

UNIVERSITY OF CALIFORNIA

Los Angeles

Composite Multiphase Groundwater Model

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Civil Engineering

by

Joon Hyun Kim

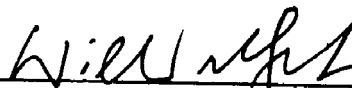
1989

© Copyright by


Joon Hyun Kim

1989

The dissertation of Joon Hyun Kim is approved.




William W-G. Yeh



John A. Dracup



Walter J. Karplus



Donald Carlisle



Michael K. Stenstrom, Committee Chair

University of California, Los Angeles

1989

DEDICATION

To my parents, grandmother, son Insung and wife Kyungbok

CONTENTS

Dedication	iii
List of Figures	vii
Acknowledgments	ix
Vita	x
Abstract of the Dissertation	xii

<u>Chapter</u>	<u>page</u>
I. INTRODUCTION	1
Groundwater Flow and Contamination	1
Purpose and Scope of the study	3
II. LITERATURE REVIEW	5
Macroscopic Volume Averaging	5
Multiphase Flow System	6
Composite Multiphase System	7
Numerical Method	9
Constitutive Equations	12
Chemical and Biological Reaction Process	13
III. MATHEMATICAL FORMULATION	15
Governing Equations	15
Saturated Groundwater Flow	25
Unsaturated Groundwater Flow	26
Solute Transport in Unsaturated System	26
Multiphase Flow System	28
Composite Multiphase Contaminant Migration	30
Constitutive and Partitioning Equations	32
Constitutive Equations	32
Saturation and Relative Permeability	32
Dispersive Coefficient	34
Partitioning Equations	36
Partition between water and oil phase	36
Partition between water and air phase	37

	Partition between solid and water phase	37
	Chemical and Biological Reactions	40
IV.	DIFFICULTIES IN EACH PROCESS	42
	Physical Process	42
	Chemical Process	42
	Biological Process	43
V.	NUMERICAL METHOD	44
	Finite Element Method	44
	Basis Function	51
	Upstream Weighting Function	56
	Boundary Conditions	57
	1st type boundary (known primary variables)	58
	2nd type boundary (known flux at node)	58
	3rd type boundary (mixed type)	60
	Nonlinearity Treatment	61
	Mass Lumping	62
	Iteration Scheme	62
	Element-wise Parameter Evaluation	63
	Stability and Error Analysis of the Algorithm	65
VI.	SIMULATION OF SPECIFIC PROBLEMS	69
	Unsaturated Groundwater Flow System	69
	Mass Transfer System	72
	One Dimensional Upstream Weighted FEM	72
	FEM with Linear Basis Function	74
	FEM with Quadratic Basis Function	76
	FEM with Hermitian Basis Function	79
	FEM with Modified Hermitian Basis Function	83
	Finite Difference Method	84
	Two Dimensional Upstream Weighted FEM	86
	Multiphase Flow System	90
	Composite Multiphase Contaminant Migration	94
	Composite Multiphase Flow	94
	Composite Multiphase Contaminant Transport	97
VII.	CONCLUSIONS AND FUTURE RESEARCH	99
	Unsaturated Groundwater Flow System	100
	Mass Transfer System	100
	Multiphase Flow System	101
	Composite Multiphase Contaminant Migration	102
	Parameter Sensitivity in Simulation	102
	Fluid Conductivity	103
	Saturation	103
	Density	103
	Initial Conditions	103

Boundary Conditions	104
Optimization of Code with Vector and Parallel Processing	104
Future Research	105

LIST OF FIGURES

<u>Figure</u>	<u>page</u>
1. Idealized contaminant migration and REV	16
2. Schematic elementary volume	17
3. Capillary head and saturation from Parker et al.(1987)	33
4. Sorption partition coefficient from Lyman et al.(1982)	38
5. Extrapolation of Hoc from How (Karickhoff, 1984)	39
6. Transformation of global coordinates into local ones	52
7. Linear basis and weighting function	53
8. One dimensional analogue of FEM and FDM	68
9. Problem definition and data from Huyakorn et al.(1986)	70
10. Water table and base pressure head from Huyakorn et al.(1986) ..	71
11. Quadratic basis function	78
12. Hermitian basis function	81
13. Problem data in solute transport system	88
14. Comparison of several basis functions in one and two dimension .	89
15. Data of multidimensional multiphase flow problem	91
16. Result of multidimensional multiphase flow problem	92
17. Parameter dependency of multiphase flow	93
18. Data of composite multiphase contaminant migration problem ...	95
19. Profile of saturation migration of TCE	96

