derivatives as well as spatial derivatives. In finite-difference formulations, the time derivative is approximated by one of the equations given above (Equation 4.55 to 4.58). A straightforward solution can be achieved by approximating the time derivative using the forward-difference method, as shown below:

$$\left(\frac{\partial C_i}{\partial t}\right)^n = \frac{C_i^{n+1} - C_i^n}{\Delta t}, \qquad \textbf{4.74}$$

where Δt refers to the time-step size, $n+1$ signifies the next time in the discrete sequence, and n is the present time. When the spatial distribution of the dependent variable C_i is known at the time step n, the system of equations can be solved explicitly. On the other hand, if the time derivative in Equation 4.74 is approximated using the backward-difference method and the spatial distribution of the dependent variable C_i is known at the time step $n+1$, a system of I equations must be solved. This method is called the implicit method. The implicit and explicit methods can also be combined so as to form a third integration method. Which method of solution to choose is not wholly arbitrary. Questions of stability, convergence, consistency, and the order of the method must be considered. Consequently, a versatile code must provide for several alternatives.

NUTS accommodates the first three weighting methods mentioned above, namely: one-point upstream, two-point upstream, and mid-point weighting. For temporal variabilities, NUTS's solution method can, in principle, vary from fully explicit to fully implicit. This spectrum of methods is cumbersome to describe. Therefore, for illustration purposes, an exemplary case will be shown in detail, but then details will be omitted from the discretized equations that follow. For the example, consider the hyperbolic, one-dimensional, purely-advective, partial differential equation given as Equation 4.75, in which the time derivative is discretized according to Equation 4.74:

$$(1-\theta)\left(-u\frac{\partial C}{\partial x}\right)^{n+1} + \theta\left(-u\frac{\partial C}{\partial x}\right)^n = \frac{C_i^{n+1} - C_i^n}{\Delta t}. \qquad \textbf{4.75}$$

In Equation 4.75: if $\theta = 1$, the temporal scheme is explicit; if $\theta = 0$, the temporal scheme is implicit; and if $\theta = 1/2$, the scheme is mixed, and this particular scheme is known as the Crank-Nicolson method.

4.4.1 Discrete Analogues of the Conservation of Mass Equations

Let us now consider the partial differential equations that govern transport in the fracture and the matrix of interest. We will discretize them in general and then develop the linear system for different numerical implementations. For the fracture, the discretized mass-transport equation in the brine is:

$$\begin{aligned}
&\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{i+\frac{1}{2}}\left(C_{wfi+1}^{n+1}-C_{wfi}^{n+1}\right)-\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{i-\frac{1}{2}}\left(C_{wfi}^{n+1}-C_{wfi-1}^{n+1}\right)+\\
&\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{j+\frac{1}{2}}\left(C_{wfj+1}^{n+1}-C_{wfj}^{n+1}\right)-\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{j-\frac{1}{2}}\left(C_{wfj}^{n+1}-C_{wfj-1}^{n+1}\right)+\\
&\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{k+\frac{1}{2}}\left(C_{wfk+1}^{n+1}-C_{wfk}^{n+1}\right)-\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{k-\frac{1}{2}}\left(C_{wfk}^{n+1}-C_{wfk-1}^{n+1}\right)+\\
&q_{wfi+\frac{1}{2}}^{n+1}C_{wfi+\frac{1}{2}}^{n+1}-q_{wfi-\frac{1}{2}}^{n+1}C_{wfi-\frac{1}{2}}^{n+1}+q_{wfj+\frac{1}{2}}^{n+1}C_{wfj+\frac{1}{2}}^{n+1}-q_{wfj-\frac{1}{2}}^{n+1}C_{wfj-\frac{1}{2}}^{n+1}+q_{wfk+\frac{1}{2}}^{n+1}C_{wfk+\frac{1}{2}}^{n+1}-q_{wfk-\frac{1}{2}}^{n+1}C_{wfk-\frac{1}{2}}^{n+1}\\
&+C_{wfi}^{*n+1}Q_{wfi}^{n+1}-\tau_{wm/fi}^{n+1}\left(\omega_{m/fi}C_{wfi}^{n+1}+\left(1-\omega_{m/fi}\right)C_{wmi}^{n+1}\right)-\\
&\left(\phi_{mi}^{n+1}S_{wmi}^{n+1}K_{wm/fi}^{n+1}\right)\left(C_{wfi}^{n+1}-C_{wmi}^{n+1}\right)=\\
&\frac{V_{Ri}}{\Delta t}\left[\left\{\phi_{fi}S_{wfi}C_{wfi}+\left(1-\phi_t\right)\rho_sC_{sfi}\right\}^{n+1}-\left\{\phi_{fi}S_{wfi}C_{wfi}+\left(1-\phi_t\right)\rho_sC_{sfi}\right\}^{n}\right]+\\
&V_{Ri}\left\{\phi_{fi}S_{wfi}C_{wfi}+\left(1-\phi_t\right)\rho_sC_{sfi}\right\}^{n+1}\lambda-V_{Ri}\sum_{l=1}^{L}\left[\left\{\phi_{fi}S_{wfi}C_{wfli}+\left(1-\phi_t\right)\rho_sC_{sfli}\right\}\lambda_l\right]^{n+1}.
\end{aligned} \tag{4.76}$$

Similarly, for the matrix continuum, the discretized mass-transport equation in the brine is:

$$\begin{aligned}
&\left(\phi_m^{n+1}S_{wm}^{n+1}K_{wm}^{n+1}\right)_{i+\frac{1}{2}}\left(C_{wmi+1}^{n+1}-C_{wmi}^{n+1}\right)-\left(\phi_m^{n+1}S_{wf}^{n+1}K_{wm}^{n+1}\right)_{i-\frac{1}{2}}\left(C_{wmi}^{n+1}-C_{wmi-1}^{n+1}\right)+\\
&\left(\phi_m^{n+1}S_{wm}^{n+1}K_{wm}^{n+1}\right)_{j+\frac{1}{2}}\left(C_{wmj+1}^{n+1}-C_{wmj}^{n+1}\right)-\left(\phi_m^{n+1}S_{wm}^{n+1}K_{wm}^{n+1}\right)_{j-\frac{1}{2}}\left(C_{wmj}^{n+1}-C_{wmj-1}^{n+1}\right)+\\
&\left(\phi_m^{n+1}S_{wm}^{n+1}K_{wm}^{n+1}\right)_{k+\frac{1}{2}}\left(C_{wmk+1}^{n+1}-C_{wmk}^{n+1}\right)-\left(\phi_m^{n+1}S_{wm}^{n+1}K_{wm}^{n+1}\right)_{k-\frac{1}{2}}\left(C_{wmk}^{n+1}-C_{wmk-1}^{n+1}\right)+\\
&q_{wmi+\frac{1}{2}}^{n+1}C_{wmi+\frac{1}{2}}^{n+1}-q_{wmi-\frac{1}{2}}^{n+1}C_{wmi-\frac{1}{2}}^{n+1}+q_{wmj+\frac{1}{2}}^{n+1}C_{wmj+\frac{1}{2}}^{n+1}-q_{wmj-\frac{1}{2}}^{n+1}C_{wmj-\frac{1}{2}}^{n+1}+q_{wmk+\frac{1}{2}}^{n+1}C_{wmk+\frac{1}{2}}^{n+1}-q_{wmk-\frac{1}{2}}^{n+1}C_{wmk-\frac{1}{2}}^{n+1}\\
&+C_{wmi}^{*n+1}Q_{wmi}^{n+1}+\tau_{wm/fi}^{n+1}\left(\omega_{m/fi}C_{wfi}^{n+1}+\left(1-\omega_{m/fi}\right)C_{wmi}^{n+1}\right)+\left(\phi_{mi}^{n+1}S_{wmi}^{n+1}K_{wm/fi}^{n+1}\right)\left(C_{wfi}^{n+1}-C_{wmi}^{n+1}\right)=\\
&\frac{V_{Ri}}{\Delta t}\left[\left\{\phi_{mi}S_{wmi}C_{wmi}+\left(1-\phi_t\right)\rho_sC_{smi}\right\}^{n+1}-\left\{\phi_{mi}S_{wmi}C_{wmi}+\left(1-\phi_t\right)\rho_sC_{smi}\right\}^{n}\right]+\\
&V_{Ri}\left\{\phi_{mi}S_{wmi}C_{wmi}+\left(1-\phi_t\right)\rho_sC_{smi}\right\}^{n+1}\lambda-V_{Ri}\sum_{l=1}^{L}\left[\left\{\phi_{mi}S_{wmi}C_{wmli}+\left(1-\phi_t\right)\rho_sC_{smli}\right\}\lambda_l\right]^{n+1}
\end{aligned} \tag{4.77}$$

4.4.2 One-Point Upstream-Winding Discretization of Transport Equations

Fracture Equation

$$\begin{aligned}
&\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{i+\frac{1}{2}}\left(C_{wfi+1}^{n+1}-C_{wfi}^{n+1}\right)-\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{i-\frac{1}{2}}\left(C_{wfi}^{n+1}-C_{wfi-1}^{n+1}\right)+\\
&\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{j+\frac{1}{2}}\left(C_{wfj+1}^{n+1}-C_{wfj}^{n+1}\right)-\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{j-\frac{1}{2}}\left(C_{wfj}^{n+1}-C_{wfj-1}^{n+1}\right)+
\end{aligned}$$

$$\begin{aligned}
&\left(\phi_f^{n+1} S_{wf}^{n+1} K_{wf}^{n+1}\right)_{k+\frac{1}{2}}\left(C_{wfk+1}^{n+1} - C_{wfk}^{n+1}\right) - \left(\phi_f^{n+1} S_{wf}^{n+1} K_{wf}^{n+1}\right)_{k-\frac{1}{2}}\left(C_{wfk}^{n+1} - C_{wfk-1}^{n+1}\right) + \\
&q_{wfi+\frac{1}{2}}^{n+1}\left(\omega_{fi+1} C_{wfi}^{n+1} + \left(1-\omega_{fi+1}\right)C_{wfi+1}^{n+1}\right) - q_{wfi-\frac{1}{2}}^{n+1}\left(\omega_{fi} C_{wfi-1}^{n+1} + \left(1-\omega_{fi}\right)C_{wfi}^{n+1}\right) + \\
&q_{wfj+\frac{1}{2}}^{n+1}\left(\omega_{fj+1} C_{wfj}^{n+1} + \left(1-\omega_{fj+1}\right)C_{wfj+1}^{n+1}\right) - q_{wfj-\frac{1}{2}}^{n+1}\left(\omega_{fj} C_{wfj-1}^{n+1} + \left(1-\omega_{fj}\right)C_{wfj}^{n+1}\right) + \\
&q_{wfk+\frac{1}{2}}^{n+1}\left(\omega_{fk+1} C_{wfk}^{n+1} + \left(1-\omega_{fk+1}\right)C_{wfk+1}^{n+1}\right) - q_{wfk-\frac{1}{2}}^{n+1}\left(\omega_{fk} C_{wfk-1}^{n+1} + \left(1-\omega_{fk}\right)C_{wfk}^{n+1}\right) + \\
&C_{wfi}^{*n+1} Q_{wfi}^{n+1} - \tau_{wm/fi}^{n+1}\left(\omega_{m/fi} C_{wfi}^{n+1} + \left(1-\omega_{m/fi}\right)C_{wmi}^{n+1}\right) - \left(\phi_{mi}^{n+1} S_{wmi}^{n+1} K_{wm/fi}^{n+1}\right)\left(C_{wfi}^{n+1} - C_{wmi}^{n+1}\right) = \\
&\frac{V_{Ri}}{\Delta t}\left\{\phi_{fi} S_{wfi} C_{wfi} + \left(1-\phi_t\right)\rho_s C_{sfi}\right\}^{n+1} - \frac{V_{Ri}}{\Delta t}\left\{\phi_{fi} S_{wfi} C_{wfi} + \left(1-\phi_t\right)\rho_s C_{sfi}\right\}^{n} + \\
&V_{Ri}\left\{\phi_{fi} S_{wfi} C_{wfi} + \left(1-\phi_t\right)\rho_s C_{sfi}\right\}^{n+1}\lambda - V_{Ri}\sum_{l=1}^{L}\left\{\left\{\phi_{fi} S_{wfi} C_{wfi} + \left(1-\phi_t\right)\rho_s C_{sfli}\right\}\lambda_l\right\}^{n+1}.
\end{aligned} \qquad \textbf{4.78}$$

By collecting similar terms, the above equation can be represented by the following linear system,

$$\begin{aligned}
&A_f C_{wfk-1}^{n+1} + B_f C_{wfj-1}^{n+1} + C_f C_{wfi-1}^{n+1} + D_{1f} C_{wfijk}^{n+1} + D_{2f} C_{wmijk}^{n+1} + \\
&E_f C_{wfi+1}^{n+1} + F_f C_{wfj+1}^{n+1} + G_f C_{wk+1}^{n+1} = R_f^{n+1}
\end{aligned} \quad , \qquad \textbf{4.79}$$

where

$$\begin{aligned}
A_f &= HD_{wfk-\frac{1}{2}}^{n+1} - \omega_{fk} q_{wfk-\frac{1}{2}}^{n+1}, \\
B_f &= HD_{wfj-\frac{1}{2}}^{n+1} - \omega_{fj} q_{wfj-\frac{1}{2}}^{n+1}, \\
C_f &= HD_{wfi-\frac{1}{2}}^{n+1} - \omega_{fi} q_{wfi-\frac{1}{2}}^{n+1}, \\
E_f &= HD_{wfi+\frac{1}{2}}^{n+1} - \left(1-\omega_{fi+1}\right) q_{wfi+\frac{1}{2}}^{n+1}, \\
F_f &= HD_{wfj+\frac{1}{2}}^{n+1} - \left(1-\omega_{fj+1}\right) q_{wfj+\frac{1}{2}}^{n+1}, \\
G_f &= HD_{wfk+\frac{1}{2}}^{n+1} - \left(1-\omega_{fk+1}\right) q_{wfk+\frac{1}{2}}^{n+1}, \\
D_{1f} &= -HD_{wfk-\frac{1}{2}}^{n+1} - HD_{wfj-\frac{1}{2}}^{n+1} - HD_{wfi-\frac{1}{2}}^{n+1} - HD_{wfk+\frac{1}{2}}^{n+1} - HD_{wfj+\frac{1}{2}}^{n+1} - HD_{wfi+\frac{1}{2}}^{n+1} - \\
&\left(1-\omega_{fk}\right) q_{wfk-\frac{1}{2}}^{n+1} - \left(1-\omega_{fj}\right) q_{wfj-\frac{1}{2}}^{n+1} - \left(1-\omega_{fi}\right) q_{wfi-\frac{1}{2}}^{n+1} + \omega_{fk+1} q_{wfk+\frac{1}{2}}^{n+1} + \omega_{fj+1} q_{wfj+\frac{1}{2}}^{n+1} + \omega_{fi+1} q_{wfi+\frac{1}{2}}^{n+1} - \\
&\frac{V_{Ri}}{\Delta t}\left\{\phi_{fi} S_{wfi} + \left(1-\phi_t\right)\rho_s \xi_{fi}\right\}^{n+1} + V_{Ri}\left\{\phi_{fi} S_{wfi} + \left(1-\phi_t\right)\rho_s \xi_{fi}\right\}^{n+1}\lambda - \tau_{wm/fi}^{n+1}\omega_{m/fi} - \phi_m^{n+1} S_{wm}^{n+1} K_{wm/fi}^{n+1},
\end{aligned}$$

$$D_{2f}^{n+1} = -\tau_{wm/fi}^{n+1}\left(1 - \omega_{m/fi}\right) + \phi_m^{n+1} S_{wm}^{n+1} K_{wm/fi}^{n+1},$$

and

$$R_f^{n+1} = \frac{V_{Ri}}{\Delta t}\left\{\phi_{fi} S_{wfi} C_{wfi} + (1-\phi_t)\rho_s \xi_{fi} C_{wfi}\right\}^n - C_{wfi}^* Q_{wfi}^{n+1} - V_{Ri}\sum_{i=1}^{L}\left\{\left\{\phi_{fi} S_{wfi} C_{wfli} + (1-\phi_t)\rho_s \xi_{fli} C_{wfli}\right\}\lambda_l\right\}^{n+1}.$$

In the above formulations HD is the hydrodynamic dispersion and is defined at every fluid-solid interface. The convention adopted in NUTS for calculating the HD is that the values of the porosity, saturation, dispersivities, and tortousity are weighted upstream. Therefore, $HD_{wfi-1/2}$ is defined as follows:

$$HD_{wfi-\frac{1}{2}}^{\;n+1} = \omega_{fi}\phi_{fi} S_{wfi} + \left(1-\omega_{fi}\right)\phi_{fi-1} S_{wfi-1}\frac{\Delta y_i \Delta z_i}{\frac{\Delta x_i + \Delta x_{i-1}}{2}}\left\{\frac{D_m^*}{\tau_f^*} + \frac{1}{\omega_{fi}\phi_{fi} S_{wfi} + \left(1-\omega_{fi}\right)\phi_{fi-1} S_{wfi-1}}\frac{1}{\sqrt{v_{xf}^2 + v_{yf}^2 + v_{xf}^2}}\left(\alpha_{Lf} v_{xf}^2 + \alpha_{Tf}\left(v_{yf}^2 + v_{zf}^2\right)\right)\right\}$$

Similar equations can be written for the HD of the other interfaces. The values of ζ used in R_f and D_{1f} are sorption coefficients and depend mainly on the isotherm selected for use. For example, in the linear adsorption isotherm ζ is equivalent to k_d. In the above formulations, ω refers to the upstream weighting and has values of 0 and 1. The value of ω is calculated depending on the sign of the velocity. For two adjacent blocks i-1, and i the value of ω_i is:

$$\omega_i = \begin{Bmatrix} 1 & \textit{if the flow from } i-1, j, k \textit{ to } i, j, k \\ 0 & \textit{otherwise} \end{Bmatrix}$$

similarly, the weighting parameters in the other direction are

$$\omega_j = \begin{Bmatrix} 1 & \textit{if the flow from } i, j-1, k \textit{ to } i, j, k \\ 0 & \textit{otherwise} \end{Bmatrix}$$

and

$$\omega_k = \begin{Bmatrix} 1 & \textit{if the flow from } i, j, k-1 \textit{ to } i, j, k \\ 0 & \textit{otherwise} \end{Bmatrix}$$

Similar convention is used in calculating the matrix/fracture weighting factor which is:

$$\omega_{m/f} = \begin{Bmatrix} 1 & \textit{if flow from the fracture to the matrix} \\ 0 & \textit{otherwise} \end{Bmatrix}$$

Matrix Equation:

$$\begin{aligned}
&\left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{i+\frac{1}{2}}\left(C_{wmi+1}^{n+1} - C_{wmi}^{n+1}\right) - \left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{i-\frac{1}{2}}\left(C_{wmi}^{n+1} - C_{wmi-1}^{n+1}\right) + \\
&\left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{j+\frac{1}{2}}\left(C_{wmj+1}^{n+1} - C_{wmj}^{n+1}\right) - \left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{j-\frac{1}{2}}\left(C_{wmj}^{n+1} - C_{wmj-1}^{n+1}\right) + \\
&\left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{k+\frac{1}{2}}\left(C_{wmk+1}^{n+1} - C_{wmk}^{n+1}\right) - \left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{k-\frac{1}{2}}\left(C_{wmk}^{n+1} - C_{wmk-1}^{n+1}\right) + \\
&q_{wmi+\frac{1}{2}}^{n+1}\left(\omega_{mi+1} C_{wmi}^{n+1} + \left(1-\omega_{mi+1}\right) C_{wmi+1}^{n+1}\right) - q_{wmi-\frac{1}{2}}^{n+1}\left(\omega_{mi} C_{wmi-1}^{n+1} + \left(1-\omega_{mi}\right) C_{wmi}^{n+1}\right) + \\
&q_{wmj+\frac{1}{2}}^{n+1}\left(\omega_{mj+1} C_{wmj}^{n+1} + \left(1-\omega_{mj+1}\right) C_{wmj+1}^{n+1}\right) - q_{wmj-\frac{1}{2}}^{n+1}\left(\omega_{mj} C_{wmj-1}^{n+1} + \left(1-\omega_{mj}\right) C_{wmj}^{n+1}\right) + \\
&q_{wmk+\frac{1}{2}}^{n+1}\left(\omega_{mk+1} C_{wmk}^{n+1} + \left(1-\omega_{mk+1}\right) C_{wmk+1}^{n+1}\right) - q_{wmk-\frac{1}{2}}^{n+1}\left(\omega_{mk} C_{wmk-1}^{n+1} + \left(1-\omega_{mk}\right) C_{wmk}^{n+1}\right) + \\
&C_{wmi}^{*n+1} Q_{wmi}^{n+1} + \tau_{wm/fi}^{n+1}\left(\omega_{m/fi} C_{wfi}^{n+1} + \left(1-\omega_{m/fi}\right) C_{wmi}^{n+1}\right) + \left(\phi_{mi}^{n+1} S_{wmi}^{n+1} K_{wm/fi}^{n+1}\right)\left(C_{wfi}^{n+1} - C_{wmi}^{n+1}\right) = \\
&\frac{V_{Ri}}{\Delta t}\left\{\phi_{mi} S_{wmi} C_{wmi} + \left(1-\phi_t\right)\rho_s C_{smi}\right\}^{n+1} - \frac{V_{Ri}}{\Delta t}\left\{\phi_{mi} S_{wmi} C_{wmi} + \left(1-\phi_t\right)\rho_s C_{smi}\right\}^{n} + \\
&V_{Ri}\left\{\phi_{mi} S_{wmi} C_{wmi} + \left(1-\phi_t\right)\rho_s C_{smi}\right\}^{n+1}\lambda - V_{Ri}\sum_{l=1}^{L}\left\{\left\{\phi_{mi} S_{wmi} C_{wmi} + \left(1-\phi_t\right)\rho_s C_{smli}\right\}\lambda_l\right\}^{n+1}.
\end{aligned} \qquad \textbf{4.80}$$

Corresponding to the fracture-flow equation, the linear system is:

$$\begin{aligned}
&A_m C_{wmk-1}^{n+1} + B_m C_{wmj-1}^{n+1} + C_m C_{wmi-1}^{n+1} + D_{1m} C_{wmijk}^{n+1} + D_{2m} C_{wfijk}^{n+1} + \\
&E_m C_{wmi+1}^{n+1} + F_m C_{wmj+1}^{n+1} + G_m C_{wmk+1}^{n+1} = R_m^{n+1}
\end{aligned} \qquad \textbf{4.81}$$

where

$$\begin{aligned}
A_m &= HD_{wmk-\frac{1}{2}}^{n+1} - \omega_{mk} q_{wmk-\frac{1}{2}}^{n+1}, \\
B_m &= HD_{wmj-\frac{1}{2}}^{n+1} - \omega_{mj} q_{wmj-\frac{1}{2}}^{n+1}, \\
C_m &= HD_{wmi-\frac{1}{2}}^{n+1} - \omega_{wi} q_{wmi-\frac{1}{2}}^{n+1}, \\
E_m &= HD_{wmi+\frac{1}{2}}^{n+1} + \left(1-\omega_{mi+1}\right) q_{wmi+\frac{1}{2}}^{n+1},
\end{aligned}$$

$$F_m = HD^{n+1}_{wmj+\frac{1}{2}} + \left(1-\omega_{mj+1}\right)q^{n+1}_{wmj+\frac{1}{2}}$$

$$G_m = HD^{n+1}_{wmk+\frac{1}{2}} + \left(1-\omega_{mk+1}\right)q^{n+1}_{wmk+\frac{1}{2}},$$

$$D^{n+1}_{1m} = -HD^{n+1}_{wmk-\frac{1}{2}} - HD^{n+1}_{wmj-\frac{1}{2}} - HD^{n+1}_{wmi-\frac{1}{2}} - HD^{n+1}_{wmk+\frac{1}{2}} - HD^{n+1}_{wmj+\frac{1}{2}} - HD^{n+1}_{wmi+\frac{1}{2}}$$
$$\left(1-\omega_{mk}\right)q^{n+1}_{wmk-\frac{1}{2}} - \left(1-\omega_{mj}\right)q^{n+1}_{wmj-\frac{1}{2}} - \left(1-\omega_{mi}\right)q^{n+1}_{wmi-\frac{1}{2}} + \omega_{mk+1}q^{n+1}_{wmk+\frac{1}{2}} + \omega_{mj+1}q^{n+1}_{wmj+\frac{1}{2}} + \omega_{mi+1}q^{n+1}_{wmi+\frac{1}{2}} -$$
$$\frac{V_{Ri}}{\Delta t}\left\{\phi_{mi}S_{wmi} + \left(1-\phi_t\right)\rho_s\xi_{mi}\right\}^{n+1} + V_{Ri}\left\{\phi_{mi}S_{wmi} + \left(1-\phi_t\right)\rho_s\xi_{mi}\right\}^{n+1}\lambda + \tau^{n+1}_{wm/fi}\left(1-\omega_{m/fi}\right) - \phi^{n+1}_m S^{n+1}_{wm} K^{n+1}_{wm/fi},$$

$$D^{n+1}_{2m} = \tau^{n+1}_{wm/fi}\,\omega_{m/fi} + \phi^{n+1}_m S^{n+1}_{wm} K^{n+1}_{wm/fi},$$

and

$$R^{n+1}_m = \frac{V_{Ri}}{\Delta t}\left\{\phi_{mi}S_{wmi}C_{wmi} + \left(1-\phi_t\right)\rho_s\xi_{mi}C_{wmi}\right\}^n - C^*_{wmi}Q^{n+1}_{wmi} -$$
$$V_{Ri}\sum_{l=1}^{L}\left\{\left\{\phi_{mi}S_{wmi}C_{wmli} + \left(1-\phi_t\right)\rho_s\xi_{mli}C_{wmli}\right\}\lambda_l\right\}^{n+1}.$$

In a similar fashion, the HD in the matrix at I-1/2 interface is defined as follows:

$$HD^{n+1}_{wmi-\frac{1}{2}} = \omega_{mi}\phi_{mi}S_{wmi} + \left(1-\omega_{mi}\right)\phi_{mi-1}S_{wmi-1}\frac{\Delta y_i \Delta z_i}{\frac{\Delta x_i + \Delta x_{i-1}}{2}}\left\{\frac{D^*_m}{\tau^*_m} + \right.$$
$$\left.\frac{1}{\omega_{mi}\phi_{mi}S_{wmi} + \left(1-\omega_{mi}\right)\phi_{mi-1}S_{wmi-1}}\frac{1}{\sqrt{v^2_{xm} + v^2_{ym} + v^2_{zm}}}\left(\alpha_{Lm}v^2_{xm} + \alpha_{Tm}\left(v^2_{ym} + {}^2_{zm}\right)\right)\right\}$$

Similar definitions for ξ and ω's will be extended to the matrix discretization.