20.	Mass fraction profile of TCE species in water phase	98
-----	---	----

ACKNOWLEDGMENTS

The guidance, support and encouragement of Professor Michael K. Stenstrom are greatly appreciated. I would like to thank Professors William W-G. Yeh, John A. Dracup, Walter J. Karplus and Donald Carlisle for serving on my doctoral committee.

I am grateful for the support of National Science Foundation Engineering Research Center for Hazardous Substances Control, Grant No. 4-442590-21178 and Standard Oil Fellowship, Grant No. 4-782590-40011.

Deborah Haines, Jun Zhou, members of SEASNET, Joan Slattow and Aeint de Boer in the Office of Academic Computing were helpful for the completion of this dissertation. I would like to give thanks to the members of laboratory for their companionship.

Finally, I would like to give special thanks to my parents and wife Kyungbok for their encouragement, patience, support and devotion.

VITA

1958 : Born, Seoul, Korea

1984 : B.S., Seoul National University, Korea

1984 : Engineer, Hyun-Dai Eng. and Construction Co., Seoul, Korea

1984-86 : M.S., University of Michigan, Ann Arbor, Michigan

1986-1989 : Post-Graduate Research Engineer, UCLA

PUBLICATIONS AND PRESENTATIONS

1. Kim, J. H., and M. K. Stenstrom, Comprehensive Groundwater Modeling Code, Poster Sessions, NSF/UCLA Engineering Research Center for Hazardous Substances Control, 1987, 1988, 1989.
2. Kim, J. H. and M. K. Stenstrom, Comprehensive Groundwater Model, Progress Report, NSF/UCLA Engineering Research Center for Hazardous Substances Control, 1988.
3. Kim, J. H., M. K. Stenstrom and J. Zhou, Composite Multiphase Groundwater Model, Poster Sessions, Toxic Substances Research and Teaching Program, Third Annual Research Symposium, Laurel Heights Conference Center, University of California, San Francisco, 1989.
4. Kim, J. H. and M. K. Stenstrom, Composite Multiphase Groundwater Model, Part 1. Theory, Submitted to *Water Resour. Res.*, 1989.
5. Kim, J. H. and M. K. Stenstrom, Composite Multiphase Groundwater Model, Part 2. Simulation, Submitted to *Water Resour. Res.*, 1989.

6. Stenstrom, M. K., J. H. Kim and H. G. Tran, A Dynamic Model for Breakpoint Chlorination, Submitted to *Water Resour Res.*, 1989.

ABSTRACT OF THE DISSERTATION

Composite Multiphase Groundwater Model

by

Joon Hyun Kim

Doctor of Philosophy in Civil Engineering

University of California, Los Angeles, 1989

Professor Michael K. Stenstrom, Chair

A general comprehensive mathematical model using the composite multiphase approach to describe groundwater flow and pollution was developed. The comprehensive governing equation was derived from the simple mass balance of chemical species over all the phases in schematic elementary volume, and traditional groundwater governing equations are explained from it. An attempt was made to include the complicated aspects of physical, chemical and biological processes such as mass fraction, compressibility, capillarity, dispersion, gravity, relative permeability, viscosity, sorption, interfacial mass change and chemical and biological reactions. To make the analysis possible, assumptions have been made for continuous flow of each phase and instantaneous equilibrium for partition. The resulting system of nonlinear governing and constitutive equations was solved numerically.

To handle the irregular geometry, complex boundary conditions and many different governing equations with simple modifications, the upstream weighted finite element method was adopted. By using the dynamic allocation of arrays, the code is flexible to work on an IBM 3090 Vector Facility, workstations and PC's for one, two and three dimensional problems. To reduce the computation time and storage requirements, decoupling of the system equations, banded global matrix and vector and parallel processing were used. The program was structured to facilitate inclusion of additional future constitutive equations. To demonstrate the model's versatility, several hypothetical problems were simulated: unsaturated flow through an embankment; one and two dimensional solute transport; one, two, three dimensional multiphase flow; composite multiphase flow and contaminant migration. The instability and convergence criteria of the nonlinear problems were studied. Parameter dependency of the model was also studied.

The code can simulate the transport and transformation of a complex mixture of groundwater contaminants (e.g. a mixture of light and heavy hydrocarbons, water-born contaminants and volatile contaminants). Investigators will be able to use the model to predict transport rates and fate of contaminants which will facilitate the design of better remediation plans.

Chapter I

INTRODUCTION

1.1 GROUNDWATER FLOW AND CONTAMINATION

According to EPA's groundwater handbook (1987), about 50% of the nation's drinking water is from underground sources. Since the 1970's, the threat of toxic and hazardous waste in underground drinking water has become a public concern. Sources of pollution can be a landfill site, a manufacturing facility, pesticides from agriculture, septic tanks, detergent from domestic use and underground storage tank among others.

The process of groundwater pollution includes complicated physical, chemical and biological aspects. Groundwater flow and migration of chemical constituents need to be studied together. Though limited by its conciseness, the mathematical models are the best choice to express this complex problem. During the past three decades, mathematical models of groundwater systems have been developed from simple groundwater flow to multispecies multiphase flow, expressed in the form of partial and ordinary differential equations. To explain the discontinuous characteristics of the subsurface domain with the fluid characteristics, traditionally many parameters are used with the mass conservation law, such as storage coefficients, saturation and fluid conductivity. Darcy's concept has been used for the momentum balance. To analyze the problem with some degree of accuracy, we need to understand the proc-

esses related to the water flow and contamination in the subsurface region.

Because of the imperfection of our understanding, no present methodology can give a fully accurate solution.

To apply the continuous partial differential equations over the discontinuous subsurface domain, the traditional concept of continuum mechanics is used. With the rapid evolution of computer hardware and software, the solution of engineering problems increasingly relies upon the use of digital computers, and coding plays an important rule in the solution of these mathematical models. A long computation time is required owing to the severe nonlinearity of the given problem, and therefore a supercomputer with vector or parallel processing is required.

In coding, if the algorithm is too complicated, then the computation time will be too long and the adoption of the code to even slightly different cases will be a formidable job. A simple and efficient logic is essential to this problem. In this sense, finite element methods are more suitable than the finite difference methods, because even though the governing equation is different from case to case, the evaluation of spatial derivatives by basis functions does not change. The basis functions depend only on the geometry. Even if the equation is quite long, we just need to add the element matrices for the new spatial derivatives. In the case of the FDM, we have to change the main analogue of the algorithm because of the newly added derivatives. In FEM, because the governing equation is integrated over the spatial domain, each term of the integrated governing equation becomes a flux and the mass balance of the fluxes is maintained better than in the case of FDM. The resulting flux

type expression of the finite element scheme is not only useful for the solution of the mathematical equations, but also for the logical understanding of the complicated processes.

1.2 PURPOSE AND SCOPE OF THE STUDY

The objectives of this study are:

1. To obtain a systematic expression of the governing equations of groundwater flow and pollutant migration, which is suitable for coding and has reasonable parameter evaluation requirements;
2. To develop comprehensive groundwater modeling codes which can handle the important problems in the aquifer domain including chemical and biological reactions;
3. To gain insight into the difficulty and importance of the parameter evaluation and sensitivity analysis;
4. To simplify the data and parameter requirements of the code;
5. To study numerical techniques, such as upstream weighting, variation of time step, several iteration technique, mass lumping, for handling very unstable property of nonlinear partial differential equations.

The outline of this study is as follows:

1. In Chapter 2, previous research on important topics in the modeling, such as governing equations, constitutive equations, parameters, numerical method and simulation examples is reviewed.