4.4.3 Two-Point Upstream-Winding Discretization of Transport Equations

In this method, two approaches for discretization are used. In the first approach, the equation is discretized fully implicitly as follows:

Fracture Equation

$$\left(\phi^{n+1}_f S^{n+1}_{wf} K^{n+1}_{wf}\right)_{i+\frac{1}{2}}\left(C^{n+1}_{wfi+1} - C^{n+1}_{wfi}\right) - \left(\phi^{n+1}_f S^{n+1}_{wf} K^{n+1}_{wf}\right)_{i-\frac{1}{2}}\left(C^{n+1}_{wfi} - C^{n+1}_{wfi-1}\right) +$$

$$
\begin{aligned}
&\left(\phi_f^{n+1} S_{wf}^{n+1} K_{wf}^{n+1}\right)_{j+\frac{1}{2}}\left(C_{wfj+1}^{n+1} - C_{wfj}^{n+1}\right) - \left(\phi_f^{n+1} S_{wf}^{n+1} K_{wf}^{n+1}\right)_{j-\frac{1}{2}}\left(C_{wfj}^{n+1} - C_{wfj-1}^{n+1}\right) + \\
&\left(\phi_f^{n+1} S_{wf}^{n+1} K_{wf}^{n+1}\right)_{k+\frac{1}{2}}\left(C_{wfk+1}^{n+1} - C_{wfk}^{n+1}\right) - \left(\phi_f^{n+1} S_{wf}^{n+1} K_{wf}^{n+1}\right)_{k-\frac{1}{2}}\left(C_{wfk}^{n+1} - C_{wfk-1}^{n+1}\right) + \\
&q_{wfi+\frac{1}{2}}^{n+1}\left[\omega_{fi+1}\left\{(1+\mu_{i1})C_{wfi}^{n+1} - \mu_{i1}C_{wfi-1}^{n+1}\right\} + (1-\omega_{fi+1})\left\{(1+\mu_{i2})C_{wfi+1}^{n+1} - \mu_{i2}C_{wfi+2}^{n+1}\right\}\right] - \\
&q_{wfi-\frac{1}{2}}^{n+1}\left[\omega_{i}\left\{(1+\mu_{i3})C_{wfi-1}^{n+1} - \mu_{i3}C_{wfi-2}^{n+1}\right\} + (1-\omega_{i})\left\{(1+\mu_{i4})C_{wfi}^{n+1} - \mu_{i4}C_{wfi+1}^{n+1}\right\}\right] + \\
&q_{wfj+\frac{1}{2}}^{n+1}\left[\omega_{fj+1}\left\{(1+\mu_{j1})C_{wfj}^{n+1} - \mu_{j1}C_{wfj-1}^{n+1}\right\} + (1-\omega_{fj+1})\left\{(1+\mu_{j2})C_{wfj+1}^{n+1} - \mu_{j2}C_{wfj+2}^{n+1}\right\}\right] - \\
&q_{wfj-\frac{1}{2}}^{n+1}\left[\omega_{fj}\left\{(1+\mu_{j3})C_{wfj-1}^{n+1} - \mu_{j3}C_{wfj-2}^{n+1}\right\} + (1-\omega_{fj})\left\{(1+\mu_{j4})C_{wfj}^{n+1} - \mu_{j4}C_{wfj+1}^{n+1}\right\}\right] + \\
&q_{wfk+\frac{1}{2}}^{n+1}\left[\omega_{fk+1}\left\{(1+\mu_{k1})C_{wfk}^{n+1} - \mu_{k1}C_{wfk-1}^{n+1}\right\} + (1-\omega_{fk+1})\left\{(1+\mu_{k2})C_{wfk+1}^{n+1} - \mu_{k2}C_{wfk+2}^{n+1}\right\}\right] - \\
&q_{wfk-\frac{1}{2}}^{n+1}\left[\omega_{fk}\left\{(1+\mu_{k3})C_{wfk-1}^{n+1} - \mu_{k3}C_{wfk-2}^{n+1}\right\} + (1-\omega_{fk})\left\{(1+\mu_{k4})C_{wfk}^{n+1} - \mu_{k4}C_{wfk+1}^{n+1}\right\}\right] + \\
&C_{wfi}^{*n+1} Q_{wfi}^{n+1} - \tau_{wm/fi}^{n+1}\left(\omega_{m/fi}C_{wfi}^{n+1} + (1-\omega_{m/fi})C_{wmi}^{n+1}\right) - \left(\phi_{mi}^{n+1} S_{wmi}^{n+1} K_{wm/fi}^{n+1}\right)\left(C_{wfi}^{n+1} - C_{wmi}^{n+1}\right) = \\
&\frac{V_{Ri}}{\Delta t}\left[\left\{\phi_{fi} S_{wfi} C_{wfi} + (1-\phi_t)\rho_s C_{sfi}\right\}^{n+1} - \left\{\phi_{fi} S_{wfi} C_{wfi} + (1-\phi_t)\rho_s C_{sfi}\right\}^{n}\right] + \\
&V_{Ri}\left\{\phi_{fi} S_{wfi} C_{wfi} + (1-\phi_t)\rho_s C_{sfi}\right\}^{n+1}\lambda - V_{Ri}\sum_{l=1}^{L}\left[\left\{\phi_{fi} S_{wfi} C_{wfli} + (1-\phi_t)\rho_s C_{sfli}\right\}\lambda_l\right]^{n+1}.
\end{aligned}
\tag{4.82}
$$

Matrix Equation:

$$
\begin{aligned}
&\left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{i+\frac{1}{2}}\left(C_{wmi+1}^{n+1} - C_{wmi}^{n+1}\right) - \left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{i-\frac{1}{2}}\left(C_{wmi}^{n+1} - C_{wmi-1}^{n+1}\right) + \\
&\left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{j+\frac{1}{2}}\left(C_{wmj+1}^{n+1} - C_{wmj}^{n+1}\right) - \left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{j-\frac{1}{2}}\left(C_{wmj}^{n+1} - C_{wmj-1}^{n+1}\right) + \\
&\left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{k+\frac{1}{2}}\left(C_{wmk+1}^{n+1} - C_{wmk}^{n+1}\right) - \left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{k-\frac{1}{2}}\left(C_{wmk}^{n+1} - C_{wmk-1}^{n+1}\right) +
\end{aligned}
$$

$$q^{n+1}_{wmi+\frac{1}{2}}\left[\omega_{mi+1}\left\{(1+\mu_{i1})C^{n+1}_{wmi}-\mu_{i1}C^{n+1}_{wmi-1}\right\}+(1-\omega_{mi+1})\left\{(1+\mu_{i2})C^{n+1}_{wmi+1}-\mu_{i2}C^{n+1}_{wmi+2}\right\}\right]-$$

$$q^{n+1}_{wmi-\frac{1}{2}}\left[\omega_{mi}\left\{(1+\mu_{i3})C^{n+1}_{wmi-1}-\mu_{i3}C^{n+1}_{wmi-2}\right\}+(1-\omega_{mi})\left\{(1+\mu_{i4})C^{n+1}_{wmi}-\mu_{i4}C^{n+1}_{wmi+1}\right\}\right]+$$

$$q^{n+1}_{wmj+\frac{1}{2}}\left[\omega_{mj+1}\left\{(1+\mu_{j1})C^{n+1}_{wmj}-\mu_{j1}C^{n+1}_{wmj-1}\right\}+(1-\omega_{mj+1})\left\{(1+\mu_{j2})C^{n+1}_{wmj+1}-\mu_{j2}C^{n+1}_{wmj+2}\right\}\right]- \qquad \mathbf{4.83}$$

$$q^{n+1}_{wmj-\frac{1}{2}}\left[\omega_{mj}\left\{(1+\mu_{j3})C^{n+1}_{wmj-1}-\mu_{j3}C^{n+1}_{wmj-2}\right\}+(1-\omega_{mj})\left\{(1+\mu_{j4})C^{n+1}_{wmj}-\mu_{j4}C^{n+1}_{wmj+1}\right\}\right]+$$

$$q^{n+1}_{wmk+\frac{1}{2}}\left[\omega_{mk+1}\left\{(1+\mu_{k1})C^{n+1}_{wmk}-\mu_{k1}C^{n+1}_{wmk-1}\right\}+(1-\omega_{mk+1})\left\{(1+\mu_{k2})C^{n+1}_{wmk+1}-\mu_{k2}C^{n+1}_{wmk+2}\right\}\right]-$$

$$q^{n+1}_{wmk-\frac{1}{2}}\left[\omega_{mk}\left\{(1+\mu_{k3})C^{n+1}_{wmk-1}-\mu_{k3}C^{n+1}_{wmk-2}\right\}+(1-\omega_{mk})\left\{(1+\mu_{k4})C^{n+1}_{wmk}-\mu_{k4}C^{n+1}_{wmk+1}\right\}\right]+$$

$$C^{*n+1}_{wmi}Q^{n+1}_{wmi}+\tau^{n+1}_{wm/fi}\left(\omega_{m/fi}C^{n+1}_{wfi}+(1-\omega_{m/fi})C^{n+1}_{wmi}\right)+\left(\phi^{n+1}_{mi}S^{n+1}_{wmi}K^{n+1}_{wm/fi}\right)\left(C^{n+1}_{wfi}-C^{n+1}_{wmi}\right)=$$

$$\frac{V_{Ri}}{\Delta t}\left[\left\{\phi_{mi}S_{wmi}C_{wmi}+(1-\phi_t)\rho_s C_{smi}\right\}^{n+1}-\left\{\phi_{mi}S_{wmi}C_{wmi}+(1-\phi_t)\rho_s C_{smi}\right\}^{n}\right]+$$

$$V_{Ri}\left\{\phi_{mi}S_{wmi}C_{wmi}+(1-\phi_t)\rho_s C_{smi}\right\}^{n+1}\lambda-V_{Ri}\sum_{l=1}^{L}\left[\left\{\phi_{mi}S_{wmi}C_{wmli}+(1-\phi_t)\rho_s C_{smli}\right\}\lambda_l\right]^{n+1},$$

where

$$\mu_{i1}=\frac{\Delta x_i}{\Delta x_i+\Delta x_{i-1}},\ \mu_{i2}=\frac{\Delta x_{i+1}}{\Delta x_{i+1}+\Delta x_{i+2}},\ \mu_{i3}=\frac{\Delta x_i}{\Delta x_i+\Delta x_{i+1}},\ \mu_{i4}=\frac{\Delta x_{i-1}}{\Delta x_{i-1}+\Delta x_{i-2}},$$

$$\mu_{j1}=\frac{\Delta y_j}{\Delta y_j+\Delta y_{j-1}},\ \mu_{j2}=\frac{\Delta y_{j+1}}{\Delta y_{j+1}+\Delta y_{j+2}},\ \mu_{j3}=\frac{\Delta y_j}{\Delta y_j+\Delta y_{j+1}},\ \mu_{j4}=\frac{\Delta y_{j-1}}{\Delta y_{j-1}+\Delta y_{j-2}},$$

$$\mu_{k1} = \frac{\Delta z_k}{\Delta z_k + \Delta z_{k-1}},\ \mu_{k2} = \frac{\Delta z_{k+1}}{\Delta z_{k+1} + \Delta z_{k+2}},\ \mu_{k3} = \frac{\Delta z_k}{\Delta z_k + \Delta z_{k+1}},\ \mathit{and}\ \mu_{k4} = \frac{\Delta z_{k-1}}{\Delta z_{k-1} + \Delta z_{k-2}}$$

the corresponding linear system of the fracture-flow equation is:

$$\begin{aligned}&A_{2f}C^{n+1}_{wfk-2} + A_{1f}C^{n+1}_{wfk-1} + B_{2f}C^{n+1}_{wfj-2} + B_{1f}C^{n+1}_{wfj-1} + C_{2f}C^{n+1}_{wfi-2} + C_{1f}C^{n+1}_{wfi-1} + D_{1f}C^{n+1}_{wfijk} + D_{2f}C^{n+1}_{wmijk} + \\ &E_{1f}C^{n+1}_{wfi+1} + E_{2f}C^{n+1}_{wfi+2} + F_{1f}C^{n+1}_{wfj+1} + F_{2f}C^{n+1}_{wfj+2} + G_{1f}C^{n+1}_{wfk+1} + G_{2f}C^{n+1}_{wfk+2} = R^{n+1}_f,\end{aligned} \qquad \mathbf{4.84}$$

where

$$A_{1f} = HD^{n+1}_{wfk-\frac{1}{2}} - \omega_{fk}(1+\mu_{k3})q^{n+1}_{wfk-\frac{1}{2}} - \omega_{fk+1}\mu_{k1}q^{n+1}_{wfk+\frac{1}{2}},$$

$$A_{2f} = -\omega_{fk}\mu_{k3}q^{n+1}_{wfk-\frac{1}{2}},$$

$$B_{1f} = HD^{n+1}_{wfj-\frac{1}{2}} - \omega_{fj}(1+\mu_{j3})q^{n+1}_{wfj-\frac{1}{2}} - \omega_{fj+1}\mu_{j1}q^{n+1}_{wfj+\frac{1}{2}},$$

$$B_{2f} = -\omega_{fj}\mu_{j3}q^{n+1}_{wfj-\frac{1}{2}},$$

$$C_{1f} = HD^{n+1}_{wfi-\frac{1}{2}} - \omega_{fi}(1+\mu_{i3})q^{n+1}_{wfi-\frac{1}{2}} - \omega_{fi+1}\mu_{i1}q^{n+1}_{wfi+\frac{1}{2}},$$

$$C_{2f} = -\omega_{fi}\mu_{i3}q^{n+1}_{wfi-\frac{1}{2}},$$

$$E_{1f} = HD^{n+1}_{wfi+\frac{1}{2}} + (1-\omega_{fi+1})(1+\mu_{i2})q^{n+1}_{wfi+\frac{1}{2}} + (1-\omega_{fi})\mu_{i4}q^{n+1}_{wfi-\frac{1}{2}},$$

$$E_{2f} = -(1-\omega_{fi+1})\mu_{i2}q^{n+1}_{wfi+\frac{1}{2}},$$

$$F_{1f} = HD^{n+1}_{wfj+\frac{1}{2}} + (1-\omega_{fj+1})(1+\mu_{j2})q^{n+1}_{wfj+\frac{1}{2}} + (1-\omega_{fj})\mu_{j4}q^{n+1}_{wfj-\frac{1}{2}},$$

$$F_{2f} = -(1-\omega_{fj+1})\mu_{j2}q^{n+1}_{wfj+\frac{1}{2}},$$

$$G_{1f} = HD^{n+1}_{wfk+\frac{1}{2}} + \left(1-\omega_{fk+1}\right)\left(1+\mu_{k2}\right)q^{n+1}_{wfk+\frac{1}{2}} + \left(1-\omega_{fk}\right)\mu_{k4}q^{n+1}_{wfk-\frac{1}{2}},$$

$$G_{2f} = -\left(1-\omega_{fk+1}\right)\mu_{k2}q^{n+1}_{wfk+\frac{1}{2}},$$

$$\begin{aligned}D^{n+1}_{1f} = & -HD^{n+1}_{wfk-\frac{1}{2}} - HD^{n+1}_{wfj-\frac{1}{2}} - HD^{n+1}_{wfi-\frac{1}{2}} - HD^{n+1}_{wfk+\frac{1}{2}} - HD^{n+1}_{wfj+\frac{1}{2}} - HD^{n+1}_{wfi+\frac{1}{2}} - \\ & q^{n+1}_{wfk-\frac{1}{2}}\left(1-\omega_{fk}\right)\left(1+\mu_{k4}\right) - q^{n+1}_{wfj-\frac{1}{2}}\left(1-\omega_{fj}\right)\left(1+\mu_{j4}\right) - q^{n+1}_{wfi-\frac{1}{2}}\left(1-\omega_{fi}\right)\left(1+\mu_{i4}\right) + \\ & q^{n+1}_{wfk+\frac{1}{2}}\omega_{fk+1}\left(1+\mu_{k1}\right) + q^{n+1}_{wfj+\frac{1}{2}}\omega_{fj+1}\left(1+\mu_{j1}\right) + q^{n+1}_{wfi+\frac{1}{2}}\omega_{fi+1}\left(1+\mu_{i1}\right) - \\ & \frac{V_{Ri}}{\Delta t}\left\{\phi_{fi}S_{wfi} + \left(1-\phi_{t}\right)\rho_{s}\xi_{fi}\right\}^{n+1} + V_{Ri}\left\{\phi_{fi}S_{wfi} + \left(1-\phi_{t}\right)\rho_{s}\xi_{fi}\right\}^{n+1}\lambda - \tau^{n+1}_{w/mfi}\omega_{m/fi} - \phi^{n+1}_{m}S^{n+1}_{wm}K^{n+1}_{wm/fi},\end{aligned}$$

$$D^{n+1}_{2f} = -\tau^{n+1}_{wm/fi}\left(1-\omega_{mLfi}\right) + \phi^{n+1}_{m}S^{n+1}_{wm}K^{n+1}_{wm/fi},$$

and

$$R^{n+1}_{f} = \frac{V_{Ri}}{\Delta t}\left\{\phi_{fi}S_{wfi} + \left(1-\phi_{t}\right)\rho_{s}\xi_{fi}\right\}^{n}C^{n}_{wfi} - V_{Ri}\sum_{l=1}^{L}\left\{\left\{\phi_{fi}S_{wfi} + \left(1-\phi_{t}\right)\rho_{s}\xi_{fli}\right\}^{n+1}\lambda_{l}C^{n+1}_{wfli}\right\} - C^{*n+1}_{wfi}Q^{n+1}_{wfi},$$

and the linear system of the matrix equation is:

$$\begin{aligned}& A_{2m}C^{n+1}_{wmk-2} + A_{1m}C^{n+1}_{wmk-1} + B_{2m}C^{n+1}_{wmj-2} + B_{1m}C^{n+1}_{wmj-1} + C_{2m}C^{n+1}_{wmi-2} + C_{1m}C^{n+1}_{wmi-1} + D_{1m}C^{n+1}_{wmijk} + D_{2m}C^{n+1}_{wfijk} + \\ & E_{1m}C^{n+1}_{wmi+1} + E_{2m}C^{n+1}_{wmi+2} + F_{1m}C^{n+1}_{wmj+1} + F_{2m}C^{n+1}_{wmj+2} + G_{1m}C^{n+1}_{wmk+1} + G_{2m}C^{n+1}_{wmk+2} = R^{n+1}_{m},\end{aligned} \qquad \textbf{4.85}$$

where

$$A_{1m} = HD^{n+1}_{wmk-\frac{1}{2}} - \omega_{mk}\left(1+\mu_{k3}\right)q^{n+1}_{wmk-\frac{1}{2}} - \omega_{mk+1}\mu_{k1}q^{n+1}_{wmk+\frac{1}{2}},$$

$$A_{2m} = -\omega_{mk}\mu_{k3}q^{n+1}_{wmk-\frac{1}{2}},$$

$$B_{1m} = HD^{n+1}_{wmj-\frac{1}{2}} - \omega_{mj}\left(1+\mu_{j3}\right)q^{n+1}_{wmj-\frac{1}{2}} - \omega_{mj+1}\mu_{j1}q^{n+1}_{wmj+\frac{1}{2}},$$

$$B_{2m} = -\omega_{mj}\mu_{j3}q^{n+1}_{wmj-\frac{1}{2}},$$

$$C_{1m} = HD^{n+1}_{wmi-\frac{1}{2}} - \omega_{mi}\left(1+\mu_{i3}\right)q^{n+1}_{wmi-\frac{1}{2}} - \omega_{mi+1}\mu_{i1}q^{n+1}_{wmi+\frac{1}{2}},$$

$$C_{2m} = -\omega_{mi}\mu_{i3}q^{n+1}_{wmi-\frac{1}{2}},$$

$$E_{1m} = HD^{n+1}_{wmi+\frac{1}{2}} + \left(1-\omega_{mi+1}\right)\left(1+\mu_{i2}\right)q^{n+1}_{wmi+\frac{1}{2}} + \left(1-\omega_{mi}\right)\mu_{i4}q^{n+1}_{wmi-\frac{1}{2}},$$

$$E_{2m} = -\left(1-\omega_{mi+1}\right)\mu_{i2}q^{n+1}_{wmi+\frac{1}{2}},$$

$$F_{1m} = HD^{n+1}_{wmj+\frac{1}{2}} + \left(1-\omega_{mj+1}\right)\left(1+\mu_{j2}\right)q^{n+1}_{wmj+\frac{1}{2}} + \left(1-\omega_{mj}\right)\mu_{j4}q^{n+1}_{wmj-\frac{1}{2}},$$

$$F_{2m} = -\left(1-\omega_{mj+1}\right)\mu_{j2}q^{n+1}_{wmj+\frac{1}{2}},$$

$$G_{1m} = HD^{n+1}_{wmk+\frac{1}{2}} + \left(1-\omega_{mk+1}\right)\left(1+\mu_{k2}\right)q^{n+1}_{wmk+\frac{1}{2}} + \left(1-\omega_{mk}\right)\mu_{k4}q^{n+1}_{wmk-\frac{1}{2}},$$

$$G_{2m} = -\left(1-\omega_{mk+1}\right)\mu_{k2}q^{n+1}_{wmk+\frac{1}{2}},$$

$$D^{n+1}_{1m} = -HD^{n+1}_{wmk-\frac{1}{2}} - HD^{n+1}_{wmj-\frac{1}{2}} - HD^{n+1}_{wmi-\frac{1}{2}} - HD^{n+1}_{wmk+\frac{1}{2}} - HD^{n+1}_{wmj+\frac{1}{2}} - HD^{n+1}_{wmi+\frac{1}{2}} -$$

$$q^{n+1}_{wmk-\frac{1}{2}}\left(1-\omega_{mk}\right)\left(1+\mu_{k4}\right) - q^{n+1}_{wmj-\frac{1}{2}}\left(1-\omega_{mj}\right)\left(1+\mu_{j4}\right) - q^{n+1}_{wmi-\frac{1}{2}}\left(1-\omega_{mi}\right)\left(1+\mu_{i4}\right) +$$
$$q^{n+1}_{wmk+\frac{1}{2}}\omega_{mk+1}\left(1+\mu_{k1}\right) + q^{n+1}_{wmj+\frac{1}{2}}\omega_{mj+1}\left(1+\mu_{j1}\right) + q^{n+1}_{wmi+\frac{1}{2}}\omega_{mi+1}\left(1+\mu_{i1}\right) -$$
$$\frac{V_{Ri}}{\Delta t}\left\{\phi_{mi}S_{wmi} + \left(1-\phi_{t}\right)\rho_{s}\xi_{mi}\right\}^{n+1} + V_{Ri}\left\{\phi_{mi}S_{wmi} + \left(1-\phi_{t}\right)\rho_{s}\xi_{mi}\right\}^{n+1}\lambda + \tau^{n+1}_{w/mfi}\left(1-\omega_{m/fi}\right) - \phi^{n+1}_{m}S^{n+1}_{wm}K^{n+1}_{wm/fi},$$
$$D^{n+1}_{2m} = \tau^{n+1}_{wm/fi}\omega_{m/fi} + \phi^{n+1}_{m}S^{n+1}_{wm}K^{n+1}_{wm/fi},$$

and

$$R^{n+1}_{m} = \frac{V_{Ri}}{\Delta t}\left\{\phi_{mi}S_{wmi} + \left(1-\phi_{t}\right)\rho_{s}\xi_{mi}\right\}^{n}C^{n}_{wmi} - V_{Ri}\sum_{l=1}^{L}\left\{\left\{\phi_{mi}S_{wmi} + \left(1-\phi_{t}\right)\rho_{s}\xi_{mli}\right\}^{n+1}\lambda_{l}C^{n+1}_{wmli}\right\} - C^{*n+1}_{wmi}Q^{n+1}_{wmi}.$$

The second approach in the discretization of the two-point upstream winding is the split-operator method. In the split-operator method, the equation is discretized similarly but the time level for the concentration that appears in the second diagonal of the numerical matrix in each dimension (i.e., *i-2,i+2, j-2, j+2, k+2, and k-2* diagonals) is treated explicitly. This, in turn, will lead to a linear system similar to Equations 4.84 and 4.85 except that the right-hand side of the equation will be equivalent to

$$R_f^{n+1} = R_f^{n+1}\ (Equation\ 4.84) + A_{2f}C^n_{wfk-2} + B_{2f}C^n_{wfj-2} + C_{2f}C^n_{wfi-2} \\ + E_{2f}C^n_{wfi+2} + F_{2f}C^n_{wfj+2} + G_{2f}C^n_{wfk+2} \qquad 4.86$$

for the fracture, and

$$R_m^{n+1} = R_m^{n+1}\ (Equation\ 4.85) + A_{2m}C^n_{wmk-2} + B_{2m}C^n_{wmj-2} + C_{2m}C^n_{wmi-2} \\ + E_{2m}C^n_{wmi+2} + F_{2m}C^n_{wmj+2} + G_{2m}C^n_{wmk+2} \qquad 4.87$$

for the matrix. The accuracy of the split operator depends on the size of the time step, which resides somewhere between that of the one-point upwinding and that of the two-point upwinding. However, the saving in computation time is substantial.

4.4.4 Mid-Point Weighting Discretization of Transport Equations

Fracture Equation

$$\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{i+\frac{1}{2}}\left(C^{n+1}_{wfi+1} - C^{n+1}_{wfi}\right) - \left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{i-\frac{1}{2}}\left(C^{n+1}_{wfi} - C^{n+1}_{wfi-1}\right) +$$

$$\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{j+\frac{1}{2}}\left(C^{n+1}_{wfj+1} - C^{n+1}_{wfj}\right) - \left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{j-\frac{1}{2}}\left(C^{n+1}_{wfj} - C^{n+1}_{wfj-1}\right) +$$

$$\left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{k+\frac{1}{2}}\left(C^{n+1}_{wfk+1} - C^{n+1}_{wfk}\right) - \left(\phi_f^{n+1}S_{wf}^{n+1}K_{wf}^{n+1}\right)_{k-\frac{1}{2}}\left(C^{n+1}_{wfk} - C^{n+1}_{wfk-1}\right) +$$

$$q^{n+1}_{wfi+\frac{1}{2}}\left[(0.5+\mu_{i2})C^{n+1}_{wfi+1} + (0.5-\mu_{i2})C^{n+1}_{wfi}\right] - q^{n+1}_{wfi-\frac{1}{2}}\left[(0.5+\mu_{i1})C^{n+1}_{wfi} + (0.5-\mu_{i1})C^{n+1}_{wfi-1}\right] +$$

$$q^{n+1}_{wfj+\frac{1}{2}}\left[(0.5+\mu_{j2})C^{n+1}_{wfj+1} + (0.5-\mu_{j2})C^{n+1}_{wfj}\right] - q^{n+1}_{wfj-\frac{1}{2}}\left[(0.5+\mu_{j1})C^{n+1}_{wfj} + (0.5-\mu_{j1})C^{n+1}_{wfj-1}\right] + \qquad 4.88$$

$$q^{n+1}_{wfk+\frac{1}{2}}\left[(0.5+\mu_{k2})C^{n+1}_{wfk+1} + (0.5-\mu_{k2})C^{n+1}_{wfk}\right] - q^{n+1}_{wfk-\frac{1}{2}}\left[(0.5+\mu_{k1})C^{n+1}_{wfk} + (0.5-\mu_{k1})C^{n+1}_{wfk-1}\right] +$$

$$C_{wfi}^{*n+1} Q_{wfi}^{n+1} - \tau_{wm/fi}^{n+1}\left(\omega_{m/fi} C_{wfi}^{n+1} + (1-\omega_{m/fi}) C_{wmi}^{n+1}\right) - \left(\phi_{mi}^{n+1} S_{wmi}^{n+1} K_{wm/fi}^{n+1}\right)\left(C_{wfi}^{n+1} - C_{wmi}^{n+1}\right) =$$

$$\frac{V_{Ri}}{\Delta t}\left[\left\{\phi_{fi} S_{wfi} C_{wfi} + (1-\phi_t)\rho_s C_{sfi}\right\}^{n+1} - \left\{\phi_{fi} S_{wfi} C_{wfi} + (1-\phi_t)\rho_s C_{sfi}\right\}^{n}\right] +$$

$$V_{Ri}\left\{\phi_{fi} S_{wfi} C_{wfi} + (1-\phi_t)\rho_s C_{sfi}\right\}^{n+1}\lambda - V_{Ri}\sum_{l=1}^{L}\left[\left\{\phi_{fi} S_{wfi} C_{wfli} + (1-\phi_t)\rho_s C_{sfli}\right\}\lambda_l\right]^{n+1}$$

Matrix Equation

$$\left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{i+\frac{1}{2}}\left(C_{wmi+1}^{n+1} - C_{wmi}^{n+1}\right) - \left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{i-\frac{1}{2}}\left(C_{wmi}^{n+1} - C_{wmi-1}^{n+1}\right) +$$
$$\left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{j+\frac{1}{2}}\left(C_{wmj+1}^{n+1} - C_{wmj}^{n+1}\right) - \left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{j-\frac{1}{2}}\left(C_{wmj}^{n+1} - C_{wmj-1}^{n+1}\right) +$$
$$\left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{k+\frac{1}{2}}\left(C_{wmk+1}^{n+1} - C_{wmk}^{n+1}\right) - \left(\phi_m^{n+1} S_{wm}^{n+1} K_{wm}^{n+1}\right)_{k-\frac{1}{2}}\left(C_{wmk}^{n+1} - C_{wmk-1}^{n+1}\right) +$$

$$q_{wmi+\frac{1}{2}}^{n+1}\left[(0.5+\mu_{i2}) C_{wmi+1}^{n+1} + (0.5-\mu_{i2}) C_{wmi}^{n+1}\right] - q_{wmi-\frac{1}{2}}^{n+1}\left[(0.5+\mu_{i1}) C_{wmi}^{n+1} + (0.5-\mu_{i1}) C_{wmi-1}^{n+1}\right] +$$

$$q_{wmj+\frac{1}{2}}^{n+1}\left[(0.5+\mu_{j2}) C_{wmj+1}^{n+1} + (0.5-\mu_{j2}) C_{wmj}^{n+1}\right] - q_{wmj-\frac{1}{2}}^{n+1}\left[(0.5+\mu_{j1}) C_{wmj}^{n+1} + (0.5-\mu_{j1}) C_{wmj-1}^{n+1}\right] + \qquad \textbf{4.89}$$

$$q_{wmk+\frac{1}{2}}^{n+1}\left[(0.5+\mu_{k2}) C_{wmk+1}^{n+1} + (0.5-\mu_{k2}) C_{wmk}^{n+1}\right] - q_{wmk-\frac{1}{2}}^{n+1}\left[(0.5+\mu_{k1}) C_{wmk}^{n+1} + (0.5-\mu_{k1}) C_{wmk-1}^{n+1}\right] +$$

$$C_{wmi}^{*n+1} Q_{wmi}^{n+1} + \tau_{wm/fi}^{n+1}\left(\omega_{m/fi} C_{wfi}^{n+1} + (1-\omega_{m/fi}) C_{wmi}^{n+1}\right) + \left(\phi_{mi}^{n+1} S_{wmi}^{n+1} K_{wm/fi}^{n+1}\right)\left(C_{wfi}^{n+1} - C_{wmi}^{n+1}\right) =$$

$$\frac{V_{Ri}}{\Delta t}\left[\left\{\phi_{mi} S_{wmi} C_{wmi} + (1-\phi_t)\rho_s C_{smi}\right\}^{n+1} - \left\{\phi_{mi} S_{wmi} C_{wmi} + (1-\phi_t)\rho_s C_{smi}\right\}^{n}\right] +$$

$$V_{Ri}\left\{\phi_{mi} S_{wmi} C_{wmi} + (1-\phi_t)\rho_s C_{smi}\right\}^{n+1}\lambda - V_{Ri}\sum_{l=1}^{L}\left[\left\{\phi_{mi} S_{wmi} C_{wmli} + (1-\phi_t)\rho_s C_{smli}\right\}\lambda_l\right]^{n+1}$$

where

$$\mu_{i1} = \frac{1}{4}\frac{\Delta x_{i-1} - \Delta x_i}{\Delta x_{i-1} + \Delta x_i},\ \mu_{i2} = \frac{1}{4}\frac{\Delta x_i - \Delta x_{i+1}}{\Delta x_i + \Delta x_{i+1}},\ \mu_{j1} = \frac{1}{4}\frac{\Delta y_{j-1} - \Delta y_j}{\Delta y_{j-1} + \Delta y_j},\ \mu_{j2} = \frac{1}{4}\frac{\Delta y_j - \Delta y_{j+1}}{\Delta y_j + \Delta y_{j+1}},$$

$$\mu_{k1} = \frac{1}{4}\frac{\Delta z_{k-1} - \Delta z_k}{\Delta z_{k-1} + \Delta z_k}, \ and \ \mu_{k2} = \frac{1}{4}\frac{\Delta z_k - \Delta z_{k+1}}{\Delta z_k + \Delta z_{k+1}}$$

A similar linear system as in Equations 4.79 and 4.81 is formed, in which

$$A_f = HD^{n+1}_{wfk-\frac{1}{2}} - (0.5 + \mu_{k1})q^{n+1}_{wfk-\frac{1}{2}}$$

$$A_m = HD^{n+1}_{wmk-\frac{1}{2}} - (0.5 + \mu_{k1})q^{n+1}_{wmk-\frac{1}{2}}$$

$$B_f = HD^{n+1}_{wfj-\frac{1}{2}} - (0.5 + \mu_{j1})q^{n+1}_{wfj-\frac{1}{2}}$$

$$B_m = HD^{n+1}_{wmj-\frac{1}{2}} - (0.5 + \mu_{j1})q^{n+1}_{wmj-\frac{1}{2}}$$

$$C_f = HD^{n+1}_{wfi-\frac{1}{2}} - (0.5 + \mu_{i1})q^{n+1}_{wfi-\frac{1}{2}}$$

$$C_m = HD^{n+1}_{wmi-\frac{1}{2}} - (0.5 + \mu_{i1})q^{n+1}_{wmi-\frac{1}{2}}$$

$$E_f = HD^{n+1}_{wfi+\frac{1}{2}} + (0.5 + \mu_{i2})q^{n+1}_{wfi+\frac{1}{2}}$$

$$E_m = HD^{n+1}_{wmi+\frac{1}{2}} + (0.5 + \mu_{i2})q^{n+1}_{wmi+\frac{1}{2}}$$

$$F_f = HD^{n+1}_{wfj+\frac{1}{2}} + (0.5 + \mu_{j2})q^{n+1}_{wfj+\frac{1}{2}}$$

$$F_m = HD^{n+1}_{wmj+\frac{1}{2}} + (0.5 + \mu_{j2})q^{n+1}_{wmj+\frac{1}{2}}$$

$$G_f = HD^{n+1}_{wfk+\frac{1}{2}} + (0.5 + \mu_{k2})q^{n+1}_{wfk+\frac{1}{2}}$$

$$G_m = HD^{n+1}_{wmk+\frac{1}{2}} + (0.5 + \mu_{k2})q^{n+1}_{wmk+\frac{1}{2}}$$

and

$$D_{1f}^{n+1} = -HD_{wfk-\frac{1}{2}}^{n+1} - HD_{wfj-\frac{1}{2}}^{n+1} - HD_{wfi-\frac{1}{2}}^{n+1} - HD_{wfk+\frac{1}{2}}^{n+1} - HD_{wfj+\frac{1}{2}}^{n+1} - HD_{wfi+\frac{1}{2}}^{n+1} - q_{wfk-\frac{1}{2}}^{n+1}(0.5+\mu_{ki}) -$$

$$q_{wfj-\frac{1}{2}}^{n+1}(0.5+\mu_{j1}) - q_{wfi-\frac{1}{2}}^{n+1}(0.5+\mu_{i1}) + q_{wfk+\frac{1}{2}}^{n+1}(0.5-\mu_{k2}) + q_{wfj+\frac{1}{2}}^{n+1}(0.5-\mu_{j2}) + q_{wfi+\frac{1}{2}}^{n+1}(0.5-\mu_{i2}) -$$

$$\frac{V_{Ri}}{\Delta t}\left\{\phi_{fi}S_{wfi} + (1-\phi_t)\rho_s\xi_{fi}\right\}^{n+1} + V_{Ri}\left\{\phi_{fi}S_{wfi} + (1-\phi_t)\rho_s\xi_{fi}\right\}^{n+1}\lambda - \tau_{w/mfi}^{n+1}\omega_{m/fi} - \phi_m^{n+1}S_{wm}^{n+1}K_{wm/fi}^{n+1}$$

$$D_{1m}^{n+1} = -HD_{wmk-\frac{1}{2}}^{n+1} - HD_{wmj-\frac{1}{2}}^{n+1} - HD_{wmi-\frac{1}{2}}^{n+1} - HD_{wmk+\frac{1}{2}}^{n+1} - HD_{wmj+\frac{1}{2}}^{n+1} - HD_{wmi+\frac{1}{2}}^{n+1} - q_{wmk-\frac{1}{2}}^{n+1}(0.5+\mu_{k1}) -$$

$$q_{wmj-\frac{1}{2}}^{n+1}(0.5+\mu_{j1}) - q_{wmi-\frac{1}{2}}^{n+1}(0.5+\mu_{i1}) + q_{wmk+\frac{1}{2}}^{n+1}(0.5-\mu_{k2}) + q_{wmj+\frac{1}{2}}^{n+1}(0.5-\mu_{j2}) + q_{wmi+\frac{1}{2}}^{n+1}(0.5-\mu_{i2}) -$$

$$\frac{V_{Ri}}{\Delta t}\left\{\phi_{mi}S_{wmi} + (1-\phi_t)\rho_s\xi_{mi}\right\}^{n+1} + V_{Ri}\left\{\phi_{mi}S_{wmi} + (1-\phi_t)\rho_s\xi_{mi}\right\}^{n+1}\lambda + \tau_{w/mfi}^{n+1}\left(1-\omega_{m/fi}\right) - \phi_m^{n+1}S_{wm}^{n+1}K_{wm/fi}^{n+1}$$

$$D_{2f}^{n+1} = -\tau_{wm/fi}^{n+1}\left(1-\omega_{m/fi}\right) + \phi_m^{n+1}S_{wm}^{n+1}K_{wm/fi}^{n+1}$$

$$D_{2m}^{n+1} = \tau_{wm/fi}^{n+1}\omega_{m/fi} + \phi_m^{n+1}S_{wm}^{n+1}K_{wm/fi}^{n+1}$$

$$R_f^{n+1} = \frac{V_{Ri}}{\Delta t}\left\{\phi_{fi}S_{wfi} + (1-\phi_t)\rho_s\xi_{fi}\right\}^n C_{wfi}^n - V_{Ri}\sum_{l=1}^{L}\left\{\left\{\phi_{fi}S_{wfi} + (1-\phi_t)\rho_s\xi_{fli}\right\}^{n+1}\lambda_l C_{wfli}^{n+1}\right\} - C_{wfi}^{*n+1}Q_{wfi}^{n+1}$$

$$R_m^{n+1} = \frac{V_{Ri}}{\Delta t}\left\{\phi_{mi}S_{wmi} + (1-\phi_t)\rho_s\xi_{mi}\right\}^n C_{wmi}^n - V_{Ri}\sum_{l=1}^{L}\left\{\left\{\phi_{mi}S_{wmi} + (1-\phi_t)\rho_s\xi_{mli}\right\}^{n+1}\lambda_l C_{wmli}^{n+1}\right\} - C_{wmi}^{*n+1}Q_{wmi}^{n+1}.$$

It is worth mentioning that in all the above methods no fictitious or interior grid points are used in the discretization of the boundary conditions. In two-point upwinding, there might be, depending on the direction of the flow, a need for such points. However, one-point upstream is used instead. This has been done in the problem at hand because the source is located in the middle of the domain with negligible or no transport near boundaries. In the cases when there is an active flow field near the boundaries, such treatment (one-point upstream) may lead to a first-order method even if two-point upstream weighting is used in the interior.

The transport equations (Equations 4.38 and 4.39) have N *G unknowns for single porosity and 2N*G unknowns for dual-porosity and dual-permeability formulations, where N is the number of equations in each grid block and G is the number of grid blocks in the simulated spatial domain.

The system of governing partial differential equations (two for each transported constituent) is strongly coupled one to the other because of the strong contribution from parental decay to the concentration of the immediate daughter. Consequently, the sequential method is used to solve the system implicitly (if the implicit method is used). In the sequential method, the solution proceeds progressively from the top to the bottom of each radioactive chain. Therefore, the contribution to any daughter from parental decay will be immediately calculable.

The matrix resulting from one-point upstream winding and the mid-point weighting resembles the following for a two-dimensional system of 3x3 grid blocks

$$\left[\begin{array}{cccccccccccccccccc}
f & m & f & 0 & 0 & 0 & f & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f & m & 0 & m & 0 & 0 & 0 & m & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f & 0 & f & m & f & 0 & 0 & 0 & f & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & m & f & m & 0 & m & 0 & 0 & 0 & m & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & f & 0 & f & m & 0 & 0 & 0 & 0 & f & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & m & f & m & 0 & 0 & 0 & 0 & 0 & m & 0 & 0 & 0 & 0 & 0 & 0 \\
f & 0 & 0 & 0 & 0 & 0 & f & m & f & 0 & 0 & 0 & f & 0 & 0 & 0 & 0 & 0 \\
0 & m & 0 & 0 & 0 & 0 & f & m & 0 & m & 0 & 0 & 0 & m & 0 & 0 & 0 & 0 \\
0 & 0 & f & 0 & 0 & 0 & f & 0 & f & m & f & 0 & 0 & 0 & f & 0 & 0 & 0 \\
0 & 0 & 0 & m & 0 & 0 & 0 & m & f & m & 0 & m & 0 & 0 & 0 & m & 0 & 0 \\
0 & 0 & 0 & 0 & f & 0 & 0 & 0 & f & 0 & f & m & 0 & 0 & 0 & 0 & f & 0 \\
0 & 0 & 0 & 0 & 0 & m & 0 & 0 & 0 & m & f & m & 0 & 0 & 0 & 0 & 0 & m \\
0 & 0 & 0 & 0 & 0 & 0 & f & 0 & 0 & 0 & 0 & 0 & f & m & f & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & m & 0 & 0 & 0 & 0 & f & m & 0 & m & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & f & 0 & 0 & 0 & f & 0 & f & m & f & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m & 0 & 0 & 0 & m & f & m & 0 & m \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & f & 0 & 0 & 0 & f & 0 & f & m \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m & 0 & 0 & 0 & m & f & m
\end{array}\right]$$

Since two-point upstream winding uses information from the grid block further in the upstream direction, the band width of the matrix will be doubled compared with the previous one. Also the matrix will be less sparse. For the same system of 3x3, the numerical matrix is:

$$\left[\begin{array}{cccccccccccccccccc}
f & m & f & 0 & f & 0 & f & 0 & 0 & 0 & 0 & 0 & f & 0 & 0 & 0 & 0 & 0 \\
f & m & 0 & m & 0 & m & 0 & m & 0 & 0 & 0 & 0 & 0 & m & 0 & 0 & 0 & 0 \\
f & 0 & f & m & f & 0 & 0 & 0 & f & 0 & 0 & 0 & 0 & 0 & f & 0 & 0 & 0 \\
0 & m & f & m & 0 & m & 0 & 0 & 0 & m & 0 & 0 & 0 & 0 & 0 & m & 0 & 0 \\
f & 0 & f & 0 & f & m & 0 & 0 & 0 & 0 & f & 0 & 0 & 0 & 0 & 0 & f & 0 \\
0 & m & 0 & m & f & m & 0 & 0 & 0 & 0 & 0 & m & 0 & 0 & 0 & 0 & 0 & m \\
f & 0 & 0 & 0 & 0 & 0 & f & m & f & 0 & f & 0 & f & 0 & 0 & 0 & 0 & 0 \\
0 & m & 0 & 0 & 0 & 0 & f & m & 0 & m & 0 & m & 0 & m & 0 & 0 & 0 & 0 \\
0 & 0 & f & 0 & 0 & 0 & f & 0 & f & m & f & 0 & 0 & 0 & f & 0 & 0 & 0 \\
0 & 0 & 0 & m & 0 & 0 & 0 & m & f & m & 0 & m & 0 & 0 & 0 & m & 0 & 0 \\
0 & 0 & 0 & 0 & f & 0 & f & 0 & f & 0 & f & m & 0 & 0 & 0 & 0 & f & 0 \\
0 & 0 & 0 & 0 & 0 & m & 0 & m & 0 & m & f & m & 0 & 0 & 0 & 0 & 0 & m \\
f & 0 & 0 & 0 & 0 & 0 & f & 0 & 0 & 0 & 0 & 0 & f & m & f & 0 & f & 0 \\
0 & m & 0 & 0 & 0 & 0 & 0 & m & 0 & 0 & 0 & 0 & f & m & 0 & m & 0 & m \\
0 & 0 & f & 0 & 0 & 0 & 0 & 0 & f & 0 & 0 & 0 & f & 0 & f & m & f & 0 \\
0 & 0 & 0 & m & 0 & 0 & 0 & 0 & 0 & m & 0 & 0 & 0 & m & f & m & 0 & m \\
0 & 0 & 0 & 0 & f & 0 & 0 & 0 & 0 & 0 & f & 0 & f & 0 & f & 0 & f & m \\
0 & 0 & 0 & 0 & 0 & m & 0 & 0 & 0 & 0 & 0 & m & 0 & m & 0 & m & f & m
\end{array}\right]$$

where *f* refers to fracture entry, *m* refers to matrix entries, and *0* refers to zero entry.

Standard Gaussian elimination is used to invert the numerical matrix. Two techniques are applied to reduce the size of the numerical matrix to more manageable dimensions. In the first technique, an optimum dimension (in standard ordering) is chosen to define the connectivity of the grid blocks. The guideline for this choice is the minimum number of grid blocks. Hence, for a two dimensional problem in x and y , the numbering will start from min(x,y) and proceed to the next dimension. The second technique is used to straighten the numerical matrix diagonals and limit the calculation to the entries between the uppermost and the lowermost diagonals. Therefore, for a single-porosity system in two dimensions, a pentagonal matrix of IBW*G is inverted instead of G*G matrix, where IBW is the band width.

4.5 Other Important Physical and Chemical Aspects of the Modeling

4.5.1 Treatment of Mobilization Involving Colloidal Materials

In order to minimize unnecessary computations, dissolved and colloidal mobilized species have been combined for transport within NUTS. This is justified because molecular diffusion and sorption are the only physical processes that could cause differences in the modeled transport of dissolved versus colloidal species, and since they have been set to zero, the species will transport identically. The combination of dissolved and colloidal species is performed prior to NUTS in an ALGEBRA calculation. In this calculation, four types of colloid-related mobilization are considered, as follows: (i) condensation of hydrolized actinide ions into intrinsic colloids, and sorption onto already existent colloidal materials of (ii) humic, (iii) microbial, and (iv) mineral origen. Maximum concentrations for these four different types of colloidal particulates are *added* to the maximum concentration for dissolution for each of the radioisotopes and (sampled) oxidation states. The sum of the five maximum concentrations is then used in a single dissolution-like computation that estimates the net mobilization due to colloids and dissolution combined.

4.5.2 Lumped Radioisotopes

To keep computational resources at a reasonable level, and because several of the radioisotopes in the inventory have been shown to be unimportant in terms of release calculations, inventory radioisotopes have been (i) combined, where possible, or (ii) omitted, where justifiable, so as to reduce the list of 29 actual radioisotopes to be monitored to 5 "equivalent" radioisotopes. The rules were to combine the effects of some radioisotopes (by adding their inventories), providing they have similar half lives, and to omit 10 others whose effects on release values have been shown to be negligible (less than 0.01 normalized EPA units). For example, ^{233}U and ^{234}U are 30 to 100 times more important than other uranium isotopes in terms of normalized EPA releases. Thus, ^{235}U, ^{236}U, and ^{238}U can be ignored without a significant impact on release calculations. ^{233}U and ^{234}U comprise less than 1% of the total uranium inventory mass. To account for that, the solubility of uranium is decreased by a factor of 100. Lastly, since the half lives are similar, ^{233}U and ^{234}U are combined into a single equivalent radioisotope. Thus, 5 radioisotopes have thereby been reduced to one equivalent radioisotope. Similarly, ^{229}Th is absorbed into ^{230}Th, their solubility reduced by 1000 to account for their representation in the inventory, and ^{232}Th is ignored altogether on account of its negligible effect on release values. In every case, revisions favored overestimating released materials.