2. In Chapter 3, the governing equations of the groundwater flow and pollutant migration are developed from the simple schematic mass balance. By the general governing equation, several groundwater flows and pollutant migration equations are explained, from a simple case to a complicated one. The constitutive equations are also developed from the most current literature and some of them have been modified.
3. In Chapter 4, some of the difficulties in analyzing the groundwater pollution are explained.
4. In Chapter 5, the nonlinear governing equation is solved by using the traditional quadrilateral finite element method. Upstream weighting is used to handle numerical oscillation and dispersion in an advective flux dominant system. To handle the severe nonlinearity of the governing equation, mass lumping and modified Picard iteration techniques are used. The coupled system of several equations is simplified by decoupling and the use of partitioning concepts. Stability and errors are discussed through the stability analysis of the simple one dimensional solute transport equation. Several methods to handle the nonlinear characteristics are introduced.
5. In Chapter 6, traditional groundwater problems are simulated from the simple unsaturated flow case to the complicated composite multiphase contaminant migration case.
6. In Chapter 7, the results of simulation are discussed and some future research is suggested.

Chapter II

LITERATURE REVIEW

The studies in groundwater flow and contamination have evolved from the simple saturated groundwater flow to the complex multiphase multispecies contaminant transport problems. The governing equation of groundwater flow has evolved from the saturated system (one water phase), the unsaturated system (water and air phase) to the multiphase system (water, air, oil) in hydrocarbon related problems.

It is almost impossible to review all the previous research in this area because the field is expanding so rapidly. Furthermore, the field is very broad, requiring knowledge of physics, chemistry and microbiology. In this chapter, even though not all the previous work is reviewed, an attempt has been made to logically explain the work related to the topic of this study in some detail.

2.1 MACROSCOPIC VOLUME AVERAGING

To apply the continuous partial differential equations over the discontinuous subsurface domain, traditional concepts in continuum mechanics are used. The work by Hassanizadeh and Gray (1979) can be considered as one of the earliest. According to them, by defining the averaging properties such as volumetric fraction, the averaging procedure is applied to the microscopic

balance of some property (mass, momentum, heat, energy etc.), so the averaged macroscopic balance equation is acquired.

Bachmat and Bear (1986) presented the criteria for the selection of the size of the representative elementary volume, which is the conceptual elementary volume for the macroscopic averaging process.

2.2 MULTIPHASE FLOW SYSTEM

Many methodologies have been developed in petrochemical engineering to analyze multiphase systems. Odeh (1986) broadly explained the existing mathematical model for the recovery of hydrocarbons. He described the behavior of the pressure and the volume of the three phases: oil or hydrocarbon, water and gas. According to him, three forces (viscous force, Darcy force and gravitational force) govern the distribution of fluids in reservoirs. He classified the simulations as follows: 1) the case of no distinct front (black oil simulator) and 2) the case of distinct moving front (enhanced oil recovery simulator). The governing equations are explained for the above two cases with the constraint conditions. The treatment of the nonlinearity (simultaneous or strongly coupled, sequential or weakly coupled process) is explained. A comparison has been made among the different matrix solvers. He explained some more difficult cases of the simulations. Suggestions for the better numerical techniques have been made as adaptive grid refinement, high order numerical techniques, non-finite difference methods and combination of analytical and numerical methods.

Henry and Metcalf (1983) performed an experiment generating multiphase flow by displacing oil with carbon dioxide. They found the pressure and temperature regions in which multiple phases exist and the reduction of the mobility by the multiple phase generation, which is beneficial for the carbon dioxide flooding of a reservoir.

Osborne and Sykes (1986) simulated the simplified multiphase system with constant air phase pressure. A two dimensional quadrilateral finite element algorithm was used and its accuracy was compared with the preexisting model. The case of the Hyde Park Landfill is simulated with the analysis of parameter uncertainty.

2.3 COMPOSITE MULTIPHASE SYSTEM

Quy and Labrid (1983) developed an enhanced oil recovery model in which they described the physical aspects of dispersion, partition and convection. They used a simple one dimensional finite difference method to solve the system equations sequentially. The system consisted of three phases (aqueous, oleic and microemulsion) and seven components (water, oil, surfactant, alcohol or cosurfactant polymer and two electrolytes of anions and bivalent cations). They compared their results to previously measured laboratory data (Labrid, 1979).

Abriola and Pinder (1985) simulated a multiphase system comprised of two species (a volatile, slightly water soluble compound and a nonvolatile, water insoluble compound). For the deformable soil matrix, they applied mass con-

servation of the soil species. Because of the very low water solubility, mass conservation of the water phase was considered by deriving the equation just in terms of the water related variables. Two more governing equations were derived for two chemical species. Many required empirical relations like effects of matrix and fluid compressibilities, gravity, phase composition, interface mass exchange, capillarity, diffusion, dispersion are described. A hypothetical one dimensional case was simulated using an implicit finite difference method with Newton iteration.