In CCA calculations, NUTS is exercised first using the five equivalent radioisotopes. If releases do not approach EPA limits, results are permitted to stand. If releases approach EPA limits, NUTS will be reexercised with actual inventories. The physical properties of the 5 "equivalent" radioisotopes are computed by the same ALGEBRA run that combines maximum concentrations due to dissolution and colloidal mobilization (see Section 4.5.1)

4.5.3 Temperature Dependency

At the user's option, NUTS can implement a temperature+ dependency for selected parameters. For example, solubility limit is considered either as (1) a constant or (2) it can be evaluated from data correlations that account for its thermal dependency in non-isothermal processes.

Additionally, molecular diffusivity varies with temperature, and the following derived formula is used to account for that dependency:

$$D^*_m(T) = D^*_{mref} \frac{T}{T_{ref}} \frac{\mu_{ref}}{\mu(T)}, \qquad \mathbf{4.90}$$

where

D^*_{mref}	is the molecular diffusion evaluated at the temperature T_{ref},
$D^*_m(T)$	is the molecular diffusion at temperature T,
μ_{ref}	is the reference viscosity at T_{ref},
$\mu(T)$	solvent viscosity at temperature T, and
T_{ref}	is the reference temperature.

Sorption is also a temperature-dependent process. The following is a zeroth-order formulation derived to account preliminarily for the temperature dependency of k_d. It can also be applied to the Freundlich and Langumir coefficients with fair accuracy.

$$k_{d2} = k_{d1} \ exp\left[\frac{\xi(T_2 - T_1)}{T_2 T_1}\right], \quad where\, \xi = \frac{\Delta H^o}{R}. \qquad \mathbf{4.91}$$

In the above formulation, k_{d1}, and k_{d2} are equilibrium partition constants at temperatures T_1 and T_2, respectively. H^o is the enthalpy of sorption at reference temperature T^o, and R is the universal gas constant.

4.5.4 Theoretical Foundations for the Gas-Transport Model Phase

In gas transport problems, it is essential to consider the chemical interaction between the flowing gas and the constituents of the porous medium through which the gas flows. That interaction or phase equilibrium can be classified as (1) gas/solid-phase equilibrium described by sorption-desorption processes, and (2) gas/liquid-phase equilibrium.

+ Temperatures should be provided to NUTS as an input by suitable thermal simulator.

The description of multiphase equilibrium in porous media is complex and requires a compositional simulator that strongly couples the transport with the flow equations. This kind of treatment arises because some mixtures like, for example VOCs + H2O, are observed under certain conditions of pressure, temperature and composition, as a single phase (liquid or vapor), a double-phase (vapor-liquid or liquid-liquid), or a triple-phase (vapor-liquid-liquid) fluid. To (a) describe such a system appropriately and (b) to predict the number and form of the phases in equilibrium, the transport by the liquid and gas phases should be calculated simultaneously. In that event, the solute would be partitioned according to its equilibrium constant between the phases in equilibrium.

Such a compositionally equilibrated system requires not only a detailed description of the pressure, temperature and initial composition fields in the computational domain, but also is very demanding in regard to computation time and resources. Because of data limitations, NUTS considers the porous medium and its liquid contents as a retardation factor for the constituents transported by the gas phase (Doctor et al., 1992). The phenomena described below are included in NUTS.

1. Solid-/gas-phase equilibrium is treated using the same linear, equilibrium, sorption isotherm described above for liquid-phase flows.

2. Liquid-/gas-phase equilibrium is modeled through gas solubility in the brine as a function of temperature. The equation that describes such equilibria is

$$K_{LG} = \frac{C_L}{C_G}, \qquad \textbf{4.92}$$

where C_L and C_G are the concentration of the solute in the liquid and gas phases, respectively. The value of the ratio on the right-hand side is represented as a linear function of temperature, namely:

$$\frac{C_L}{C_G} = A - BT, \qquad \textbf{4.93}$$

M

where A and B are the intercept and slope, respectively, of the empirical data. This linear representation is valid only for a limited range of temperatures and only at low pressures, where compositional dependency is usually weak.

Because migration in the liquid phase is much slower than in the gas phase, and because the gas- and the liquid-phase transports are decoupled in NUTS, liquid transport of a constituent transported abundantly by the gas phase will not be tracked. With the above mentioned assumptions, the transport in the gas phase of a dual-permeability system can be described by the following finite-difference equations:

Equation for Material Transport in the Fracture Component of the Gaseous Flow:

$$\begin{aligned}
&\Delta_x\left[\phi_f S_{Gf} K_{Gf} \Delta_x C_{nGf}\right] + \Delta_y\left[\phi_f S_{Gf} K_{Gf} \Delta_y C_{nGf}\right] + \Delta_z\left[\phi_f S_{Gf} K_{Gf} \Delta_z C_{nGf}\right] + \\
&\Delta_x q_{Gf} C_{nGf} + \Delta_y q_{Gf} C_{nGf} + \Delta_z q_{Gf} C_{nGf} + C^{*}_{nGf} Q_{Gf} - \tau_{Gm/f} C_{nGm/f} - \phi_m S_{Gm} K_{Gm/f}\left(C_{nGf} - C_{nGm}\right) = \\
&\frac{V_R}{\Delta t}\Delta_t\left[\phi_f S_{Gf} C_{nGf} + (1-\phi_t)\rho_s C_{nsf} + \phi_f\left(1 - S_{Gf}\right) K_{LG} C_{nGf}\right] + \qquad 4.94 \\
&V_R\left[\phi_f S_{Gf} C_{nGf} + (1-\phi_t)\rho_s C_{nsf} + \phi_f\left(1 - S_{Gf}\right) K_{LG} C_{nGf}\right]\lambda_n - \\
&V_R \sum_{j=1}^{J}\left\{\left\{\phi_f S_{Gf} C_{nGfpj} + (1-\phi_t)\rho_s C_{nsfpj} + \phi_f\left(1 - S_{Gf}\right) K_{LG} C_{nGfpj}\right\}\lambda_{npj}\right\} \quad n = 1,2,\ldots,N
\end{aligned}$$

Equation for Material Transport in the Matrix Component of the Gaseous Flow:

$$\begin{aligned}
&\Delta_x\left[\phi_m S_{Gm} K_{Gm} \Delta_x C_{nGm}\right] + \Delta_y\left[\phi_m S_{Gm} K_{Gm} \Delta_y C_{nGm}\right] + \Delta_z\left[\phi_m S_{Gm} K_{Gm} \Delta_z C_{nGm}\right] + \\
&\Delta_x q_{Gm} C_{nGm} + \Delta_y q_{Gm} C_{nGm} + \Delta_z q_{Gm} C_{nGm} + C^{*}_{nGm} Q_{Gm} + \tau_{Gm/f} C_{nGm/f} + \phi_m S_{Gm} K_{Gm/f}\left(C_{nGf} - C_{nGm}\right) = \\
&\frac{V_R}{\Delta t}\Delta_t\left[\phi_m S_{Gm} C_{nGm} + (1-\phi_t)\rho_s C_{nsm} + \phi_m\left(1 - S_{Gm}\right) K_{LG} C_{nGm}\right] + \qquad 4.95 \\
&V_R\left[\phi_m S_{Gm} C_{nGm} + (1-\phi_t)\rho_s C_{nsm} + \phi_m\left(1 - S_{Gm}\right) K_{LG} C_{nGm}\right]\lambda_n - \\
&V_R \sum_{j=1}^{J}\left\{\left\{\phi_m S_{Gm} C_{nGmpj} + (1-\phi_t)\rho_s C_{nsmpj} + \phi_m\left(1 - S_{Gm}\right) K_{LG} C_{nGmpj}\right\}\lambda_{npj}\right\} \quad n = 1,2,\ldots,N
\end{aligned}$$

where

C_{nG}	= solute concentration of component n in the gas phase[kg/m^3],
C^*_{nG}	= injected or produced solute concentration of component n [kg/m^3],
C_{ns}	= sorbate concentration of component n [kg/kg],
K	= dispersion coefficient [m^3/s],
K_{LG}	= gas-liquid equilibrium constant [dimensionless],
N	= total number of nuclides,
Q	= gas injection/production rate [m^3/s],
q	= gas interfacial volumetric rate [m^3/s],
S	= gas saturation, fraction,
t	= time [s],
V_R	= grid block volume [m^3],
Δ_t	= value at time n+1 - value at time n,
Δ_x	= centeral finite difference operator in x-direction,
Δ_y	= centeral finite difference operator in y-direction,
Δ_z	= centeral finite difference operator in z-direction,
λ	= decay constant [s^{-1}],

ϕ	= porosity [dimensionless],
ρ	= density [kg/m^3],
τ	= matrix /fracture transfer function [m^3/s],

Subscripts:

f	= fracture
m	= matrix
m/f	= matrix/fracture
n	= component number
pj	= parent number
s	= solid phase (rock)
t	= total (fracture + matrix)
G	= gas phase

5.0 INHERENT CAPABILITIES AND LIMITATIONS OF THE SOFTWARE

NUTS's inherent capabilities and limitations are listed below.

1. The flow and transport equations are decoupled, which is permitted only if solute concentrations are dilute. Decoupling does not affect fluid density, viscosity, or other related physical properties.

2. Local equilibrium is assumed to exist between interactive phases.

3. The radioisotopes transported by NUTS are assumed to be thermodynamically stable within their solubility limits. No gas phases or radioisotope-rich liquid phases are allowed. However, precipitation is possible and is calculated when and if solute concentrations exceed their solubility limits.

4. Dissolved radioisotopes are assumed to be in equilibrium with their precipitates. Precipitation is treated as a reversible process and is controlled by the value of solute concentration compared to the solubility limit, which defines the upper bound for dissolution.

5. Sorption is assumed to be controlled entirely by one of three equilibrium isotherms.

6. Dispersion is assumed to obey Fick's second law with dispersivities appropriate to an isotropic porous medium. Thus, the dispersivity tensor reduces to two numerical constants for the two principal flow directions.

7. The solubility of each isotope is evaluated as follows:

   Isotopic Solubility = Mole Fraction of that particular isotope x the elemental solubility

8. Gas-liquid equilibrium is assumed to obey the linear relationship represented by Equation 4.92, with temperature dependence given by Equation 4.93.

9. Because gases travel much faster than liquids, the liquid phase is regarded as a retarding agent to the constituents transported by the gas phase.

These and other inherent capabilities and limitations of the model are discussed item for item in Sections 4.0 and 6.0, and in the examples.

6.0 USER INTERACTIONS WITH THE SOFTWARE

6.1 Exercising NUTS Interactively

NUTS may be exercised either interactively or through command files. To exercise NUTS interactively, type "NUTS" at the VMS $ prompt and strike the carriage return. The title page and disclaimer will scroll by on screen. The program will then prompt the user with a series of questions including the names of a sequence of required input and output files. The sequence of queries is addressed in order in Section 6.2. The files themselves are discussed in detail in the remainder of this chapter, and illustrative examples are given in the appendices. If all NUTS's prompts are answered satisfactorily, the final "carriage return" will cause NUTS to exercise. If the run is successful, a "normal completion" notice will appear on screen. If not, an error message will result.

NUTS may be compiled on PC, VAX, and UNIX systems with minimal effort. However, for the WIPP CCA PA, NUTS will operate exclusively in a cluster of Alpha-VAX microcomputers. Consequently, the other applications will not be discussed explicitly in this manual. However, to gain the flexibility to run on different platforms, NUTS was intentionally designed to run in two slightly different configurations. The first is called the "stand-alone" configuration and is written NUTS-SA. The second, designed specifically for the WIPP's Compliance Assessment Methodology Controller (CAMCON), is called CAMCON configuration and is written NUTS-CC. The main differences between the two are (1) the types of libraries with which each compiles, (2) the flexibilities associated with the two sets of input/output capabilities, and (3) that NUTS-SA and NUTS-CC have two *different* executables. NUTS-CC will be used exclusively for the WIPP CCA PA. However, because NUTS-CC's operational procedures are a subset of NUTS-SA, we will begin with a discussion of the Stand-Alone input/output features. That discussion will be followed with the requirements for and limitations of NUTS-CC.

6.2 User-Interactive Options Afforded by NUTS-SA

NUTS requires the following three types of Input/Output:

1. A User-Interactive Input Sequence
2. A Parameter Statement
3. Input/Output Files

These three types of input/output* are reviewed in the three subsections that follow.

6.3 User-Interactive Input Sequence

A limited number of control flags are available for user specification during the course of executing the program. These controls are part of the input and mainly specify the following information:

* The units adopted in NUTS are SI units unless specified otherwise.

a) Names, sources, and types of the I/O files (ASCII, binary, etc.).
b) The type of porous media to be modeled (single-, or dual-porosity and/or permeability).
c) The number of phases in the transporting medium (usually 1), and the type of phase (gas or liquid).
d) The type of the compiler and terminal (for QA purposes).

The following is a description of user-interactive input*:

6.3.1 Is This a DEBUG Run? (Y/N) {N}

Accepts Y, N, y, or n. Press the carriage-return key to select the default option.

6.3.2 The Default Input-File Name is NUTS.IN

Do You Want To Use This Name (Y/N)--> {NUTS.IN}

Accepts Y, N, y, & n. Press the carriage-return key to select the default option.

a. If the answer is "Y" or "y" the program will proceed to the next question.

b. If the answer is "N" or "n" , the program will prompt the following:

Enter NUTS Input File Name

The user may specify an input-file name up to 80 characters in length and strike the carriage-return key, wherein the sequence of questions will proceed.

If instead, the user strikes the "Return key" the following will appear on screen:

The Default Input File Name NUTS.IN
Is Assigned To The Input File
Do You Like To Change It (Y/N)

Accepts Y, N, y, or n. Press the carriage-return key to select the default option.
If the answer is "Y" , the program will start again from "b". Otherwise it will proceed.

If an invalid input filename is entered, the program will deliver an error message and terminate.

6.3.3 The Default Output File Name is NUTS.OUT

Do You Want To Use This Name (Y/N)--> {NUTS.OUT}

Accepts Y, N, y, or n. Press the carriage-return key to select the default option.
Depending on the answer, the program will proceed exactly as in (6.3.2). If an original filename is assigned, be sure to use a character*80 name.

* Quantities listed within braces are the default-option replies to program queries. They are selected by pressing the carriage return.

6.3.4 Read the material map from BRAGFLO input file (Y/N)? {Y}

Accepts Y, N, y, or n. Press the carriage-return key to select the default option.

a. If the answer is "Y" or "y", NUTS will read the material map from BRAGFLO's input file**.

b. If the answer is "N" or "n", NUTS will expect to read the material map from its own ASCII input file.

6.3.5 Is This a Test Run? (Y/N) {N}

Accepts Y, N, y, or n. Press the carriage-return key to select the default option.

a. If the answer is "Y" or "y", NUTS will ask the following:
Enter name of TEST file
The user may specify an original filename up to 80 characters in length.

b. If the answer is "N" or "n", the program will prompt the following:
Enter BRAGFLO input binary/CDB file name.
The user may specify an original filename up to 80 characters in length.
NUTS will then inquire:
Will BRAGFLO data be read from CDB or binary+ ?
Enter BIN for binary or CDB for CDB {BIN}
Accepts BIN, bin, Bin, CDB, cdb, or Cdb. Press the carriage-return key to select the default option.

If an invalid filename is entered (whether a test-file or BRAGFLO-file name), the program will issue an error message and abort.

6.3.6 Enter Type of Output File {ASC}

The following output files are allowed:

1. **ASC for ASCII file**
2. **BIN for binary file**
3. **CDB for CAMCON database**
4. **ASC-BIN for ASCII and binary files**
5. **ASC-CDB for ASCII and CDB files**
6. **BIN-CDB for binary and CDB files**
7. **ASC-BIN-CDB for ASCII, binary, and CDB files**

M

Accept ASC, BIN, CDB, ASC-BIN, ASC-CDB, BIN-CDB, or ASC-BIN-CDB†.

The case of the characters (either upper or lower case) and the sequence of the combination of the file types is not restricted; i.e., ASC-BIN, BIN-ASC, asc-bin, or bin-

** NUTS expects to open a BRAGFLO ASCII input file that has the same name as BRAGFLO's binary file, but has an INP extension

+ NUTS has the ability to read BRAGFLO binary outputs before and after post-processing into CDB

† Because of some technical limitation in CAMCON system, we recommend using CDB option in NUTS_CC version of the code.

asc will all be allowed. However, the combination between upper and lower cases is not allowed (Example: ASC-bin is not allowed).

If "BIN" is among the options required by the user, NUTS will prompt:

Enter Binary Output Format

The following types are allowed

1. Enter BRAG OR brag FOR BRAGFLO Output Format

2. Enter NUTS OR nuts FOR NUTS Output Format

Accepts BRAG, brag, NUTS, or nuts

6.3.7 Enter Type of Porous Medium {MATRIX}

The following types are allowed:

1. **"FRACTURE, Fracture, fracture, FRACT, fract, Fract, F, or f " for fracture calculations.**
2. **"MATRIX, Matrix, matrix, mat, MAT, Mat, M, or m" for matrix calculations.**
3. **"DUAL-POROSITY, dual-porosity, Dual-porosity, DP, Dp, dp, D-P, d-p, or D-p" for dual-porosity.**
4. **" DUAL-PERMEABILITY, dual-permeability, DPM, dpm, D-PM, d-pm, Dpm, or D-pm" for dual-permeability.**

Accepts FRACTURE, Fracture, fracture, FRACT, fract, Fract, F, f; MATRIX, Matrix, matrix, mat, MAT, Mat, M, m; DUAL-POROSITY, dual-porosity, Dual-porosity, DP, Dp, dp, D-P, d-p, or D-p; DUAL-PERMEABILITY, dual-permeability, DPM, dpm, D-PM, d-pm, Dpm, or D-pm. Press the carriage-return key to select the default option.

6.3.8 Enter Number of Phases {1}

Accepts an integer ≥ 1

6.3.9 Enter the Type of Phase (Liquid/Gas) {L}

The following types are allowed:

1. **LIQUID, liquid, Liquid, L, l, LIQ, liq for liquid phase**
2. **GAS, gas, Gas, G, g, GS, gs for gas phase**

Accepts LIQUID, liquid, Liquid, L, l, LIQ, liq; GAS, gas, Gas, G, g, GS, or gs. Press the carriage-return key to select the default option.

6.3.10 Enter the Type of Compiler *(character*32)*

The following are allowed:

1. **Enter VAX for VAX SYSTEM**
2. **Enter UNIX for UNIX SYSTEM**
3. **Enter PC for PC LAHEY SYSTEM**

Accepts VAX, vax, UNIX, unix, PC, or pc.

6.3.11 Enter the name of the computer system
Ex: CRAY, GATEWAY2000, ALPHA, etc.

6.4 NUTS's Parameter Statement

NUTS uses an INCLUDE file (NUT_PARAM.INC) that contains a single **PARAMETER** statement to set the dimensions of different kinds of arrays used in the computations. Each parameter value is allotted space based on the needs of the WIPP's PA calculations. For larger problems, the parameters of NUT_PARAM.INC can be easily adjusted. As in any parameter statement, the size of the array may be greater (but not less) than that required by the problem. An example of a parameter statement used for Compliance Certification Application (CCA) calculations is as follows:

```
C
C      Note+:
C      SET MBW >= {[(MAX(MX,MY)+1)* # OF PHASES* # OF                   CONTINUUM - #
OF CONTINUUM]*2 + 1} * 2

       INTEGER
     +    MX, MY, MZ, NB, NS,
     +    NC, NP, NDR, MBW, NFVARA, NMVARA,
     +    NFVARB, NMVARB, NFVARC, NMVARC,
     +    NVARTIT, NVPR, MXHIV, MVHIV,
     +    MGVAR, NMATM, MMATTIME, MWASTE,
     +    IHNTVARB, IHGNTVARB,
     +    MXPOINT, MYPOINT
       PARAMETER
     +    (
     +    MX         = 200,
     +    MY         = 40,
     +    MZ         = 1,
     +    NB         = 8000,
     +    NS         = 2,
     +    NC         = 9,
     +    NP         = NC,
     +    NDR        = 2,
     +    MBW        = 410,
     +    NFVARA     = 14,
     +    NMVARA     = 14,
     +    NFVARB     = 14,
     +    NMVARB     = 14,
     +    NFVARC     = 14,
     +    NMVARC     = 14,
     +    NVARTIT    = 28,
     +    NVPR                = 300,
     +    MXHIV      = 540,
     +    MVHIV      = MXHIV,
     +    MGVAR      = 10,
     +    NMATM      = 80,
     +    MMATTIME   = 50,
     +    MWASTE     = 50,
     +    IHNTVARB   = 28*NC,
     +    IHGNTVARB  = 10*NC,
     +
     +    MXPOINT    = 30,
     +    MYPOINT    = 30
```

+ This statement calculates a size of numerical matrix suitable for two-point upstream weighting option. For other options use half this size

+)

The above parameter statement variables are declared to satisfy the maximum memory requirement anticipated in the CCA calculations. In general, numbers smaller than 1 are not allowed in these declarations. The definition of the above-mentioned parameters is given in Table 2 for the variables and Table 3 for the constants+.

+ These parameters are used in the parameter statement and should not be changed.

Table 2. Variables used in NUT_PARAM.INC

Parameter	Description
MX	Maximum number of grid blocks in x-direction.
MY	Maximum number of grid blocks in y-direction.
MZ	Maximum number of grid blocks in z-direction.
NB	Maximum total number of the grid blocks (MX*MY*MZ)
NS	Maximum number of radioactive sites.
NC	Maximum number of isotopes or constituents.
NP	Maximum number of parents for each isotopes.
NDR	Maximum number of rock density ranges.
MBW	Maximum numerical matrix band width.
NVPR*	Number of element array maps from BRAGFLO.
MXHIV*	Maximum total number of history variables allowed in BRAGFLO.
MVHIV*	Maximum number of history variables per output distribution in BRAGFLO.
MGVAR*	Number of global variables printed out in an ASCII or binary BRAGFLO output file.
NMATM	Maximum number of materials specified in BRAGFLO
MMATTIME	Maximum number of times material map is specified in BRAGFLO
MWASTE	Maximum number of waste regions.
IHNTVARB**	Maximum total number of history variables to be printed in BRAGFLO style binary file.
IHGNTVARB**	Maximum total number of global variables to be printed in BRAGFLO style binary file.
MXPOINT	Maximum number of rows allowed in an interpolation table
MYPOINT	Maximum number of columns allowed in an interpolation table

* These parameters are related to BRAGFLO output and relevant only when BRAGFLO binary output is dealt with directly by NUTS (not through the CAMCON).

** These parameters are relevant only when NUTS binary output is to be post-processed with BRAGFLO binary output

Table 3. Constants used in NUT_PARAM.INC

Parameter	Description
NFVARA	Number of fracture arrays to be output in an ASCII file = 14.
NMVARA	Number of matrix arrays to be output in an ASCII file = 14.
NFVARB	Number of fracture arrays to be output in a binary file = 14.
NMVARB	Number of matrix arrays to be output in a binary file = 14.
NFVARC	Number of fracture arrays to be output in a CDB file = 14.
NMVARC	Number of matrix arrays to be output in a CDB file = 14.
NVARTIT	Number of descriptive variables = 28.

6.5 Input/Output Files Used by NUTS-SA

NUTS is capable of accommodating different combinations of input/output files in each run, depending upon the user's specifications and requirements. We will classify these files as input, output, and debug* files. The convention generally used in naming NUTS's input/output files is to form a root name by concatenating the names of the flux-field input file and the radioisotope input file, and then adding any of ASC, BIN, or DBG as extensions. A subset or the entire list of these files may be opened in a single application. The files are briefly listed below, and then described in considerable detail in the subsections that follow.

M

1. An ASCII radioisotope input file

2. An ASCII flux-field input file

3. A binary flux-field input file (output from BRAGFLO) if not provided as an ASCII

4. Binary output files
   a. BRAGFLO-type binary output file (to be post-processed by BRAGFLO's post-processor)
   b. NUTS-type binary output file

* Strictly speaking, debug files are text output files. However, we prefer to classify them separately.

5. ASCII output file
    a. Extensive output file
    b. Specific ranges output file

6. Debug ASCII output files

An expanded description of NUTS's input and output files follows.

6.5.1 ASCII Radioisotope Input File

This file consists of pieces of information provided by different procedures and controlled by the flags listed during NUTS's initial interrogation (and also in the user interactive input). The input can be categorized as:

1) General input the user will encounter in all NUTS input files (isotope-related input).
2) Input related to the type of the porous media encountered (Continuum-related input).
3) Material-properties-map input.

The subsections that follow, numbered **6.5.1.n** , where **n** is various combinations of letters and numbers, are a sequential description of the ASCII radioisotope input file. A sample ASCII input file is given in Appendix A.

6.5.1.A Program Controls and Flags

The very first part of the ASCII-radioisotope (A/R) input file is mainly flags that control both subsequent inputs and the computation itself. These flags and most of the description that follows will be presented in a form resembling computer language, to simplify the readability of the manual.

6.5.1.A1 Site Description and Flags

Line 1. Descriptor: NUTS title of the run. *Accepts a character string up to 100 characters as a descriptive line.*

Line 2. NUTS_TITLE.
NUTS_TITLE: This is a descriptive line for the run. The title is saved and concatenated with BRAGFLO's titles when they are printed out, which are simple clues for the user as to which NUTS and BRAGFLO runs were considered. *Accepts a character string up to 100 characters in length.*

Line 3. Descriptor: Number of sites and materials. *Accepts a character string up to 100 characters in length as a descriptive line.*

Line 4. NSITES, NMAT_INPUT

NSITES: is the number of contamination sites used in the run. Any grid block or combination of grid blocks can be a site. Thus, this is a useful representation if similar constituents from different grid blocks are to be tracked or the overlap of contamination from different places is to be assessed. *Accepts an integer > 0 , but < the total number of the grid blocks.*

NMAT_INPUT: is the number of porous materials encountered in the domain. It describes the number of materials having different properties such as porosity, permeability, density, etc. This number is compared in NUTS with the number read from BRAGFLO ASCII input file while reading the material map. If the two numbers are different, NUTS will output an aborting message and stop. *Accepts an integer > 0 , but < the total number of the grid blocks.*

DO I = 1, NSITES

Line 5. SITE_NAME(I), NCOMPONENT(I)

SITE_NAME: represents the name of the site to be modeled. *Accepts a 20 character string.*

NCOMPONENT: is the number of radioactive components to be modeled in each site. Similar components from different sites or parents are considered to be distinct, i.e., if ^{238}U exists at two sites, it will be counted as a separate constituent at each site. NCOMPONENT is the total number of all components that pre-exist, plus any that are anticipated to appear as a result of the decay process. *Accepts an integer > 0.*

END DO

Line 6. Descriptor: 1. Site name, 2. Component, Daughter, Parent, and Group names. *Accepts a character string up to 100 characters in length as a descriptive line.*

M = 0

DO I = 1, NSITES

Line 7. SNAME(I)

SNAME: a dummy variable for the SITE_NAME. SITE_NAME is renamed as a precaution in case of an input error, which, if it occurs, will not affect the stored name. *Accepts a 20-character string.*

DO J = 1, NCOMPONENT(I)

M = M + 1

Line 8. COMPONENT_NAME(I,M), DAUGHTER_NAME(M), PARENT_NAME(M), GROUP_NAME(M)

COMPONENT_NAME:	the name of the radioactive isotope[+]. *Accepts a 20-character string.*
DAUGHTER_NAME :	for isotopes that decay into a daughter, this variable refers to the name of the next isotope in the decay chain. For constituents that do not undergo decay or whose half-lives are greater than the simulation time, the character string 'NONE' in upper case should be used. *Accepts a 20-character string.*
PARENT_NAME:	the name of the preceding isotope in the decay chain. If the component has no parent, the character string 'NONE' in upper case should be used. *Accepts a 20-character string.*
GROUP_NAME:	Refers to the element name of the isotope. *Accepts a 20-character string.*

M

END DO
END DO

Note: In the Do Loop of Line 8, it is very important to strictly follow the chain from the top down to the last member. The branches in diverging or converging chains should be treated as a separate chain. This may require an adjustment in the decay constant, inventory, and/or the name of the component at which the convergence or the divergence occurs. It is also necessary in the above entries to be consistent in the spelling and the letter case of the component, daughter, parent and group names throughout the rest of the input, except "NONE" that should be used in upper case.

Line 9. Descriptor: 1. Number of element, 2. element name, temperature dependency of solubility and table look-up solubility. *Accepts a character string up to 100 characters as a descriptive line.*

Line 10. NOELEMENT

NOELEMENT: Number of elements. *Accepts an integer >0 , but < the total number of isotopes.*

DO I = 1, NOELEMENT

Line 11. ELEMENT_NAME(I), ELTEMP_SOLB(I), SOLB_TABLE(I)

ELEMENT_NAME:	the element's name. It should be identical to GROUP_NAME. *Accepts a 20-character string.*
ELTEMP_SOLB:	a logical flag that indicates the temperature dependency of the solubility of the element. *Accepts T or F.*
SOLB_TABLE:	a logical flag to indicate whether or not the solubility of the element will be provided by a table. *Accepts T or F.*

[+] Throughout this manual, isotope, radioactive isotope, component, and constituent are used interchangeably. They all refer to the chemical species to be modeled and only their properties will define whether that species is radioactive or not.

END DO

Note: A classical definition of the element is a substance that can not be decomposed by chemical means into simpler substances. However, in this manual, element is used to refer to the ionic state of all compounds of an element in more than one oxidation state that compete on the same solubility.

Line 12. Descriptor: Colloidal transport (T/F). *Accepts a character string up to 100 characters as a descriptive line.*

Line 13. COLLOID

COLLOID: a logical flag to indicate whether colloidal transport is intended. *Accepts a T or F.*

If the answer above is T, then enter Line 14 to Line 18.

Line 14. Descriptor:
1. Number of colloids
2. location of the colloid in Line 8.
3. location of the dissolved species of the same kind.
4. preferential solubility.

Accepts a character string up to 100 characters as a descriptive line.

Line 15. NCOLLOID

NCOLLOID: Number of colloids to be transported. *Accepts an integer >0 , but < the total number of isotopes.*

Line 16. (LOC_COLLOID(I), I = 1, NCOLLOID)

LOC_COLLOID: Location of the colloid in the input sequence of line 8 of Section 6.5.1.A1. *Accepts an integer >0 , but < the total number of isotopes.*

Line 17. (IDIS(LOC_COLLOID(I)), I = 1, NCOLLOID)

IDIS: Location of the dissolved species competing with the colloid, in going to solution, in the input sequence of line 8 of Section 6.5.1.A1. *Accepts an integer* >0 , but < the total number *of isotopes.*

Line 18. (PREF_SOLB(LOC_COLLOID(I)), I = 1, NCOLLOID)

PREF_SOLB: Logical flag that identifies whether the colloid has preference to go to solution. Accepts a T or F.

Note: Colloid transport in NUTS is used in a simple form. Even though the colloids are treated like the none-colloid species in having a distinguished name and properties, transport with colloid has two different treatments. The first treatment is the ability to scale the velocity of the brine by a scaling factor. This will adjust the velocity field for the whole transport process. The second treatment is the preferential solubility for the colloids. The preferential solubility is given to the colloid only at the inventory limit. Therefore, if a

dissolved and a colloid ^{238}U starts with some part in solution and some precipitate, at the inventory limit, the colloid will use up its own precipitate first, then the precipitate of the dissolved part of ^{238}U followed by the non-colloid dissolved ^{238}U and up to the solubility limit of the colloid each time.

Line 19. Descriptor: pH required (Y/N). *Accepts a character string up to 100 characters as a descriptive line.*

Line 20. PHREQ

PHREQ: Flag related to brine-pH information. It is associated with the solubility calculation. pH will be required if correlations are to be used to determine the solubility of an element. *Accepts Y, y, N, or n.*

Line 21. Descriptor: Order of the numerical method

Line 22. MORDER:

MORDER: Order of the method. The following options are allowed:
MORDER =1, for one-point upstream winding.
MORDER =2, for two-point upstream winding.
MORDER =3, for split-operator two-point upstream winding.
MORDER =4, for mid-point weighting.
Accepts an integer >0 , but ≤ 4.

Line 23. Descriptor: Degree of implicitness.

Line 24. BETA2:

BETA2: A multiplier used in the differential equation to weight the contribution from the explicit and implicit parts of the transport equation. The following values of BETA2 are of special interest:
BETA = 1 , for fully implicit.
BETA = 0 , for fully explicit.
BETA = 1/2 , for Cranck-Niclson.
Accepts a real number ≥ 0, but ≤ 1.

6.5.1.A2 Single-Porosity Fracture Controls and Flags

These controls are applied if a single-porosity, fracture medium is to be modeled:

Fracture Sorption Controls

Line 1. Descriptor: Is fracture sorption required (Y/N)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 2. ADSTYPEF

ADSTYPEF: This character identifies whether or not sorption is required.
Accepts Y, y, N, or n.
If the above answer is 'Y' or 'y', then input lines 3 and 4.

Line 3. FRAC_ISOTHERM

FRAC_ISOTHERM: Type of sorption isotherm. The following are considered by NUTS:
Linear isotherm. *Accepts L or l.*
Freundlich isotherm. *Accepts F, or f.*
Langumir isotherm. *Accepts LA, la, or La.*

DO I = 1, NUCLIDE ' NUCLIDE = Total number of constituents'

Line 4.NAME(I), FSORPTION(I), FADSTEMPDEP(I)

NAME: Dummy variable for the name of the component.
Accepts a 20-character string.

FSORPTION: Character to identify whether the component is sorbable.
Accepts ADSORP for sorbable constituents, or NON_ADSORP for non-sorbable constituents.

FADSTEMPDEP: Logical flag for temperature dependency of the fracture sorption isotherm. *Accepts T, or F.*

END DO

Fracture Dispersion Controls

Line 5.Descriptor: Does dispersion take place within the fracture (Y/N)
Accepts a character string up to 100 characters in length as a descriptive line

Line 6.FDISPREQ

FDISPREQ: Fracture-dispersion flag.
Accepts Y, y, N, or n.

Line 7. Descriptor: Do you have symmetrical dispersion in the fracture (F/T) and will data be read from NUTS ASCII (F/T).
Accepts a character string up to 100 characters in length as a descriptive line. The input on Line 8 is required only when the answer to line 6 is Y or y.

Line 8.FSYMDISP, FDSPNUT INPUT

FSYMDISP: Logical flag to specify whether mechanical dispersion is symmetrical or only downstream (default = downstream).
Accepts T or F.

FDSPNUTINPUT: Logical flag to specify whether the following dispersion data will be read from NUTS ASCII input (if the flag is true) and from the CDB (if the flag is false) (default = NUTS input).

Accepts T or F.

Fracture Source Controls

Line 9. Descriptor: Does injection/production occur in the fracture (Y/N)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 10. FSTATUSINJ
FSTATUSINJ: Character to inform NUTS injection/production is to be activated in the fracture. *Accepts Y, y, N, or n.*

Line 11. Descriptor: Do Dirichlet boundary conditions apply in the fracture (T/F)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 12. FDIRICHLET
FDIRICHLET: Logical flag to inform NUTS that Dirichlet boundary condition will be considered in some computational nodes.
Accepts T or F

Note: Dirichlet boundary conditions in NUTS are implemented by using the penalty method+. In this method, the main diagonal of the linear equation of the grid block is multiplied by a large number, β and the right-hand side of the same equation is replaced by β*C_{dir}, where Cdir is Dirichlet boundary condition concentration. As long as the multiplier is a very large number, the concentration of the grid block will be maintained at Dirichlet condition.

Fracture Initial Concentration Controls

Line 13. Descriptor: Is manual initialization required in the fracture (T/F)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 14. FMAN_CONC_INIT
FMAN_CONC_INIT: Logical flag to inform NUTS that the concentration in the fracture is initialized manually.
Accepts T or F.

Fracture Printed Element Variables Controls

This module specifies the format (ASCII, BINARY or CAMCON DATA BASE [CDB]) in which fracture concentration, precipitation, etc. to be represented in the output files.

+ Becker, E.B., Cary, G.F., and J.T. Oden (1981): *Finite Elements: An Introduction*, Volume 1, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

If an ASCII file is to be output, then:

Line 15. Descriptor: Variables to be printed as an ASCII file
Accepts a character string up to 100 characters in length as a descriptive line.

Line 16. (IPRNTFA(I), I = 1, NFVARA)

IPRNTFA: This integer array has fourteen flags for the fracture-element variables to be printed as an ASCII file.
Choose 1 to print the variable and 0 to omit the variable from printing.
This flag will activate or deactivate printing of the following variables:

IPRNTFA(1):	Fracture total mass of dissolved constituents in the brine in each grid block, kg
IPRNTFA(2):	Fracture total mass of precipitated constituents in each grid block, kg
IPRNTFA(3):	Fracture total mass of dissolved, precipitated and sorbed constituents in each grid block, kg
IPRNTFA(4):	Fracture total dissolved curies of all constituents in each grid block, curies
IPRNTFA(5):	Fracture total precipitated curies of all constituents in each grid block, curies
IPRNTFA(6):	Fracture total dissolved, precipitated, and sorbed curies of all constituents in each grid block, curies
IPRNTFA(7):	Fracture volumetric concentration of the dissolved isotope, kg/(m^3 brine).
IPRNTFA(8):	Fracture dissolved mass in the brine of a certain constituent , kg.
IPRNTFA(9):	Fracture precipitated mass of a certain constituent, kg.
IPRNTFA(10):	Fracture soil base concentration of the isotope, mg/(kg soil).
IPRNTFA(11):	Fracture curies of volumetric concentration of a certain constituent, curies/(m^3 brine).
IPRNTFA(12):	Fracture curies of dissolved mass in the brine of a certain constituent, curies.
IPRNTFA(13):	Fracture curies of precipitated mass of a certain constituent, curies.
IPRNTFA(14):	Fracture curies of dissolved, precipitated and sorbed mass of a certain constituent, curies.

Line 17. Descriptor: Spatial range (s) of data to be printed in an ASCII file for specific variable

Accepts a character string up to 100 characters in length as a descriptive line

Line 18. ASC_FPRINT_RANGE

ASC_FPRINT_RANGE: Allows a range(s) of data to be output in an ASCII file.
Accepts Y, y, N, or n.

If a BINARY* file is to be output, then:

Line 19. Descriptor: Variables to be printed to a BINARY file
Accepts a character string up to 100 character in length as a descriptive line.

Line 20. (IPRNTFB(I), I = 1, NFVARB)

IPRNTFB: integer array similar to IPRNTFA(I) has fourteen flags for the element fracture variables to be printed in a binary file.
Choose 1 to print the variable, and 0 to omit printing.

If a spatial range(s) is to be printed (line 18 is "Y" or 'y'), then:

Line 21. Descriptor: Ranges to be printed in an ASCII file
Accepts a character string up to 100 characters in length as a descriptive line.

Line 22. NO_ASC_FRAC_RANGES:

NO_ASC_FRAC_RANGES: Number of ranges to be printed
Accepts an integer > 0 , but < the total number of grid blocks.

DO I = 1, NO_ASC_FRAC_RANGES

Line 23 IARANGE(I), MFASTRTI(I), MFASTRTJ(I), MFASTRTK(I), MFAENDI(I), MFAENDJ(I), and MFAENDK(I)

IARANGE:	Range number *Accepts an integer > 0, but* < NO_ASC_FRAC_RANGES
MFASTRTI,J,K	I,J,K indices of the grid block starting the range *Accepts an integer > 0 and I < NX, J < NY, K < NZ.*
MFAENDI,J,K	I,J,K indices of the grid block ending the range *Accepts an integer > 0 and I < NX, J < NY, K < NZ.*

Line 24. (IPRNSFA(I,J), J = 1, NFVARA)

IPRNSFA: integer array similar to IPRNTFA

END DO

* Lines 19 and 20 are required for the CDB output in the NUT-CC version of the code.

6.5.1.A3 Single-Porosity Matrix Controls and Flags

These controls are applied if a single-porosity matrix is to be modeled.

Matrix Sorption

M

Line 1. Descriptor: Is matrix sorption required (Y/N)
Accepts a character string up to 100 characters in length as a descriptive line

Line 2.ADSTYPEM
ADSTYPEM: This character activates or deactivates sorption in the matrix.
Accepts Y, y, N, or n. If the answer is 'Y' or 'y' then:

Line 3.MAT_ISOTHERM
MAT_ISOTHERM: one of three types of matrix sorption isotherm, namely:
Linear isotherm. *Accepts L or l.*
Freundlich isotherm. *Accepts F, or f.*
Langumir isotherm. *Accepts LA, la, or La.*

DO I = 1, NUCLIDE

Line 4.NAME(I), MSORPTION(I), MADSTEMPDEP(I)
NAME: Dummy variable for the name of the component.
Accepts a 20-character string.
MSORPTION: Character to identify whether the component is sorbable.
Accepts ADSORP for sorbable or NON_ADSORP for non-sorbable constituents.
MADSTEMPDEP: logical flag for temperature dependency of the matrix sorption isotherm.
Accepts T or F.

END DO

Matrix Dispersion Controls

Line 5.Descriptor: Do you have dispersion in the matrix (Y/N)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 6.MDISPREQ
MDISPREQ: Matrix-dispersion flag.
Accepts 'Y' or 'y' to consider, and 'N' or 'n' to omit dispersion in the matrix.

Line 7. Descriptor: Do you have symmetric dispersion in the matrix (F/T), and will data be read from NUTS ASCII (F/T).
Accepts a character string up to 100 characters in length as a descriptive line. The input on line 8 is required only when the answer to line 6 is Y or y.

Line 8. MSYMDISP, MDSPNUTINPUT

MSYMDISP: Logical flag to specify whether mechanical dispersion is symmetric or only downstream (default = downstream). *Accepts T or F.*

MDSPNUTINPUT: Logical flag to specify whether the following dispersion data will be read from NUTS ASCII input (if the flag is true) and from the CDB (if the flag is false) (default = NUTS input). *Accepts T or F.*

Matrix Source Controls

M

Line 9. Descriptor: Is injection/production present in the matrix (Y/N)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 10. MSTATUSINJ

MSTATUSINJ: Informs NUTS whether or not injection/production is present in the matrix.
Accepts Y, y, N, or n.

Line 11. Descriptor: Are Dirichlet boundary conditions present in the matrix (T/F)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 12. MDIRICHLET

MDIRICHLET: Logical flag to inform NUTS that Dirichlet boundary condition will be considered at some computational nodes.
Accepts T or F.

Matrix Initial Concentration Controls

Line 13. Descriptor: Is manual initialization present in the matrix (T/F)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 14. MMAN_CONC_INIT

MMAN_CONC_INIT: Logical flag informing NUTS that the concentration in the matrix is initialized manually.
Accepts T or F.

Matrix Initial Concentration from NUTS Undisturbed CDB

In the CCA calculations, two intrusion times are considered in BRAGFLO: 350 and 1000 years. In probabilistic evaluation of the intrusion times, times other than 350 and 1000 years need to be considered for the transport purposes. This step has been done by interpolating the flow fields results from BRAGFLO for 350 years to perform the transport with an intrusion that occurs at 100 years. Similarly, the flow fields of the 1000 years intrusion are extrapolated to intrusions that take place at 3000, 5000, 7000, and 9000 years. To account for the fact that the period before the intrusion is undisturbed, the output CDB from undisturbed NUTS transport calculations is used to initialize the calculation at that specific time of intrusion.

Line 15. Descriptor: Concentration is initialized from NUTS undisturbed CDB (T/F)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 16. CONC_CDB_INITIALIZATION

CONC_CDB_INITIALIZATION: Logical flag informing NUTS that the concentration in the matrix is initialized from the undisturbed CDB.
Accepts T or F.

Matrix Printed Element Variable Controls

This module specifies the output-file format (ASCII, BINARY or CAMCON DATA BASE [CDB]) of results for matrix concentration, precipitation, etc.

If an ASCII file is to be output, then:

Line 17. Descriptor: Variables to be printed in an ASCII file
Accepts a character string up to 100 characters in length as a descriptive line.

Line 18. (IPRNTMA(I), I = 1, NMVARA)

IPRNTMA: An integer array containing fourteen flags for the matrix-element variables to be printed in an ASCII file.
Choose 1 to print and 0 not to print the variable in question.
IPRNTMA(I) refers to following variables:

IPRNTMA(1):	Matrix total mass of dissolved constituents in the brine in each grid block, kg
IPRNTMA(2):	Matrix total mass of precipitated constituents in each grid block, kg
IPRNTMA(3):	Matrix total mass of dissolved, precipitated and sorbed constituents in each grid block, kg
IPRNTMA(4):	Matrix total dissolved curies of all constituents in each grid block, curies
IPRNTMA(5):	Matrix total precipitated curies of all constituents in each grid block, curies
IPRNTMA(6):	Matrix total dissolved, precipitated, and sorbed curies of all constituents in each grid block, curies
IPRNTMA(7):	Matrix volumetric concentration of the dissolved isotope, kg/(m^3 brine).
IPRNTMA(8):	Matrix dissolved mass in the brine of a certain constituent, kg.

IPRNTMA(9): Matrix precipitated mass of a certain constituent, kg.
IPRNTMA(10): Matrix soil base concentration of the isotope, mg/(kg soil).
IPRNTMA(11): Matrix curies of volumetric concentration of a certain constituent, curies/(m^3 brine).
IPRNTMA(12): Matrix curies of dissolved mass in the brine of a certain constituent, curies.
IPRNTMA(13): Matrix curies of precipitated mass of a certain constituent, curies.
IPRNTMA(14): Matrix curies of dissolved, precipitated and sorbed mass of a certain constituent, curies.

Line 19. Descriptor: Spatial range (s) of data to be printed in an ASCII file for specific variables
Accepts a character string up to 100 characters in length as a descriptive line.

Line 20. ASC_MPRINT_RANGE
ASC_MPRINT_RANGE: This character *accepts Y, y, N, or n* answers for the range(s) of data to be output in an ASCII file.

If a BINARY* file is to be output, then:

Line 21. Descriptor: Matrix variables to be printed in a BINARY file
Accepts a character string up to 100 characters in length as a descriptive line.

Line 22. (IPRNTMB(I), I = 1, NMVARB)
IPRNTMB: A fourteen-flag integer array similar to IPRNTMA(I)
Choose 1 to print, or 0 to omit the element matrix variables in a binary file.

If spatial range(s) is to be printed (Line 20 is "Y" or "y'), then:

Line 23. Descriptor: Ranges to be printed in an ASCII file
Accepts a character string up to 100 characters in length as a descriptive line.

Line 24. NO_ASC_MATRIX_RANGES
NO_ASC_MATRIX_RANGES: Number of ranges to be printed
Accepts an integer > 0 , but < the total number of grid blocks.

* Lines 21 and 22 are required for the CDB output in the NUT-CC version of the code.

DO I = 1, NO_ASC_MATRIX_RANGES

Line 25. IARANGE(I) ,MASTRTI(I) ,MASTRTJ(I) ,MASTRTK(I), MAENDI(I), MAENDJ(I), MAENDK(I)

IARANGE:	Range number *Accepts an integer > 0, but <* NO_ASC_MATRIX_RANGES
MASTRTI,J,KI,J,K:	Indices of the grid block starting the range *Accepts an integer > 0 and I < NX, J < NY, K < NZ*
MAENDI,J,K:	I,J,K indices of the grid block ending the range *Accepts an integer > 0 and I > NX, J > NY, K > NZ*

Line 26. (IPRNSMA(I,J), J = 1, NMVARA)

IPRNSMA: an integer array similar to IPRNTMA

END DO

6.5.1.A4 Dual-Porosity/Dual-Permeability Controls and Flags

If a dual-porosity or a dual-permeability medium is to be modeled, the flag sequence will be as follows:

a. **Enter the flags described in Section 6.5.1.A2, Line 1 to Line 24**
b. **Enter the flags described in Section 6.5.1.A3, Line 1 to Line 8**
c. **Enter the following flags for**

Fracture/Matrix Dispersion

Line 8a. Descriptor: Is fracture/matrix dispersion present (Y/N)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 8b. MFDISPREQ

MFDISPREQ: Fracture/matrix dispersion flag.
Accepts 'Y' or 'y' if dispersion from matrix to fracture is present and 'N' or 'n' if it is not.

d. **Enter the flags described in Section 6.5.1.A3, Line 9 to Line 26.**

6.5.1.A5 k_d Temperature-Dependency Information

This module provides NUTS with information about the constituents in which k_d is temperature dependent.

Line 1. Descriptor: Component name, reference temperature, and sorption exponential coefficient.
Accepts a character string up to 100 characters in length as a descriptive line.

IF either ADSTYPEF or ADSTYPEM (above) was selected as 'Y' or 'y', then:
DO I = 1, NUCLIDE
IF either FADSTEMPDEP(I) or MADSTEMPDEP(I) (above) was selected as "TRUE," then:

Line 2.NAME(I), REFTEMPKD(I), ADSEXPCOEFF(I)

NAME:	Dummy variable for the name of the constituent. *Accepts a 20-character string.*
REFTEMPKD:	The reference temperature at which k_d is evaluated. *Upper and lower bounds are not provided.*
ADSEXPCOEFF:	Sorption exponential coefficient. *Upper and lower bounds are not provided.*

END DO

6.5.1.A6 Molecular-Diffusion Temperature-Dependency Flags

These flags inform NUTS whether the molecular diffusion of a given constituent is temperature dependent in cases where **FDISPREQ** or **MDISPREQ** or **MFDISPREQ** is required ('Y' or 'y'). If none is required, no information will be provided by this item.

If FDISPREQ or MDISPREQ or MFDISPREQ is 'Y' or 'y', then:

Line 1. Descriptor: Component, name, and molecular-diffusion temperature dependency flag
Accepts a character string up to 100 characters in length as a descriptive line.

DO I = 1, NUCLIDE

Line 2.NAME(I), DMOLTEMDEP(I)

NAME:	Dummy variable for the name of the constituent. *Accepts a 20-character string.*
DMOLTEMDEP:	Logical flag for temperature dependency of the molecular diffusion. *Accepts T or F.*

END DO

6.5.1.A7 Printing-Frequency Information

This module specifies the frequency at which results will be printed to ASCII, BINARY, or CAMCON DATA BASE (CDB) output files.

If an ASCII file is to be output, then:

Line 1.Descriptor: Printing frequency in an ASCII file

Accepts a character string up to 100 characters in length as a descriptive line.

Line 2.IPRFRQA, TIMEASCMAX

IPRFRQA: Frequency of printing to an ASCII file, e.g.: 1 if every time step is to be printed, 2 for every other time step and so on.
Accepts an integer > 0.