Pinder and Abriola (1986) presented a more general and standard formulation of the composite multiphase governing equation. They employed the partitioning concept and simulated a two dimensional hypothetical saturated system by using finite differences. They used Lin's experimental result (1982) to evaluate saturation and relative permeability. They concluded that more data were required to evaluate the constitutive equations for a wide variety of soil and contaminant types and a more efficient algorithm was required to avoid numerical difficulties.

Corapcioglu and Baehr (1987) derived the governing equation of composite multiphase flow in the unsaturated zone by summing up mass conservation terms in each phase, including source terms for solution, precipitation, condensation, volatilization, desorption, adsorption, biological and chemical reactions. New forms for the dispersion coefficient and partition coefficients were suggested. One dimensional finite difference simulation was performed for a hypothetical case consisting of gasoline comprised of eight constituents

(benzene, toluene, 1-hexane, cyclohexane, n-hexane, aromatics, alkanes, heavy ends).

2.4 NUMERICAL METHOD

All the previous investigators provided numerical methods to solve the governing equations. The procedure can be divided into two types - the finite difference method (FDM) and the finite element method (FEM). A detailed review of the numerical method is beyond the scope of this study. Therefore, only a cursory review is provided here. Because of the nonlinearities and complicated governing equations, the simple standard finite element or finite difference algorithm must be modified. Some of these techniques were discussed previously, and a brief review is provided here.

Pinder et al. (1973) suggested an element-wise evaluation of the parameters in the finite element method by using the basis function to handle the variable property over each element. To illustrate this, they tested a steady state radial flow problem in which transmissivity was a parameter.

Pinder and Shapiro (1979) developed modified Hermitian basis functions by adding asymmetric upstream weighting term to the Hermitian basis function and applied to the solute transport equation.

Faust (1985) simulated one dimensional linear waterflood and two dimensional unsaturated flow using the simplified form of multiphase flow by finite differences and Newton iteration. The simulation shows that dense, low viscosity contaminants travel deeper and faster than lighter contaminants. Fur-

thermore, lighter contaminants do not necessarily form a distinct lens above the water table, even though floating lenses are commonly accepted in conceptual models.

Huyakorn et al. (1986) derived a three dimensional finite element algorithm for variably saturated flow. The element shapes were rectangular and triangular prisms. Instead of the numerical integration of the basis functions, the influence coefficients technique was used. To handle the large number of nodal points, a slice successive overrelaxation procedure was used to solve the matrix.

Huyakorn et al. (1986) suggested the curvilinear finite element method for the solute transport problem. According to their simulation result, it is superior to the standard Galerkin method by at least one order of magnitude. The three dimensional domain is divided into layers and each layer is divided into orthogonal curvilinear elements. In a curvilinear system, the vertical coordinate is unchanged but horizontal coordinates are along the normal direction to streamlines. By using the complex mapping, the irregular curvilinear elements are converted to regular shape. A curvilinear flow net can be obtained through an analytical solution or by a numerical solution of Laplace's equation with conformal transformation. In applying the upstream weighting, numerical dispersion can be reduced owing to the directional property of finite element flow net.

Pelka and Peters (1986) proposed computer-independent program techniques to fully utilize vector and parallel computers. Additional intrinsic vector functions with the Fortran 77 standard and more memory are required. They found that the barriers to vectorization are (1) conditional and branch state-

ments, (2) sequential dependencies, (3) nonlinear and indirect indexing, (4) subroutine calls within loops and (5) recursive operations. To vectorize the program, the scalar variables must be arranged in vectors for the continuous operation and the global matrix is solved in an iterative fashion using a previously developed vectorized solver. The global matrix requires more storage for rectangular shaped matrix. They tested their techniques using a simple three dimensional groundwater flow problem using CDC CYBER 175 and 205 computers. They also provided benchmarks.

Kuppusamy et al. (1987) simulated p-cymene in an unsaturated aquifer using the constitutive formulation by Parker et al. (1987). The governing equation was written in terms of total head.