TIMEASCMAX: Maximum time between printings. If this time is exceeded, NUTS will print the results regardless of the frequency specified above.
Accepts a real number >0 , but < the total simulation time.

If a BINARY file is to be output, then:

Line 3.Descriptor: Printing frequency to a BINARY file
Accepts a character string up to 100 characters in length as a descriptive line.

Line 4.IPRFRQB, TIMEBINMAX

IPRFRQB: Frequency of printing to a BINARY file.
Accepts an integer > 0; n means every nth timestep will be printed.

TIMEBINMAX: Maximum time allowed between printing interval.
Accept real number >0 , but < the total simulation time.

If a CDB file is to be output, then:

Line 5.Descriptor: Printing frequency to a CDB file
Accepts a character string up to 100 characters in length as a descriptive line.

Line 6.IPRFRQC, TIMECDBMAX

IPRFRQC:Frequency of printing to a CDB file.
Accepts an integer > 0; n means every nth timestep will be printed.
TIMECDBMAX: Maximum duration allowed between printings.
Accepts real numbers >0 , but < the total simulation time.

6.5.1.A8 External Source Flag

This flag identifies whether there is a dynamic source interfacing with NUTS. Two kinds of dynamic sources are used in NUTS. The first kind is an Actinide Source Submodel that has the ability to look at the distribution of the waste canisters, their corrosion and breach of the radioactive material, the chemical reaction of the waste with the brine, and the transport of the waste (by diffusion) from the breached canister to the porous media. This repository-scale calculation can be conducted through an interface between NUTS and the Actinide Source Submodel and is not used in the CCA calculations. Another dynamic source model is used in

NUTS for testing purposes. This source is a sort of interpolating function that has the ability to change the form (spike, slab, triangle, double hump, etc.) and the amount of the release. This function is built in NUTS and used to generate the sources in many of the test cases for code evaluation (see the QA documentation package NUTS).

Line 1. Descriptor: Is external source interfacing present (T/F)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 2.STOCKMAN

STOCKMAN: Logical flag to inform NUTS about the external source model.
Accepts T or F.

6.5.1.A9 Time-Minimum Limit

When a high degree of non-linearity is encountered in BRAGFLO calculations, the automatic time stepping module in BRAGFLO will reduce the time step to whatever value leads to the solution convergence. This value in some occasions could be as low as a fraction of a second. Because NUTS reads the time step information from the CDB, which is stored in single precision, and because of the precision (time is read and written in seconds), the difference between two successive times (time step size) for such a highly nonlinear step, can be zero. This inaccuracy in the time information is limited to a few time steps and irrelevant for the transport calculations. However, in the mathematical formulation of the transport equation, the time step appears as a divisor; in these instances, a division by zero problem will halt the calculation. To avoid this problem a time step minimum limit in the order of 10^{-16} or less is used whenever zero time step is encountered.

Line 1.Descriptor: Input timestep lower limits.
Accepts a character string up to 100 characters in length as a descriptive line.

Line 2. TIMELIMIT

TIMELIMIT: Time lower limit.
The value should not exceed 1.E-16 s

6.5.1.A10 Intrusion Time Interpolation and Extrapolation

Line 1. Descriptor: Actual time of intrusion, Interpolated/Extrapolated time, Tolerance. *Accepts a character string up to 100 characters in length as a descriptive line.*

If concentration initialization is to be done from the CDB (CONC_CDB_INITIALIZATION is true) then:

Line 2. TIME_INTRUSION, RESET_TIME_INTRUSION, TIMETOL

TIME_INTRUSION: The actual intrusion time specified in BRAGFLO run (350 and 1000 yrs for the CCA calculations).
RESET_TIME_INTRUSION: Interpolated/Extrapolated intrusion time.
TIMETOL:* Accepted tolerance to initialize at closest time to the RESET_TIME_INTRUSION.

6.5.1.B Material Map Input

NUTS's interactive input will determine the location from which NUTS will read the material map. If the user specified **BRAGFLO** input as the source for the material map, NUTS will open BRAGFLO's input file and read the material map and various reference condition parameters. NUTS assumes BRAGFLO's input and output files have the same name with an extension **.INP** for input and **.BIN or .CDB** for the output. If the user specified NUTS as a source for the material map, NUTS will proceed reading the input file in the following sequence:

Line 1.Descriptor: Number of times for specifying material map
*Accepts a character string up to *17 characters as a descriptive line.*

Note: NUTS will key on this label to start reading the material map, regardless of whether BRAGFLO or NUTS input is specified. Because of the format of BRAGFLO input, **this label should be used exactly preceded by two or three empty spaces (see the example in Appendix B).**

Line 2. NMATTIMES
NMATTIMES: Number of times the material map will change during simulation time.
Accepts an integer > 0.

DO I = 1, NMATTIMES

Line 3. Descriptor(I): Start time for material map
Accepts a character string up to 17 characters in length as a descriptive line.

Line 4.TIMEMAST(I)
TIMEMAST: Starting time for material map (s).
Accepts a real number > 0.

Line 5.Descriptor(I): Material-type grid map
Accepts a character string up to 17 characters in length as a descriptive line.

Line 6.(((IMAT(I,J,K), I = 1, NX), J = 1, NY), K = 1, NZ)
IMAT(I,J,K): An array that holds the material map of the grid blocks for the entire spatial domain.

* NUTS will read the time in the CDB file and add the tolerance. If the total of time + TIMETOL ≥ RESET_TIME_INTRUSION, this time will be picked up for initialization.

Accepts an integer > 0.

END DO

Line 7. Descriptor: Material name
Accepts a character string up to 17 characters in length as a descriptive line.

DO I = 1, NMAT

Line 8. J,MAT_NAME(I)

J: Material number in the material map. *Accepts an integer.*

MAT_NAME: Material name .
Accepts a 50-character string.
Note: NMAT is the number of the material specified in the material map. NMAT is compared with the input material number (NMAT_INPUT). If the two numbers do not agree, *NUTS will print a warning message and abort.*

END DO

Line 9.Descriptor: Number of waste regions.
Accepts a character string up to 17 characters in length as a descriptive line.

Line 10. NWST

NWST: Number of waste regions. If the NWST>NSITES=1, NUTS will combine the waste regions into one site. In the cases when NWST>NSITES>1, NUTS will print a warning message on screen and it will terminate.
Accepts an integer > 0 and = NSITES.

Line 11. Descriptor: Names of waste regions.
Accepts a character string up to 17 characters in length as a descriptive line.

Line 12. (MAT_WASTE(I), I = 1,NWST)

Line 13. (MAT_WASTE(I), I = 1,NWST)

MAT_WASTE: Waste region material index.
Accepts any integer.

If the material map input is from BRAGFLO, NUTS will key on ' REFERENCE TEMPE' to read the following:

Line 13. RTEMP_BRAG, RPRES_BRAG

RTEMP_BRAG: Reference temperature.
Accepts 273.15<any real number < 473.15

RPRES_BRAG: Reference pressure.
Accepts 0 < any real, positive number < 1.E10

If the material map input is from BRAGFLO, NUTS will also key on" SALT% DEN" to read:

Line 14. A1, A2, A3, A4, A5

A1: Reference brine density.

A2, A3, and A4: BRAGFLO parameters irrelevant to NUTS applications.

A5: Brine compressibility.

Note: A1 and A2 are not checked directly, however they are used to calculate density under reservoir conditions. Since the density bounds are checked, A1 and A2 are implicitly checked.

If the material map input is from BRAGFLO, NUTS will also key on "# LAMBDA"to read:

DO I=1, NMAT
Line 15, IA1, A2, BRESD_MAT(I)

IA1, A2: BRAGFLO parameters irrelevant to NUTS applications.

BRESD_MAT: Brine residual saturation of the material.

END DO

An example of the material map with a 9-grid-block spatial domain for one time is given in Appendix B.

6.5.1.C Properties Input

6.5.1.C1 Physical Properties

Line 1. **Descriptor:** End of material map and start of physical properties input
Accepts a character string up to 100 characters in length as a descriptive line.

Line 2. **Descriptor:** Element solubility input
Accepts a character string up to 100 characters in length as a descriptive line.

If the data are to be read from NUTS's input file and not the CDB, then:

DO I = 1, NOELEMENT
If the solubility is not temperature dependent and not table look-up (ELTEMP_SOLB(I) .and. SOLB_TABLE(I) = .FALSE.) then:

Line 3. NAME(I), ALOG_ELEMNT_SOLB_LIMIT(I)

NAME: Dummy variable for the element name.
Accepts a 20-character string.

ALOG_ELEMNT_SOLB_LIMIT: Log_{10} of the element solubility limit (mol/liter).

Accepts any real number for which Log_{10} inverse exists.

END DO

DO I = 1, NOELEMENT
If the solubility is table look-up (SOLB_TABLE(I) = .TRUE.) then:

Line 4. NAME(I),EQSPACED(I),ID_INT
NAME: Dummy variable for the element name.
Accepts a 20-character string.
EQSPACED: Logical flag for equally spaced entries.
Accepts T or F.
ID_INT: Interpolation table dimensions identifier.
Accepts two values: ID_INT = 1 for 1D interpolation, and ID_INT = 2 for 2D interpolation tables

If 1D table then:
Line 5. NROW1D
NROW1D: Number of rows in the table.
Accepts an integer > 0.

DO K = 1, NROW1D
Line 6. ROW1D, FX
ROW1D: Row entry in the table.
Accepts any real number.
FX: Table entries.
Accepts any real number.

END DO
If 2D table then:
Line 7. NROW(I), NCOLUMN(I)
NROW: Number of rows in the table.
Accepts an integer > 0.
NCOLUMN: Number of columns in the table.
Accepts an integer > 0.
Line 8. (COLUMN(J,I), J = 1,NCOLUMN(I)
COLUMN: Column entry in two-dimensional table.
Accepts any real number.

DO K = 1, NROW(I)
Line 9. ROW(K,I),(FXY(K,NNC,I), NNC = 1, NCOLUMN(I))
ROW: Row entry in two-dimensional table.
Accepts any real number.
FXY: Table entries.
Accepts any real number.

END DO

END DO

Note: In the above table the units *must* be consistent.

DO I = 1, NOELEMENT
If the solubility is temperature dependent (ELTEMP_SOLB(I) = .TRUE.) then:

Line 10. NAME(I), C0(I),C1(I),C2(I),C3(I),C4(I),C5(I)

NAME: Dummy variable for the element name.
Accepts a 20-character string.

C0-5: Coefficients of the correlation polynomial.
Accepts any real number.

END DO

Line 11. Descriptor: Name, molecular (or atomic) weight, inventory, half-life
Accepts a character string up to 100 characters in length as a descriptive line.

If the data are to be read from NUTS's input file and not the CDB, then:
DO I = 1, NUCLIDE

Line 12. NAME(I),XMOLWT(I),CUINVCHD(I),CUINVRHD(I),HALF-LIFE(I)

NAME: Dummy variable for component name.
Accepts a 20-character string.

XMOLWT: Molecular or atomic weight of the component (kg-mol).
Accepts any real number > 0 and < 1 .

CUINVCHD: Curies of contact handled inventory (Ci).
Accepts any real number > 0.

CUINVRHD: Curies of remote handled inventory (Ci).
Accepts any real number > 0.

HALF-LIFE: Component half-life (s).
Accepts any real number >= 0.

Note: 0 is used for HALF-LIFE as a flag to identify stable components.
END DO

If phase type is gas, then:

Line 13. Descriptor: Component name, intercept and slope of equilibrium line
Accepts a character string up to 100 characters in length as a descriptive line.

DO I = 1, NUCLIDE

Line 14. NAME(I), EQCI(I), EQCS(I)

NAME: Dummy variable for component name.
Accepts a 20-character string.

EQCI: Intercept of gas-liquid equilibrium line.
Accepts any meaningful real number.

EQCS: Slope of gas-liquid equilibrium line.
Accepts any meaningful real number.

END DO

Line 15. Descriptor: Ground-water pH Input
Accepts a character string up to 100 characters in length as a descriptive line.

If PHREQ is 'Y' or 'y', then:

If the data are to be read from NUTS's input file and not the CDB, then:

Line 16. (RPH(I), I = 1, NMAT)
RPH: pH of ground water for the range specified by the material map index.
Accepts 0 < RPH < 14.

Line 17. Descriptor: Standard conditions brine density if a test run.
Accepts a character string up to 100 characters in length as a descriptive line.

If the run is a test run (densities can not be read from BRAGFLO), then:

Line 18. RBR_DEN
RBR_DEN: Standard condition brine density.
Accepts 0 < real number < 2000.

6.5.1.C2 Component Molecular-Diffusion Input

Line 1. Descriptor: Molecular diffusion input
Accepts a character string up to 100 characters in length as a descriptive line.

If MDISPREQ or FDISPREQ or MFDISPREQ is 'Y' or 'y', then:
If the data are to be read from NUTS input file and not the CDB, then:

DO I = 1, NUCLIDE

Line 2. NAME(I), DMOL(I)
NAME: Dummy variable for component name.
Accepts a 20-character string.
DMOL: Molecular diffusion of the component (m^2/s).
Accepts a real number > 0 and < a number that gives a molecular diffusional velocity+ of 0.6 m/s

END DO

Line 3. Descriptor: Reference viscosity and temperature
Accepts a character string up to 100 characters in length as a descriptive line.

DO I = 1, NUCLIDE
If DMOLTEMDEP(I) is .TRUE. then:
C2.1.3 VISREF(I), TREF(I)

+ diffusional velocity is calculated as = DMOL * distance (in the direction of the flow)/area (perpendicular to the direction of the flow). The value of 0.6 m/s is based on a Reynolds number = 1 in the porous medium, which assures a linear velocity dependence (i.e., satisfies Darcy's Law).

VISREF: Reference viscosity at which molecular diffusion is measured (Pa/s).
Accepts a real number > 0.
TREF: Reference temperature at which molecular diffusion is measured (K).
Accepts a real number > 0.

END DO

6.5.1.C3 Rock-Density Input

This module reads the density of each material mentioned in the material map.

Line 1. Descriptor: Rock density
Accepts a character string up to 100 characters in length as a descriptive line.

If soil base concentration is required to be output (option IPRNTFA(10) or IPRNTFB(10) or IPRNTFC(10) or IPRNTMA(10) or IPRNTMB(10) or IPRNTMC(10)) is 1 or sorption is to be calculated (ADSTYPEM or ADSTYPEF is "Y" or "y") and if the data are to be read from NUTS input file and not the CDB, then,

Line 2. (RROCK_DENS(I), I = 1, NMAT)
RROCK_DENS: Rock grain density for the range specified by the material map index (kg/m^3). *Accepts a real number > 0 and < 5000.*

6.5.1.C4 Waste-Matrix Input

In the cases where the waste is placed in a certain grid block belonging to a certain site, this input specifies the number of constituents, which constituents they are, and where in the grid-block map these constituents are located. It is this matrix that defines where to initialize the concentrations of the waste material.

Line 1. Descriptor: Waste matrix input
Accepts a character string up to 100 characters in length as a descriptive line.

Line 2. NWSTCOMP_TOTAL
NWSTCOMP_TOTAL: Number of Component placed in the waste.
Accepts an integer > 0 and < the total number of components.

If NWSTCOMP_TOTAL is zero, then stop waste matrix input here. Otherwise,

DO I = 1, NWSTCOMP_TOTAL

Line 3. NAME(I),LOCWASTINP(I),NWASTREGION(I)
NAME: Dummy variable for component name.
Accepts a 20-character string.
LOCWASTINP: Location of the component in the input (COMPONENT_NAME list).
Accepts an integer > 0 and < the total number of components.
NWASTREGION: Waste site number.
Accepts an integer > 0 and < NSITES.

END DO

Line 4. Descriptor: (1. site name, number of grid; 2. indices of waste matrix)
Accepts a character string up to 100 characters in length as a descriptive line.

DO I = 1, NSITES

Line 5. SNAME(I),NGRIDSITE(I)

SNAME:	A dummy character for the site name. *Accepts a 20-character string.*
NGRIDSITE:	Number of grid block of waste in that particular site. *Accepts an integer > 0 and < the total number of grid blocks.*

Line 6. (IS(I,J),JS(I,J),KS(I,J), J=1, NGRIDSITE(I))

IS,JS,KS:	i,j, and k indices of the grid blocks in the waste site. *Accepts an integer > 0 and IS ≤ NX, JS ≤ NY, and KS ≤ NZ.*

6.5.1.C5 Single-Porosity Fracture Input

M

If a single-porosity fracture medium is to be modeled, then:

Fracture Sorption Input

This module inputs the constituent sorption properties of the fracture. Three equilibrium isotherms are considered depending on the value of FRAC_ISOTHERM.

Line 1. Descriptor: Fracture sorption
Accepts a character string up to 100 characters in length as a descriptive line.

If the data are to be read from NUTS's input file and not the CDB, then:

If ADSTYPEF is 'N' or 'n', then stop the input for this module here. Otherwise,

If FRAC_ISOTHERM IS 'L' or 'l' then:

DO I = 1, NUCLIDE

Line 2. NAME(I)

NAME: Dummy variable for component name.
Accepts a 20-character string.

Line 3. (RXLF(J), J = 1, NMAT)

RXLF: Linear sorption coefficient (k_d) of the fracture for the range specified by the material map index (m^3/kg).
Accepts a number > 1.

END DO

If FRAC_ISOTHERM is 'F' or 'f' then:

DO I = 1, NUCLIDE

Line 4. NAME(I)

NAME: Dummy variable for component name.
Accepts a 20-character string.

Line 5.(RXFDCF(J), J = 1, NMAT)

Line 6.(RXFCF(J), J = 1, NMAT)

RXFDCF: Freundlich distribution coefficient of the fracture for the range specified by the material map index (m^3fluid/kg solid).
Upper and lower bounds are not specified.

RXFCF: Freundlich coefficient of the fracture for the range specified by the material map index (dimensionless).
Upper and lower bounds are not specified

END DO

If FRAC_ISOTHERM is 'LA' or 'la' or 'La', then:

DO I = 1, NUCLIDE

Line 7. NAME(I)

NAME: Dummy variable for component name.
Accepts a 20-character string.

Line 8. (RXLDCF(J), J = 1, NMAT)

Line 9. (RXLCF(J), J = 1, NMAT)

RXLDCF: Langumir distribution coefficient of the fracture for the range specified by the material map index (m^3fluid/kg solid).
Upper and lower bounds are not specified.

RXLCF: Langumir coefficient of the fracture for the range specified by the material map index (dimensionless).
Upper and lower bounds are not specified.

END DO

Fracture Dispersion Input

This module inputs dispersion parameters for the fracture.

Line 10. Descriptor: Fracture dispersion

Accepts a character string up to 100 characters in length as a descriptive line.

If the data are to be read from NUTS's input file and not the CDB, then:
If FDISPREQ is 'N' or 'n' stop the input for this module here. Otherwise,

Line 11. Descriptor: Longitudinal dispersivity in the fracture
Accepts a character string up to 100 characters in length as a descriptive line.

Line 12. (RALPHALF(I), I = 1, NMAT)
RALPHALF: Longitudinal dispersivity of the fracture for the range specified by the material map index (m).
Accepts a real number > 0 and < the smallest hydrological dimension of the spatial domain.

Line 13. Descriptor: Transverse dispersivity in the fracture
Accepts a character string up to 100 characters in length as a descriptive line.

Line 14. (RALPHATF(I), I = 1, NMAT)
RALPHATF: Transverse dispersivity of the fracture for the range specified by the material map index (m).
Accepts a real number > 0 and < longitudinal dispersivity.

Line 15. Descriptor: Tortuosity in the fracture
Accepts a character string up to 100 characters in length as a descriptive line.

Line 16. (RTORF(I), I = 1, NMAT)
RTORF: Fracture tortuosity for the range specified by the material map index (dimensionless).
Accepts a real number > 1.

Fracture Source Input

Line 17. Descriptor: Start fracture source input for injection/production
Accepts a character string up to 100 characters in length as a descriptive line.

If FSTATUSINJ is 'N' or 'n', then skip the next Lines 18 to 25, otherwise,

Line 18. NTMNINJCOMP
NTMNINJCOMP: Total number of injected components.
Accepts an integer 0<NTMNINJCOMP < total number of component

DO J = 1, NTMNINJCOMP

Line 19. NAME(J), LOCINPM(J), NMTBLOKINJ(J)
NAME: Dummy variable for component name.

Accepts a 20-character string.

LOCINPM: Location of the injected component in the component input sequence.
Accepts an integer 0<LOCINPM < total number of component

NMTBLOKINJ: Total number of grid blocks at which this component is injected.
Accepts an integer 0 < NMTBLOKINJ *< total number of grid blocks*

Line 20. (INJBIM(I,J), INJBJM(I,J), INJBKM(I,J), I = 1, NMTBLOKINJ(J))

INJBIM: I index of the injection block.
*Accepts an integer 0<*INJBIM *< total number of grid blocks in x direction.*

INJBJM: J index of the injection block.
*Accepts an integer 0<*INJBJM *< total number of grid blocks in y direction.*

INJBKM: K index of the injection block.
*Accepts an integer 0<*INJBKM *< total number of grid blocks in z direction*

END DO

DO I = 1, NTMNINJCOMP

Line 21. NAME(I)

NAME: Dummy variable for component name.
Accepts a 20-character string.

Line 22. (CONCMIJKINJ(INJBIM(J,I), INJBJM(J,I), INJBKM(J,I)), J = 1, NMTBLOKINJ(I))

CONCMIJKINJ: Concentration of the injected component (kg/m^3).
Accepts a real number > 0.

END DO

Line 23. Descriptor: Injection production rates
Accepts a character string up to 100 characters in length as a descriptive line.

Line 24. NMBLOCKSOURCE, TIMEFSTRT, TIMEFEND

NMBLOCKSOURCE: Number of grid blocks having the source.
Accepts an integer 0 < NMBLOCKSOURCE *< total number of grid blocks.*

TIMEFSTRT: Starting time of injection/production in fracture (s).
Accepts a real number 0 < TIMEFSTRT < TIMEFEND

TIMEFEND: Ending time of injection/production in fracture (s).
Accepts a real number 0 < TIMEFEND

DO I = 1, NMBLOCKSOURCE

Line 25. IQIM(I), IQJM(I), IQKM(I), RATE(I)

IQIM, IQJM, IQKM: I,J,K indices of the injection/production block.

Accepts an integer 0 < IQIM, IQJM, IQKM < *total number of grid blocks in x, y, z direction.*

RATE: Injection/production rate (m^3/s).

Accepts a real number > 0.

END DO

Line 26. Descriptor: Dirichlet boundary condition in the fracture

Accepts a character string up to 100 characters in length as a descriptive line.

If FDIRICHLET then:

Line 27. NTMDIRCOMP,TYPEDIR

NTMDIRCOMP: Number of components with Dirichlet boundary condition (DBC)

Accepts a real number 0 < NTMDIRCOMP < *total number of component.*

TYPEDIR: Character to identify whether DBCs are specified in the repository only or may occur elsewhere in the domain.

Accepts two values; (1) GENERAL : DBCs occur only in the repository, or (2) NOT-GENERAL: DBCs may occur anywhere.

M

DO J = 1, NTMDIRCOMP

If TYPEDIR is GENERAL or general, then:

Line 28. NAME(J),LOCMDIRINP(J),GCONDIR(J)

NAME: Dummy variable for component name.

Accepts a 20-character string.

LOCMDIRINP: Location of the component in the input sequence.

Accepts an integer 0 < LOCMDIRINP < *total number of component.*

GCONDIR: Dirichlet B.C.s on Concentration (kg/m^3).

Accepts a real number > 0.

If TYPEDIR is NOT-GENERAL, then:

Line 29. NAME(J), LOCMDIRINP(J), NMTDIRBLOK(J)

NAME: Dummy variable for component name.

Accepts a 20-character string.

LOCMDIRINP: Location of the component in the input sequence.

Accepts an integer 0 < LOCMDIRINP < *total number of components.*

NMTDIRBLOK: Total number of grid blocks with D.B.C.

Accepts an integer 0 < NMTDIRBLOK < *total number of grid blocks.*

Line 30. (IDRBIM(I,J), IDRBJM(I,J), IDRBKM(I,J), I = 1, NMTDIRBLOK(J))

IDRBIM, IDRBJM, IDRBKM: I,J,K indices of the D.B.C grid blocks.

Accepts an integer 0 <IDRBIM, IDRBJM, IDRBKM < *total number of grid blocks in x, y, z direction, respectively.*

END DO

DO I = 1, NTMDIRCOMP

Line 31. NAME(I)

NAME: Dummy variable for component name.
Accepts a 20-character string.

Line 32. (DIRCONCMIJK(IDRBIM(J,I), IDRBJM(J,I), IDRBKM(J,I)), J = 1, NMTDIRBLOK(I))

DIRCONCMIJK DBC on concentration of the ijk th component (kg/m^3).
Accepts a real number > 0.

END DO

Fracture Dynamic Source Function Input

This module is created to generate a dynamic source function in the fracture. The shape of the source here can vary by changing the correlation parameters that specify the time range and the normalization factor. The function is activated by the same flag used for the Actinide Source submodel. This source is usually needed in the testing processes outside the CAMCON environment.

Line 33. Descriptor: Testing dynamic source input.

If (.NOT. STOCKMAN) skip Line 34 to Line 40.