Kaluarachchi and Parker (1989) described many efficient schemes to handle nonlinearities and to reduce the computational time. The integration of the element matrix was performed analytically so that the code can be used on a personal computer, even though it requires the rectangular shape element. Cooley's iteration method (1983) and mass lumping were used for the nonlinearities. One and two dimensional oil infiltration with varying fluid property were simulated.

The governing system of equations used in this study is more complicated than that used in the previously cited research. Therefore, simple finite element method has been expanded to more general and stable algorithm by the preparation of the following procedures:

1. multidimensional basis and weighting functions
2. element-wise evaluation of parameters

3. modified Picard iteration
4. decoupling of the governing equations
5. evaluation of boundary conditions by basis functions
6. logical expression of variables with dimension, species, phase, direction and nodal point

2.5 CONSTITUTIVE EQUATIONS

To explain the discontinuous characteristic of the subsurface domain with the fluid characteristic, many parameters such as the storage coefficient, saturation, fluid conductivity have been used with the mass conservation law.

Darcy's concept from Allen (1984) has been used for the momentum balance with the parameter fluid conductivity to express the velocity of each phase.

Parker, Lenhard and Kuppusamy (1987) developed the closed form of expressions for the saturation and capillary pressure relations in two and three phase using the effective saturation concept. They derived the expression of relative permeability by extending the method of Van Genuchten. Parker and Lenhard (1987) derived a more general expression for fluid saturation, relative permeability, pressure including residual saturation. Lenhard and Parker (1988) experimentally validated the theory of extending two phase to three phase saturation pressure relationships. Lenhard et al.(1988) performed an experiment of multiphase flow in sandy porous medium by using Soltrol 170 and compared the result to the one dimensional finite element multiphase code.

Plumb and Whitaker (1988) applied the volume averaging concept to acquire the large scale dispersive coefficient which includes the local heterogeneity of the domain. The resulting transport equation has additional terms of the second derivative with respect to the time and mixed space-time derivative.

Parker's approach (1987) is used in this study to evaluate the saturation and relative permeability. The saturation derivative is evaluated analytically from the capillary head.

2.6 CHEMICAL AND BIOLOGICAL REACTION PROCESS

Moltz et al. (1986) simulated microbial growth-degradation by applying modified Monod kinetics with solute transport equations. They assumed microcolonies attached to solid matrix surfaces. The limiting factors of the microbial growth are carbon, energy source (substrate) and oxygen. The one dimensional Eulerian-Lagrangian procedure was used for simulation.

Borden et al. (1986) presented the solute transport equations including microbial reaction for oxygen and hydrocarbon and the transport equation for subsurface microorganisms. They included the effects of microbial kinetics, horizontal dispersion, adsorption and re-aeration. In using the USGS MOC code for the field case simulation, they calibrated the model to the observed chloride distribution and estimated parameters.

Yeh and Tripathi (1989) combined the transport model with the chemical reactions. They used the flow equations, multicomponent transport equations and derived equilibrium chemical equations. To solve the system of equations,

three approaches were studied: (1) mixed differential and algebraic equation, (2) direct substitution, (3) sequential iteration. They recommend the sequential iteration with total analytical concentrations of aqueous components.

Chapter III

MATHEMATICAL FORMULATION

3.1 GOVERNING EQUATIONS

To use the general conservation law, the porous media is idealized as the continuous domain (even though actually it is not) by using the concept of the macroscopic volume averaging (Bear 1979, Hassanizadeh 1979). All possible phases and migration patterns are shown in the localized problem domain and detailed view of the representative elementary volume in Figure 1.

The derivation from this averaging process is explained in detail in Appendix B. The schematic elementary volume is developed to explain the composite multiphase profile and to derive the governing equation physically. The governing equation is derived below without using the volume averaging technique by the schematic elementary volume as in Figure 2.