Line 34. NTMDEPSRC

NTMDEPSRC: Total number of the radioactive istopes that have time dependent source. *Accepts a real number 0* < NTMDEPSRC < *total number of component.*

If NTMDEPSRC is zero skip line 35 to Line 40.

DO J = 1, NTMDEPSRC

Line 35. NAME(J), LOCSRCINP(J), NMTBLOKSRC(J)

NAME: Dummy variable for component name.
Accepts a 20-character string.

LOCMDIRINP: Location of the component in the input sequence.
Accepts an integer 0 < LOCMDIRINP < *total number of components.*

NMTBLOKSRC: Total number of grid blocks with dynamic source.

Accepts an integer 0 < NMTBLOKSRC < *total number of grid blocks.*

Line 36. (ISRCI(I,J), ISRCJ(I,J), ISRCK(I,J), I = 1, NMTBLOKSRC(J))

ISRCI: I index of the grid block that has the source.

ISRCJ: J index of the grid block that has the source.

ISRCK: K index of the grid block that has the source.

Accepts an integer 0 <ISRCI, ISRCJ, ISRCK < *total number of grid blocks in x, ,y, z direction, respectively.*

END DO

DO J = 1, NTMDEPSRC

Line 37. NAME(J)

NAME: Dummy variable for component name.

Accepts a 20-character string.

Line 38. (QC0IJK(ISRCI(I,J), ISRCJ(I,J), ISRCK(I,J), I = 1, NMTBLOKSRC(J))

QC0IJK: The source strength of the grid block in kg/s.

Accepts any real number.

Line 39.

**((TBIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
TCIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
TDIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
TEIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
TFIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J))),
I = 1, NMTBLOKSRC(J))**

TBIJK: Start of time range 1.
TCIJK: Start of time range 2.
TDIJK: Start of time range 3.
TEIJK: Start of time range 4.
TFIJK: Start of time range 5.

Line 40.

**((GBIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
GCIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
GDIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
GEIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
GFIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J))),
I = 1, NMTBLOKSRC(J))**

GBIJK: Start of normalization factor range 1.

GCIJK: Start of normalization factor range 2.
GDIJK: Start of normalization factor range 3.
GEIJK: Start of normalization factor range 4.
GFIJK: Start of normalization factor range 5.

END DO

Fracture Concentration Initialization Input

Line 41. Descriptor: Initial concentration in the fracture if any.
Accepts character string up to 100 characters in length as a descriptive line.

If FMAN_CONC_INIT is true, then:

Line 42. NINITCOMP

NINITCOMP: Number of components to be initialized.
Accepts 0 < integer < total number of components.

DO J = 1, NINITCOMP

Line 43. NAME(J),LOCINP(J)NBLOKINI(J)

NAME: Dummy variable for component name.
Accepts a 20-character string.

LOCINP: Location of the component in the input sequence.
Accepts an integer 0 <LOCINP < *total number of components.*

NBLOKINI: Total number of grid blocks initialized.
Accepts an integer 0 < NBLOKINI < *total number of grid blocks.*

Line 44. (INII(I,J), IINIJ(I,J), INIK(I,J), I = 1, NBLOKINI(J))

INII, INIJ, INIK: I,J,K indices of the initialized grid blocks.
Accepts an integer 0<INII, INIJ, INIK < *total number of grid blocks in x, y, z direction, respectively.*

END DO

DO J = 1, NINITCOMP

Line 45. NAME(J)

NAME: Dummy variable for component name.
Accepts a 20-character string.

Line 46. CONCIJKINI(INII(I,J), IINIJ(I,J), INIK(I,J), I = 1, NBLOKINI(J))

CONCIJKINI: Initialized concentration of the grid.
Accepts a real number.

END DO

Fracture Colloid Transport Scaling Factors Input

Line 47. Descriptor: Colloidal transport scaling factors

Accepts a character string up to 100 characters in length as a descriptive line.

If colloidal transport is required, then:

Line 48. Descriptor: Scaling factor for velocities in x-direction.
Accepts a character string up to 100 characters in length as a descriptive line.

Line 49. (SCALEX(I,J,K),I=1,NX),J=1,NY),K=1,NZ)
SCALEX: Scaling factors of x-direction velocities.
Accepts an integer ≥ 1.

Line 50. Descriptor: Scaling factor for velocities in y-direction.
Accepts a character string up to 100 characters in length as a descriptive line.

Line 51. (SCALEY(I,J,K),I=1,NX),J=1,NY),K=1,NZ)
SCALEY: Scaling factors of y-direction velocities.
Accepts an integer ≥ 1.

Line 52. Descriptor: Scaling factor for velocities in z-direction.
Accepts a character string up to 100 characters in length as a descriptive line.

Line 53. (SCALEZ(I,J,K),I=1,NX),J=1,NY),K=1,NZ)
SCALEZ: Scaling factors of z-direction velocities.
Accepts an integer ≥ 1.

6.5.1.C6 Single-Porosity Matrix Input

If a single-porosity matrix is to be modeled, then:

Matrix Sorption Input

This module inputs the constituent sorption properties of the matrix. Three equilibrium isotherms are considered, depending on the value of MAT_ISOTHERM.

Line 1. Descriptor: Matrix sorption
Accepts a character string up to 100 characters in length as a descriptive line.

If the data are to be read from NUTS's input file and not the CDB, then:
If ADSTYPEM is 'N' or 'n', terminate inputs for this module here. Otherwise,

If MAT_ISOTHERM IS 'L' or 'l', then:

DO I = 1, NUCLIDE

Line 2. NAME(I)
NAME: Dummy variable for component name.

Accepts a 20-character string.

Line 3. (RXLM(J), J = 1, NMAT)

RXLM: Linear sorption coefficient (k_d) of the matrix for the range specified by the material map index (m^3/kg).
Accepts a number > 0

END DO

If MAT_ISOTHERM is 'F' or 'f' then:

DO I = 1, NUCLIDE

Line 4. NAME(I)

NAME: Dummy variable for component name.
Accepts a 20-character string.

Line 5. (RXFDCM(J), J = 1, NMAT)

Line 6. (RXFCM(J), J = 1, NMAT)

RXFDCM: Freundlich distribution coefficient of the matrix for the range specified by the material map index (m^3/kg).
Upper and lower bounds are not specified.

RXFCM: Freundlich coefficient of the matrix for the range specified by the material map index (dimensionless).
Upper and lower bounds are not specified.

END DO

If MAT_ISOTHERM is 'LA' or 'la' or 'La', then:

DO I = 1, NUCLIDE

Line 7. NAME(I)

NAME: Dummy variable for component name.
Accepts a 20-character string.

Line 8. (RXLDCM(J), J = 1, NMAT)

Line 9. (RXLCM(J), J = 1, NMAT)

RXLDCM: Langumir distribution coefficient of the Matrix for the range specified by the material map index (m^3/kg).
Upper and lower bounds are not specified.

RXLCM: Langumir coefficient of the matrix for the range specified by the material map index (dimensionless).
Upper and lower bounds are not specified.

END DO

Matrix Dispersion Input

This module inputs dispersion parameters for the Matrix.

Line 10. Descriptor: Matrix dispersion
Accepts a character string up to 100 characters in length as a descriptive line.

If the data are to be read from NUTS input file and not the CDB, then:
If MDISPREQ is 'N' or 'n' stop the input for this module here, otherwise,

Line 11. Descriptor: Longitudinal dispersivity in the matrix
Accepts a character string up to 100 characters in length as a descriptive line.

Line 12. (RALPHALM(I), I = 1, NMAT)
RALPHALM: Longitudinal dispersivity of the matrix for the range specified by the material map index (m).
Accepts 0 < any real number < the smallest hydrological dimension of the spatial domain.

Line 13. Descriptor: Transverse dispersivity in the matrix
Accepts a character string up to 100 characters in length as a descriptive line.

Line 14. (RALPHATM(I), I = 1, NMAT)
RALPHATM: Transverse dispersivity of the matrix for the range specified by the material map index (m).
Accepts a real number > 0 and < longitudinal dispersivity.

Line 15. Descriptor: Tortuosity in the matrix
Accepts a character string up to 100 characters in length as a descriptive line.

Line 16. (RTORM(I), I = 1, NMAT)
RTORM Matrix tortuosity for the range specified by the material map index (dimensionless). *Accepts a real number > 1.*

Matrix Source Input

Line 17. Descriptor: Start matrix source input for injection/production
Accepts a character string up to 100 characters in length as a descriptive line.

If MSTATUSINJ is 'N' or 'n', then skip Lines 18 to 25. Otherwise,

Line 18. NTMNINJCOMP
NTMNINJCOMP: Total number of injected components.
Accepts 0 < NTMNINJCOMP < total number of components.

DO J = 1, NTMNINJCOMP

Line 19. NAME(J), LOCINPM(J), NMTBLOKINJ(J)
NAME: Dummy variable for component name.
Accepts a 20-character string.
LOCINPM: Location of the injected component in the component input sequence.

Accepts an integer 0 < LOCINPM < total number of components.

NMTBLOKINJ: Total number of grid blocks at which this component is injected.
Accepts an integer 0 < NMTBLOKINJ *< total number of grid blocks.*

Line 20. (INJBIM(I,J), INJBJM(I,J), INJBKM(I,J), I = 1, NMTBLOKINJ(J))

INJBIM: I index of the injection block.
Accepts an integer 0 < INJBIM *< total number of grid blocks in x direction.*

INJBJM: J index of the injection block.
Accepts an integer 0 < INJBJM *< total number of grid blocks in y direction.*

INJBKM: K index of the injection block.
Accepts an integer 0 < INJBKM *< total number of grid blocks in z direction.*

END DO

DO I = 1, NTMNINJCOMP

Line 21. NAME(I)

NAME: Dummy variable for component name.
Accepts a 20-character string.

Line 22. (CONCMIJKINJ(INJBIM(J,I), INJBJM(J,I), INJBKM(J,I)), J = 1, NMTBLOKINJ(I))

CONCMIJKINJ: Concentration of the injected component (kg/m^3).
Accepts a real number > 0.

END DO

Line 23. Descriptor: Injection production rates
Accepts a character string up to 100 characters in length as a descriptive line.

Line 24. NMBLOCKSOURCE, TIMEMSTRT, TIMEMEND

NMBLOCKSOURCE: Number of grid blocks having the source.
Accepts an integer 0 < NMBLOCKSOURCE *< total number of grid blocks.*

TIMEFSTRT: Starting time of injection/production ithe matrix (s).
Accepts a real number 0 < TIMEFSTRT < TIMEFEND.

TIMEFEND: Ending time of injection/production in the matrix (s).
Accepts a real number 0 < TIMEFEND.

DO I = 1, NMBLOCKSOURCE

Line 25. IQIM(I), IQJM(I), IQKM(I), RATE(I)

IQIM, IQJM, IQKM: I,J,K indices of the injection/production block.
*Accepts an integer 0<*IQIM, IQJM, IQKM *< total number of grid blocks in x ,y, z direction, respectively.*

RATE: Injection/production rate (m^3/s).
Accepts a real number > 0.

END DO

Line 26: Descriptor: Dirichet boundary condition in the matrix
Accepts a character string up to 100 characters in length as a descriptive line.

If MDIRICHLET, then:

Line 27. NTMDIRCOMP,TYPEDIR

NTMDIRCOMP: Number of components with Dirchlet boundary condition (DBC).
Accepts 0 < NTMDIRCOMP *< total number of components.*

TYPEDIR: Character to identify whether DBCs are specified in the repository only or elsewhere in the domain.
Accepts two values (1) GENERAL; for the whole repository DBC, or (2) NOT-GENERAL; for DBC specified in specific places.

DO J = 1, NTMDIRCOMP

If TYPEDIR is GENERAL or general then:

Line 28. NAME(J),LOCMDIRINP(J),GCONDIR(J)

NAME: Dummy variable for component name.
Accepts a 20-character string.

LOCMDIRINP: Location of the component in the input sequence.
Accepts an integer 0 < LOCMDIRINP *< total number of components.*

GCONDIR: Dirchlet boundary condition on concentration (kg/m^3).
Accepts a real number > 0.

If TYPEDIR is NOT-GENERAL or not-general, then:

Line 29. NAME(J), LOCMDIRINP(J), NMTDIRBLOK(J)

NAME: Dummy variable for component name.
Accepts a 20-character string.

LOCMDIRINP: Location of the component in the input sequence.
Accepts an integer 0 < LOCMDIRINP *< total number of components.*

NMTDIRBLOK: Total number of grid blocks with D.B.C.
Accepts an integer 0 < NMTDIRBLOK *< total number of grid blocks.*

Line 30. (IDRBIM(I,J), IDRBJM(I,J), IDRBKM(I,J), I = 1, NMTDIRBLOK(J))

IDRBIM, IDRBJM, IDRBKM: I,J,K indices of the D.B.C grid blocks.
Accepts an integer 0 < IDRBIM, IDRBJM, IDRBKM *< total number of grid blocks in x ,y, z direction, respectively.*

Endif
END DO

DO I = 1, NTMDIRCOMP

Line 31. NAME(I)

NAME: Dummy variable for component name.
Accepts a 20-character string.

Line 32. (DIRCONCMIJK(IDRBIM(J,I), IDRBJM(J,I), IDRBKM(J,I)),J = 1, NMTDIRBLOK(I))

DIRCONCMIJK: Concentration of the component having DBC (kg/m^3).
Accepts a real number > 0.

END DO

Matrix Dynamic Source Function Input

This module is created to generate a dynamic source function in the matrix. The shape of the source here can vary by changing the correlation parameters that specify the time range and the normalization factor. The function is activitated by the same flag used for the Actinide Source submodel. This source is usually needed in the testing processes outside the CAMCON environment.

Line 33. Descriptor: Testing Dynamic source input.

If (.NOT. STOCKMAN) skip Line 34 to Line 40.

Line 34. NTMDEPSRC

NTMDEPSRC Total number of the radioactive istopes that have time dependent source. *Accepts a real number 0 < NTMDEPSRC < total number of component.*

If NTMDEPSRC is zero skip line 35 to Line 40.

DO J = 1, NTMDEPSRC

Line 35. NAME(J), LOCSRCINP(J), NMTBLOKSRC(J)

NAME: Dummy variable for component name.
Accepts a 20-character string.

LOCMDIRINP: Location of the component in the input sequence.
Accepts an integer 0 < LOCMDIRINP < total number of components.

NMTBLOKSRC: Total number of grid blocks with dynamic source.
Accepts an integer 0 < NMTBLOKSRC < total number of grid blocks.

Line 36. (ISRCI(I,J), ISRCJ(I,J), ISRCK(I,J), I = 1, NMTBLOKSRC(J))

ISRCI: I index of the grid block that has the source.
ISRCJ: J index of the grid block that has the source.
ISRCK: K index of the grid block that has the source.
Accepts an integer 0 <ISRCI, ISRCJ, ISRCK < total number of grid blocks in x, ,y, z direction, respectively.

END DO

DO J = 1, NTMDEPSRC

Line 37. NAME(J)
NAME: Dummy variable for component name.
Accepts a 20-character string.

Line 38. (QC0IJK(ISRCI(I,J), ISRCJ(I,J), ISRCK(I,J), I = 1, NMTBLOKSRC(J))
QC0IJK: The source strength of the grid block in kg/s.
Accepts any real number.

Line 39.
((TBIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
TCIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
TDIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
TEIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
TFIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J))),
I = 1, NMTBLOKSRC(J))

TBIJK: Start of time range 1.
TCIJK: Start of time range 2.
TDIJK: Start of time range 3.
TEIJK: Start of time range 4.
TFIJK: Start of time range 5.

Line 40.
((GBIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
GCIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
GDIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
GEIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J)),
GFIJK(ISRCI(I,J),ISRCJ(I,J),ISRCK(I,J))),
I = 1, NMTBLOKSRC(J))

GBIJK: Start of normalization factor range 1.
GCIJK: Start of normalization factor range 2.
GDIJK: Start of normalization factor range 3.
GEIJK: Start of normalization factor range 4.
GFIJK: Start of normalization factor range 5.

END DO

Matrix Concentration Initialization Input

Line 41. Descriptor: Initial concentration in the matrix if any.
Accepts a character string up to 100 characters in length as a descriptive line.

If MMAN_CONC_INIT is true, then:

Line 42. NINITCOMP

NINITCOMP: Number of components to be initialized.
Accepts 0 < integer < total number of components.

DO J = 1, NINITCOMP

Line 43. NAME(J),LOCINP(J)NBLOKINI(J)

NAME: Dummy variable for component name.
Accepts a 20-character string.

LOCINP: Location of the component in the input sequence.
Accepts an integer 0 < LOCINP *< total number of components.*

NBLOKINI: Total number of grid blocks initialized.
Accepts an integer 0 < NBLOKINI *< total number of grid blocks.*

Line 44. (INII(I,J), IINIJ(I,J), INIK(I,J), I = 1, NBLOKINI(J))

INII, INIJ, INIK: I,J,K indices of the initialized grid blocks.
Accepts an integer 0 < INII, INIJ, INIK *< total number of grid blocks in x ,y, z direction, respectively.*

END DO

DO J = 1, NINITCOMP

Line 45. NAME(J)

NAME: Dummy variable for component name.
Accepts a 20-character sting.

Line 46. CONCIJKINI(INII(I,J), IINIJ(I,J), INIK(I,J), I = 1, NBLOKINI(J))

CONCIJKINI: Initialized concentration, kg/s

Matrix Colloid Transport Scaling Factors Input

Line 47. Descriptor: Colloidal transport scaling factors
Accepts a character string up to 100 characters in length as a descriptive line.

If colloidal transport required, then:

Line 48. Descriptor: Scaling factor for velocities in x-direction.
Accepts a character string up to 100 characters in length as a descriptive line.

Line 49. (SCALEX(I,J,K),I=1,NX),J=1,NY),K=1,NZ)

SCALEX: Scaling factors of x-direction velocities.
Accepts an integer ≥ 1.

Line 50. Descriptor: Scaling factor for velocities in y-direction.
Accepts a character string up to 100 characters in length as a descriptive line.

Line 51. (SCALEY(I,J,K),I=1,NX),J=1,NY),K=1,NZ)
SCALEY: Scaling factors of y-direction velocities.
Accepts an integer ≥ 1.

Line 52. Descriptor: Scaling factor for velocities in z-direction.
Accepts a character string up to 100 characters in length as a descriptive line.

Line 53. (SCALEZ(I,J,K),I=1,NX),J=1,NY),K=1,NZ)
SCALEZ: Scaling factors of z-direction velocities.
Accepts an integer ≥ 1.

6.5.1.C7 Dual-Porosity or Dual-Permeability Input

If the medium is dual-porosity or dual-permeability, then:

Input Section 6.5.1C5, lines 1 to 53 and Section 6.5.1C6, lines 1 to 53 following the same order and conditions mentioned above.

6.5.2 ASCII Flux-Field Input File (homogeneous)

An example of NUTS's ASCII flux-field input file for homogeneous properties is given in Appendix C.

Line 1. Descriptor Is the input homogeneous (T/F)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 2. DUMP
DUMP: Logical flag to specify whether the following input is homogeneous. The value of this parameter is important only for CDB runs.
Accepts T or F.

6.5.2A Title Information

Line 3.Descriptor: Test run title
Accepts a character string up to 100 characters in length as a descriptive line.

Line 4.TEST_TITLE
TEST_TITLE: This is a descriptive line of the test run. The line is saved and concatenated with NUTS's title when printed out to give an idea about both NUTS and test runs considered.
Accepts up to a 100-character string.

6.5.2B Time Information

Line 5.Descriptor: Time step size and number of time steps
Accepts a character string up to 100 characters in length as a descriptive line.

Line 6.DT, NTIMESTEP

DT: Time step size (s).
Accepts a real number > 0.

NTIMESTEP: Number of time steps required to run NUTS.
Accepts an integer > 0.

6.5.2C Dimension Information

Line 7. Descriptor: Number of grid blocks in x, y, z
Accepts a character string up to 100 characters in length as a descriptive line.

Line 8.NX, NY, NZ

NX: Number of grid blocks in x-direction.
Accepts an integer > 1.

NY: Number of grid blocks in y-direction.
Accepts an integer > 1.

NZ: Number of grid blocks in z-direction.
Accepts an integer > 1.

Line 9.Descriptor: Dimension of grid blocks in x, y, z
Accepts a character string up to 100 characters in length as a descriptive line.

Line 10. DXTEST, DYTEST, DZTEST

DXTEST: Dimension of the grid block in x-direction (m).
Accepts a real number > 0.

DYTEST: Dimension of the grid block in y-direction (m).
Accepts a real number > 0.

DZTEST: Dimension of the grid block in z-direction (m).
Accepts a real number > 0.

6.5.2D Fracture Information

If the medium to be modeled is fractured, then:

Line 11. Descriptor: Fracture porosity and saturation
Accepts a character string up to 100 characters in length as a descriptive line.

Line 12. PHIFINT, SWFINT

PHIFINT: Initial porosity of the fracture (fraction).
Accepts a real number $0 \leq PHIFINT \leq 1$.

SWFINT: Initial saturation of the fracture (fraction).
Accepts a real number $0 \leq SWFINT \leq 1$.

Line 13. Descriptor: Fracture Constant velocity field
Accepts a character string up to 100 characters in length as a descriptive line.

Line 14. TVELXF, TVELYF, TVELZF

TVELXF:	Fracture velocity of the phase in x-direction (m/s). *Accepts a real number TVELXF < 0.6*
TVELYF:	Fracture velocity of the phase in y-direction (m/s). *Accepts a real number TVELYF < 0.6*
TVELZF:	Fracture velocity of the phase in z-direction (m/s). *Accepts a real number TVELZF < 0.6*

Line 15. Descriptor: Block temperature and brine viscosity
Accepts a character string up to 100 characters in length as a descriptive line.

Line 16. TTEMP_FRAC, TVISW_FRAC

TTEMP_FRAC:	Temperature of the grid block (K). *Accepts a real number > 273.15*
TVISW_FRAC:	Fracture brine viscosity (Pa/s). *Accept a real number > 0*

6.5.2E Matrix Information

If the matrix to be modeled is a continuum, then:

Line 17. Descriptor: Matrix porosity and saturation
Accept character string up to 100 chars as a descriptive line

Line 18. PHIMINT, SWMINT

PHIMINT:	Initial porosity of the matrix (fraction). *Accepts a real number 0 ≤ PHIMINT ≤ 1*
SWMINT:	Initial saturation of the matrix (fraction). *Accepts a real number 0 ≤ SWMINT ≤ 1*

Line 19. Descriptor: Matrix constant velocity field
Accept character string up to 100 chars as a descriptive line

Line 20. TVELXM, TVELYM, TVELZM

TVELXM:	Matrix velocity of the phase in x-direction (m/s). *Accepts a real number TVELXM < 0.6*
TVELYM:	Matrix velocity of the phase in y-direction (m/s). *Accepts a real number TVELYM < 0.6*
TVELZM:	Matrix velocity of the phase in z-direction (m/s). *Accepts a real number TVELZM < 0.6*

Line 21. Descriptor: Block temperature and brine viscosity
Accepts a character string up to 100 characters in length as a descriptive line.

Line 22. TTEMP_MAT, TVISW_MAT

TTEMP_MAT:	Temperature of the grid block (K). *Accepts a real number > 273.15.*
TVISW_MAT:	Matrix brine viscosity (Pa/s). *Accepts a real number > 0.*

6.5.2F Dual-Porosity/Dual-Permeability Information

If a dual-porosity or dual-permeability medium is to be modeled, then:

Line 23. Descriptor: Shape factor and the transfer function
Accepts a character string up to 100 characters in length as a descriptive line.

Line 24. TSIGMA, TTAUW

TSIGMA:	Shape factor (m^2). *Accepts a real number > 0.*
TTAUW:	Fracture/matrix transfer function (m^3/s) *Accepts a real number > 0.*

M

6.5.3 ASCII Flux-Field Input File (heterogeneous)

An example of NUTS's ASCII flux-field input file for heterogeneous properties is given in Appendix D.

Line 1. Descriptor: Is the input homogeneous (T/F)
Accepts a character string up to 100 characters in length as a descriptive line.

Line 2. DUMP

DUMP:	Logical flag to specify whether the following input is homogeneous. The value of this parameter is important only for CDB runs. *Accepts T or F.*

6.5.3A Title Information

Line 3. Descriptor: Test run title
Accepts a character string up to 100 characters in length as a descriptive line.

Line 4. TEST_TITLE

TEST_TITLE:	This is a descriptive line of the test run. The line is saved and concatenated with NUTS title when printed out to give an idea about both NUTS and test runs considered. *Accepts up to a 100-character string.*

6.5.3B Time Information

Line 5. Descriptor: Time step size and number of time steps
Accepts a character string up to 100 characters in length as a descriptive line.

Line 6. DT, NTIMESTEP
DT: Time step size (s).
Accepts a real number > 0.
NTIMESTEP: Number of time steps required to run NUTS.
Accepts an integer > 0.

M

6.5.3C Dimension Information

Line 7. Descriptor: Number of grid blocks in x, y, z
Accepts a character string up to 100 characters in length as a descriptive line.

Line 8. NX, NY, NZ
NX: Number of grid blocks in x-direction.
Accepts an integer ≥ 1.
NY: Number of grid blocks in y-direction.
Accepts an integer ≥ 1.
NZ: Number of grid blocks in z-direction.
Accepts an integer ≥ 1.

Line 9. Descriptor: Dimension of grid blocks in x-direction
Accepts a character string up to 100 characters in length as a descriptive line.

Line 10. ((DX3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
DX3D: Dimension of the grid block in x-direction (m).
Accepts a real number > 0

Line 11. Descriptor: Dimension of grid blocks in y-direction
Accepts a character string up to 100 characters in length as a descriptive line.

Line 12. ((DY3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
DY3D: Dimension of the grid block in y-direction (m).
Accept a real number > 0

Line 13. Descriptor: Dimension of grid blocks in z-direction
Accepts a character string up to 100 characters in length as a descriptive line.

Line 14. ((DZ3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
DZ3D: Dimension of the grid block in z-direction (m).
Accepts a real number > 0.

6.5.3D Fracture Information

If the continuum to be modeled is fractured, then:

Line 15. Descriptor: Fracture porosities
Accepts a character string up to 100 characters in length as a descriptive line.

Line 16. ((PHIFINT3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
PHIFINT3D: Porosity of the fracture (fraction).
Accepts a real number $0 \leq PHIFINT \leq 1$.

Line 17. Descriptor: Fracture saturations
Accepts a character string up to 100 characters in length as a descriptive line.

Line 18. ((SWFINT3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
SWFINT3D: Saturation of the fracture (fraction).
Accepts a real number $0 \leq SWFINT \leq 1$.

Line 19. Descriptor: Fracture velocity field in x-direction
Accepts a character string up to 100 characters in length as a descriptive line.

Line 20. ((VELWXF3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
VELWXF3D: Fracture velocity of the phase in x-direction (m/s).
Accepts a real number $VELWXF3D < 0.6$.

Line 21. Descriptor: Fracture velocity field in y-direction
Accepts a character string up to 100 characters in length as a descriptive line.

Line 22. ((VELWYF3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
VELWYF3D: Fracture velocity of the phase in y-direction (m/s).
Accepts a real number $VELWYF3D < 0.6$.

Line 23. Descriptor: Fracture velocity field in z-direction
Accepts a character string up to 100 characters in length as a descriptive line.

Line 24. ((VELWZF3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
VELWZF3D: Fracture velocity of the phase in z-direction (m/s).
Accepts a real number $VELWZF3D < 0.6$.

Line 25. Descriptor: Fracture grid block brine temperature
Accepts a character string up to 100 characters in length as a descriptive line.

Line 26. ((FTEMP3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
FTEMP3D: Temperature of the brine in the grid block (K).
Accepts a real number > 273.15.

Line 27. Descriptor: Fracture grid block brine viscosity.
Accepts a character string up to 100 characters in length as a descriptive line.

Line 28. ((FVIS3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)

FVIS3D: Fracture brine viscosity (Pa/s).
Accepts a real number > 0.

6.5.3E Matrix Information:

If the matrix to be modeled is a continuum, then:

Line 29. Descriptor: Matrix porosities
Accepts a character string up to 100 characters in length as a descriptive line.

Line 30. ((PHIMINT3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
PHIMINT3D: Porosity of the matrix (fraction).
Accepts a real number 0 < PHIMINT < 1

Line 31. Descriptor: Matrix saturations
Accepts a character string up to 100 characters in length as a descriptive line.

Line 32. ((SWMINT3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
SWMINT3D: Saturation of the matrix (fraction).
Accepts a real number 0 < SWMINT < 1.

Line 33. Descriptor: Matrix velocity field in x-direction
Accepts a character string up to 100 characters in length as a descriptive line.

Line 34. ((VELWXM3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
VELWXM3D: Matrix velocity of the phase in x-direction (m/s).
Accepts a real number TVELXM3D < 0.6.

Line 35. Descriptor: Matrix velocity field in y-direction
Accepts a character string up to 100 characters in length as a descriptive line.

Line 36. ((VELWYM3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
VELWYM3D: Matrix velocity of the phase in y-direction (m/s).
Accepts a real number TVELYM3D < 0.6.

Line 37. Descriptor: Matrix velocity field in z-direction
Accepts a character string up to 100 characters in length as a descriptive line.

Line 38. ((VELWZM3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
VELWZM3D: Matrix velocity of the phase in z-direction (m/s).
Accepts a real number TVELZM3D < 0.6

Line 39. Descriptor: Matrix Block temperature
Accepts a character string up to 100 characters in length as a descriptive line.

Line 40. ((XMTEMP3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)
XMTEMP3D: Matrix Block brine temperature (K).
Accepts a real number > 273.15

Line 41. Descriptor: Matrix Block brine viscosity

Accepts a character string up to 100 characters in length as a descriptive line.

Line 42. ((XMVIS3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)

XMVIS3D: Matrix brine viscosity (Pa/s).
Accepts a real number > 0.

6.5.3F Dual-Porosity/Dual-Permeability Information

If medium to be modeled has dual-porosity or dual-permeability, then:

Line 43. Descriptor: Shape factor
Accepts a character string up to 100 characters in length as a descriptive line.

Line 44. ((SIGMA3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)

SIGMA3D: Shape factor (m^2).
Accepts a real number > 0.

Line 45. Descriptor: Transfer function
Accepts a character string up to 100 characters in length as a descriptive line.

Line 46. ((TAUW3D(I,J,K),I=1,NX),J=1,NY),Z=1,NZ)

TAUW3D: Fracture/matrix transfer function (m^3/s)
Accepts a real number > 0.

6.5.4 Binary Flux-Field Input File

This file is the binary BRAGFLO output. It is read only for the single porosity matrix. It provides NUTS with the following variables:

a. QA information
b. Time information
c. Grid blocks dimension and volumetric information
d. Grid block saturation, pressure, velocity or fluxes field, and fluid densities

A full description of this file and the variables printed is available in BRAGFLO User's Manual.

6.5.5 Binary Output Files

6.5.5A Binary Output Files (BRAGFLO Style)

This NUTS binary output is meant to be post-processed with the results of BRAGFLO. Therefore, the sequence and type of information in this file are in harmony with BRAGFLO's binary output (for a detailed description, the reader is referred to BRAGFLO User's Manual). The post-processor used for this objective is BINCOMBO_NV.

6.5.5B Binary Output Files (NUTS Style)

To illustrate the structure of NUTS's output and to enable the reader to post-process its results, a FORTRAN program that reads NUTS's binary output file is given in Appendix K. The first part of the output, nested by an IF(IFLAGTIME .EQ. 0) statement, contains QA information, input echoes, and variable initializations. This information is output to the binary file once, at time zero. Temporally-varying quantities are an exception to the once-at-time-zero rule. They may be reported additionally at subsequent time steps, but only variables specified in the input controls will be written out at subsequent times.

6.5.6 ASCII Output Files

6.5.6A Extensive ASCII Output File

This file contains information similar to that in the NUTS-style binary output file. An illustrative example of NUTS's extensive ASCII output file is given in Appendix E.

6.5.6B Specific-Ranges ASCII Output File

The purpose of this file is to produce an ASCII output file for output variables of interest to the user, and in specified spatial ranges. Its feature can be activated from the input file together with the ASCII file option.

6.5.7 Debug (Scratch) ASCII Output File

This feature produces a scratch or debug file in ASCII format that identifies the beginning and the end of the major procedures in NUTS, and writes the main variables associated with those procedures. The file stays open for only one time step. Therefore, it will echo some of the input data and the time-dependent variables at the first time step. The purposes of this output are (1) to facilitate debugging, and (2) to enable verification of numerical calculations.

6.6 Generating an Executable for NUTS-SA:

A command file to build NUTS-SA's executable is provided in Appendix F. This command file can build either a debug executable or a normal executable. In the building process, the main program NUT_MAIN.FOR will be compiled and will be linked to the NUT_LIB_STANDALONE.FOR.

6.7 Input/Output (I/O) and Other Files Used by NUTS-CC

Being a subset of NUTS-SA, NUTS-CC uses the following I/O files:

1. User Interactive Input

2. Parameter Statements
3. Input/Output Files

6.7.1 User-Interactive Input

A limited number of control flags are provided by the user while executing the program. These controls are part of the input and mainly specify the following:

a) Name of the flux fields filename. *Accepts an 80-character string.*
b) Type of the run, whether it is a test. *Accepts Y or N.*
c) Type of the output file (ASCII/Binary). *Accepts ASC, BIN or ASC-BIN.*
d) NUTS input filename. *Accepts an 80-character string.*
e) NUTS output filename. *Accepts an 80-character string.*
f) NUTS output debug filename. *Accepts an 80-character string.*
g) Type of fluid. *Accepts L o r G.*
h) BRAGFLO ASCII input filename (*.INP) if the answer in b) is Yand CANCEL if the answer in b) is N. *Accepts an 80 character string.*
i) Will NUTS input or the data base be used to provide chemical constituent properties and rock properties. *Accepts N for NUTS or CDB for the data base.*
j) Properties CDB when the answer in i) is CDB and CANCEL otherwise. *Accepts an 80 character string.*
k) NUTS udisturbed scenario CDB output file name used to initialized certain kind of computation. This kind of initialization is required only for intrusion time other than that used in BRAGFLO. *Accepts NUTS undisturbed scenario CDB output file name for intrusion calculation and up to 80 character or CANCEL otherwise.*

An example of such an interactive input included in a command file is as follows:

```
$ NUTS -                          ! The Executable
   TEST_CON_DISP_DEC.IN -         !Test Flux Field Input File
   Y -                            !Test Run? (y)
   ASC -                          !ASC/BIN Output
   NUT_CON_DISP_DEC.IN -          !Nuts Input File
   NUT_CON_DISP_DEC.OUT -         !NUTS Output ASCII File
   NUT_CON_DISP_DEC.DBG -         !DEBUG File
   L -                            !Liquid/Gas
   CANCEL -                       !(Would have been BRAGFLO.INP)
   N -                            !Read Nuclide Data from NUTS Input
   CANCEL -                       ! Properties CDB input file
   CANCEL                         ! NUTS undisturbed scenario output file
$ EXIT
```

6.7.2 NUTS-CC Parameter Statement

NUTS-CC uses the same parameter statement (NUT_PARAM.INC) used by NUTS-SA. In addition, the CAMCON SYSTEM requires additional parameters. These parameters are included in NUT_CDBXFER.INC.

6.7.3 Other Files Used by NUTS-CC

NUTS_CC accesses different combinations of files in each run depending on user specifications. These files can be classified as input, output, or debug files. The names of these files are provided in the interactive input given earlier in this chapter. A specific combination or all of the following files can be opened simultaneously:

1. ASCII radioisotope input file

2. Either (A) ASCII flux fields input files
   a. Homogenous properties file
   b. Heterogeneous properties file

3. Or (B) Binary flux-field input files (CDB files from BRAGFLO's output), and BRAGFLO ASCII input file associated with this case.

4. Binary (CDB) output files

5. Extensive ASCII output file.

6. Properties CDB input file.

7. NUTS undisturbed scenario output file for initialization of well bore intrusion calculations (the same output of item 4 above).

8. Debug (scratch) ASCII output file.

The files in items 1, 2, 5 and 8 are similar to those used in NUTS-SA. The only difference between the two is in the way the binary files are treated. In NUTS-SA, if two ASCII files are used as inputs (one for the nuclides and rock properties, and the other for the flux fields), then a binary output file can be generated. However, in NUTS-CC, a binary CDB is generated only if the flux fields are read from a binary CDB.

6.7.4 Binary (CDB) Flux-Field and the Associated ASCII Input Files

NUTS-CC uses this file to read the output from BRAGFLO. Time = zero information will be read first. It includes some QA information, the geometrical description of the problem, and initial saturation and porosity data. At each timestep thereafter, brine pressure (PRESBRIN), porosity (POROS), gas saturation (SATGAS), interface volumetric fluxes (FLOWBRX, FLOWBRY), and well fluxes (WELLBRIN) will be read. BRAGFLO ASCII input file is used here to provide NUTS with the material maps information, brine reference density and compressibility, and brine residual saturation.

6.7.5 NUTS's Binary (CDB) Output File

In the binary (CDB) output file, the following variables will be added to the input CDB data:

CNCNJM#:	Concentration of the injected component.
FLUXIM#:	Mass flux at the interface I (the convention is to put the flux at the left interface (i-1) of each grid block (i,j,k) and the sign is positive to the right).
FLUXJM#:	Mass flux at the interface J (the convention is to put the flux at the lower interface (j-1) of each grid block (i,j,k) and the sign is positive in upward direction).
CNDMMS#:	Mass of the condensed component # (has a non-zero value only in gas transport).

Depending on user specifications, concentration or mass of the component(s) will be output to the CDB file and in any form specified by the output array control flags (IPRNTMB (1-14)) as shown in Table 4.

M

Table 4. CDB Output Variables and Their Description

Printing Flag	Description of the Matrix Variable	NUTS CDB output variable name
IPRNTMB(1)	Total mass of dissolved constituents in the brine in each grid block, kg.	SM_DIS
IPRNTMB(2)	Total mass of precipitated constituents in each grid block, kg.	SM_PREC
IPRNTMB(3)	Total mass of dissolved, precipitated and adsorbed constituents in each grid block, kg.	SM_TOT
IPRNTMB(4)	Total dissolved curies of all constituents in each grid block, Ci.	CSM_DIS
IPRNTMB(5)	Total precipitated curies of all constituents in each grid block, Ci.	CSM_PREC
IPRNTMB(6)	Total dissolved, precipitated, and adsorbed curies of all constituents in each grid block, Ci.	CSM_TOT
IPRNTMB(7)	Volumetric concentration of the dissolved isotope, kg/(m^3 brine).	CM#
IPRNTMB(8)	Dissolved mass in the brine of a certain constituent, kg.	BMDISM#
IPRNTMB(9)	Precipitated mass of a certain constituent, kg.	BMPRCM#
IPRNTMB(10)	Soil base concentration of the isotope, mg/(kg soil).	ADPRCNM#
IPRNTMB(11)	Curies of volumetric concentration of a certain constituent, Ci/(m^3 brine).	VOLMCC#
IPRNTMB(12)	Curies of dissolved mass in the brine of a certain constituent, Ci.	DISMMC#
IPRNTMB(13)	Curies of precipitated mass of a certain constituent, Ci.	PRCPMMC#
IPRNTMB(14)	Curies of dissolved, precipitated and adsorbed mass of a certain constituent, Ci.	TOTMMC#

In the above notations, # refers to the location of the component in the input sequence, and M refers to the matrix continuum. Similar notation is also available for the fracture continuum.

6.7.6 NUTS's Properties Binary (CDB) Input File

When the answer to item (j) in the user interactive input (Section 6.7.1) is CDB, NUTS-CC expects to read the isotopes' property from a CDB file instead of NUTS's ASCII file. These properties include but are not limited to the inventories, atomic weight, solubility limit, and half-life of the isotope. Other properties that are not used in the CCA calculations, such as pH of the brine, sorption coefficients, grain densities, molecular diffusion, dispersivities, and tortousities, can also be read from the same file provided that the flag related to the respective module is activated in NUTS's ASCII input (through certain flags, one can activate sorption, dispersion, etc.).

6.8 Generating an Executable for NUTS-CC

A command file that builds NUTS-CC's executable is provided in Appendix G. This command file builds either (i) a debug file or (ii) a normal executable. In the building process, the main program NUT_MAIN.FOR will be compiled and linked with NUT_CDBLIB.FOR, NUT_LIB.FOR, CAMDAT_LIB.FOR, CAMCON_LIB.FOR, and CAMSUPES_LIB.FOR.

6.9 Exercising NUTS from a Command Line

Appendix H provides several examples of command files (*.com) to run NUTS from WIPP's Alpha-VAX cluster.

M

7.0 DESCRIPTION OF NUTS'S INPUT FILES

NUTS's input files are discussed in detail in Sections 6.3.1 (ASCII radioisotope file) and 6.3.2 (ASCII flux-field input file). Appendices A, B, C, and D provide examples of NUTS's ASCII Radioisotope input files, its flux-field input files (homogenous and heterogeneous cases), and a sample material map, all designed to run on the WIPP Alpha-VAX microcomputer cluster.

8.0 ERROR MESSAGES

Selected error messages and steps toward their correction are listed in Section 6.0, at the places where they may arise.

A complete listing of NUTS's error and warning messages is given in Appendix I. Error messages will be written on the screen and in the debug file and will cause NUTS to stop. Warning messages will be written only in the debug file and will generally not result in an abort. General warning messages will appear on the screen if NUTS detects undesirable value(s) in the input data.

9.0 DESCRIPTION OF NUTS'S OUTPUT FILES

NUTS's output is capable of appearing in various user-readable and user non-readable formats, depending on user inputs. To assist the user, examples in both formats are included. The first, which is user readable, is NUTS's normal debug file. An annotated version of that file appears in Appendix J.

The second example is of a user-readable output file, namely, NUTS's Extensive ASCII Output File. It is by far the most complete, user-readable, NUTS output file.

Several of NUTS's important input and output files may, at the user's option, be written in binary format. As a service to users, we have provided a translation program that renders NUTS's binary files readable. That program has not been QAed and consequently it should not be regarded as an official part of NUTS's QA documentation. However, to assist the user, the translation program is included herein as Appendix K.

10.0 REFERENCES

Asfari, A., and P.A. witherspoon (1973). "Numerical Simulation of Naturally Fractured Reservoirs", Paper SPE 4290, presented at the 3rd Numerical Simulation of Reservoir Performance Symposium, Houston, TX, 10-12 Jan.

Bear, J. (1988). "Dynamic of Fluids in Porous Media", Dover Publications, Inc., NY.

Bear, J. (1993) " Flow and Contaminant Transport in Fractured Rock", Academic Press, Inc.

Beckner, B.L., Firoozabadi, A., and K. Aziz (1988). "Modeling Transverse Imbibition in Double-Porosity Simulators", Paper SPE 17414 presented at the SPE California Regional Meeting, Long Beach, CA 23-25 March.

Beckner, B.L., Chan, H.M., McDonald, A.E., Wooten, S.O., and T.A. Jones (1991). "Simulating naturally fractured Reservoirs Using Subdomain Method", paper SPE 21241, presented at the 11th SPE Symposium on reservoir Simulation, Anaheim, CA, 17-20 Feb.

Chen, W.H., Wasserman, M.L., and R.E. Fitzmorris (1987). "A Thermal Simulator for Naturally Fractured Reservoirs", paper SPE 16008 presented at the 9th SPE Symposium on Reservoir Simulation, San Antonio, TX, 1-4, Feb.

Coats, K.H. (1989). " Implicit Compositional Simulation of Single-Porosity and Dual-Porosity Reservoirs", Paper SPE 18427 presented at the 10th SPE Symposium on Reservoir Simulation, Houston, TX, Feb. 6-8, 1989.

Dean, R.H., and L.L. Lo (1988) " Simulations of Naturally Fractured Reservoirs", SPE Reservoir Engineering, 638-648.

DeSwaan, A. (1978). "Theory of Waterflooding in Fractured Reservoirs", SPEJ(117-122.

DeSwaan A.D. (1982). "A Three-Phase Model for Fractured Reservoirs Presenting Fluid Segregation", paper SPE 10510, presented at the 6th SPE Symposium on Reservoir Simulation, New Orleans, LA Jan. 31- Feb. 3.

Doctor, P.G. et al. (1992). "An Example Postclosure Risk Assessment Using the Potential Yucca Mountain Site", Pacific Northwest Laboratory, Washington.

Duguid, J.O., and P.C.Y. Lee (1977). "Flow in Fractured Porous Media", Water Resources Research, 558-566.

+ SPEJ refers to the Society of Petroleum Engineers Journal

Dutra, T.V., and K. Aziz (1991). " A New Double-Porosity Reservoir Model for Oil/Water Flow Problems", paper SPE 21248, presented at the 11th SPE Symposium on reservoir Simulation, Anaheim, CA, 17-20 Feb.

Dykhuizen, R.C. (1990). " A new Coupling Term for Dual-Porosity Models", Water Resources Research, Vol. 26, No. 2, 351-356.

Evans, R.D. (1982) " A Proposed Model for Multi-Phase Flow Through naturally Fractured Reservoir", SPEJ, 669-680.

Gilman, J.R. and H. Kazemi (1983). " Improvement in Simulation of naturally Fractured Reservoirs", SPEJ 695-707.

Gilman, J.R. (1986). " An Efficient Finite-Difference Method for Simulating Phase Segregation in the Matrix Blocks in Double-Porosity Reservoirs", SPE Reservoir Engineering, 403-413.

Gilman, J.R. and H. Kazemi (1988). " Improved Calculations for Viscous and gravity Displacement in Matrix Blocks in Dual-Porosity Simulators", JPT, 60-70.

Hill, A.C. and G.W. Thomas. (1985). "A New Approach for Simulating Complex Fractured Reservoirs", Paper SPE 13537 presented at the 1985 Res. Sim. Sym., Dallas, TX, 10-13 Feb.

Huang_Zhang C. (1983). "Numerical Simulation of Conning Behavior in a Naturally Fractured reservoir", SPEJ, 879-884.

Kazemi, H., L.S. Merrill, K.L. Porterfield and P.R. Zeman. (1976). "Numerical Simulation of Water-Oil Flow in Naturally Fractured Reservoirs", SPEJ 317-326.

Kazemi, H., and L.S. Merrill (1979). "Numerical Simulation of Water Imbibition in Fractured Cores", SPEJ 175-182.

Kazemi, H., Gilman, J.R., and A.M. El-Sharkawy (1989). "Analytical and Numerical Solution of Oil Recovery From Fractured Reservoirs Using Empirical Transfer Functions", paper SPE 19849 presented at the 64th SPE Annual Meeting, San Antonio, TX, 8-11 Oct.

Litvak, B.L. (1985). "Simulation and Characterization of Naturally Fractured Reservoirs", Proc. Res. Characterization Technical Conf., Dallas Academic Press, NY.

Narasimhan, T.N. (1982). "Multi-Dimensional Numerical Simulation of Fluid Flow in Fractured porous Media", Water Resources Research, Vol. 18, No. 4, 1235-1247.

Pruess, K. and T.N. Narasimhan (1985). "A Practical Method for Modeling Fluid and Heat Flow in fractured Porous Media", SPEJ, 15-26.

Rechard, R. P. , A. P. Gilkey, H. J. Iuzzolino, D. K. Rudeen, and K. A. Byle. (1993). Programmer's Manual for CAMCON: Compliance Assessment Methodology Controller. SAND90 - 1984. Sandia National Laboratories, Albuquerque NM.

Rechard, R. P., ed. (1992). User's Reference Manual for CAMCON: Compliance Assessment Methodology Controller; Version 3.0. SAND90 - 1983. Sandia National Laboratories, Albuquerque NM.

Reiss, L.H., Bossie-Codreanu, D.N., and L. DuPrey (1973). "Flow in Fissured Reservoirs", Paper SPE 4343, presented at the 2nd Annual European Meeting of SPE-AIME, London, England, 2-3 April.

Rossen, R.H. (1977). " Simulation of Naturally Fractured Reservoirs with Semi-Implicit Source Terms", SPEJ 201-210.

Saidi, A. (1983). " Simulation of Naturally Fractured Reservoirs", Paper SPE 12270 presented at the SPE Symposium on Reservoir Simulation, San Francisco, 16-18 Nov.

Shinta, A.A. and H. Kazemi. (1993). "Tracer Transport in Characterization of Dual-Porosity Reservoirs", Paper SPE 26636 presented at the 1993 SPE Annual Tech. Conf and Exh., Houston, TX, 3-6 Oct.

Sonier, F., Souillard, P., and F.T. Blaskovich (1986). "Numerical Simulation of Naturally Fractured Reservoirs", SPE Reservoir Engineering, 1114-1122.

Thomas, L.K., Dixon, T.N., and R.G. Pierson (1983). " Fractured Reservoir Simulation", SPEJ, 42-54.

WIPP PA Department (1992). Preliminary Performance Assessment for the Waste Isolation Pilot Plant, December 1992. Volume 1: Third Comparison with 40 CFR 191, Subpart B. SAND92 - 0700/1. Sandia National Laboratories, Albuquerque NM.

WIPP PA Department (1992). Preliminary Performance Assessment for the Waste Isolation Pilot Plant, December 1992. Volume 2: Technical Basis. SAND92 - 0700/2. Sandia National Laboratories, Albuquerque NM

WIPP PA Department (1992). Preliminary Performance Assessment for the Waste Isolation Pilot Plant, December 1992. Volume 3: Model Parameters. SAND92 - 0700/3. Sandia National Laboratories, Albuquerque NM.

WIPP PA Department (1992). Preliminary Performance Assessment for the Waste Isolation Pilot Plant, December 1992. Volume 4: Uncertainty and Sensitivity Analyses for 40 CFR 191, Subpart B. SAND92 - 0700/4. Sandia National Laboratories, Albuquerque NM.

WIPP PA Department (1992). Preliminary Performance Assessment for the Waste Isolation Pilot Plant, December 1992. Volume 5: Uncertainty and Sensitivity Analyses of Gas and Brine Migration for Undisturbed Performance. SAND92 - 0700/5. Sandia National Laboratories, Albuquerque NM.

Wu, Y.S., and K. Pruess (1988). " A Multiple-Porosity Method for Simulation of Naturally Fractured Petroleum Reservoirs", SPE Reservoir Engineering, 327-336.