As shown in Figure 2, the schematic elementary volume is a general multiphase multispecies system expressed by the saturation or volumetric fraction of each phase and by the mass fraction of each species. The mass transfer among the different phases is summed up to be zero, according to Corapcioglu and Baehr (1987). The mass balance of species i over all phases is expressed by

$$\sum_{\alpha=1}^4 (m_x^{i,n+1} - m_x^{i,n}) = \left(\frac{m_x^{i,n} V_x^i - m_x^{i,n} V_{x+\Delta x}^i}{\Delta x} \dots + g_x^i \right) \Delta t \quad (3-1)$$

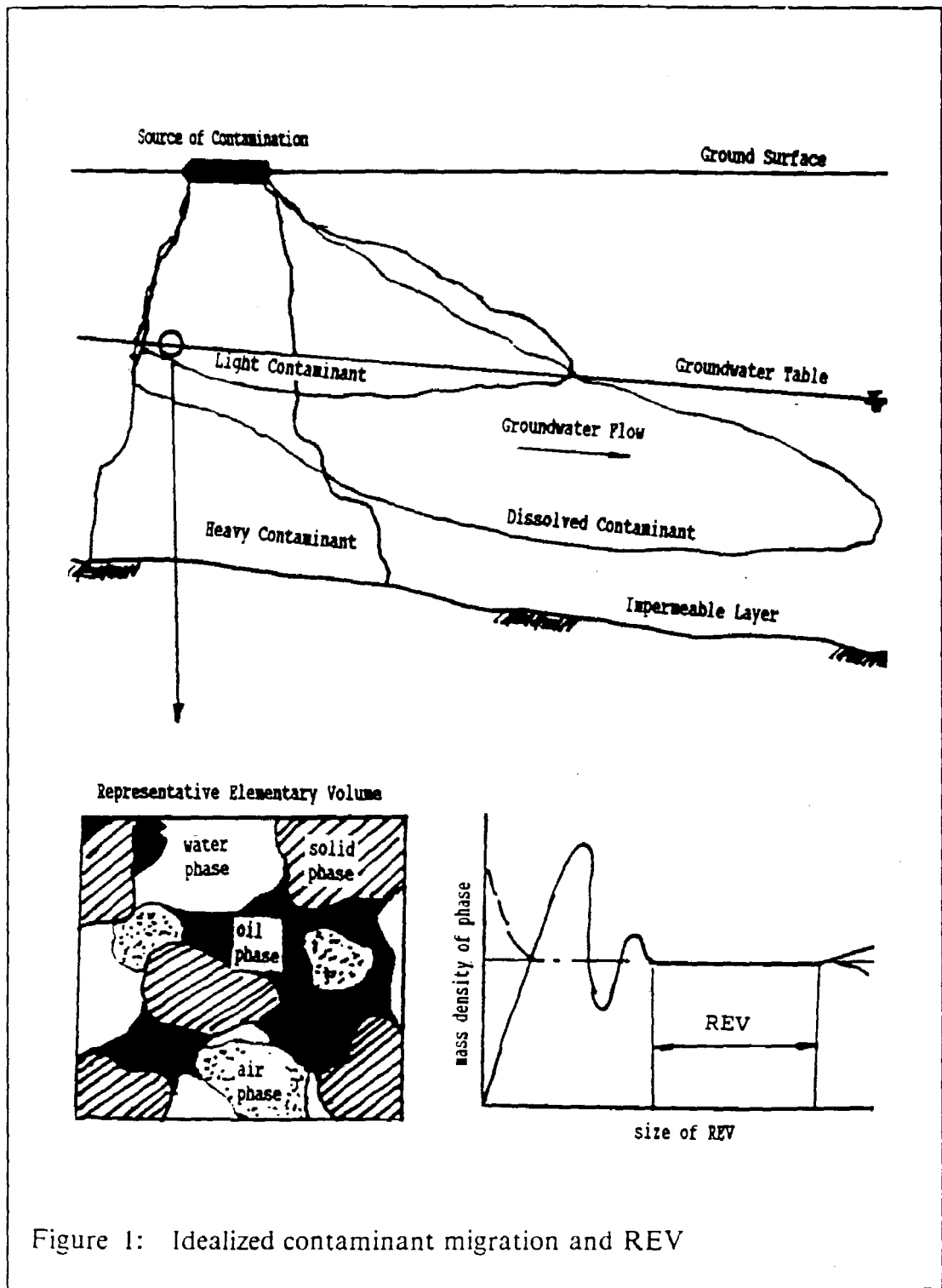


Figure 1: Idealized contaminant migration and REV

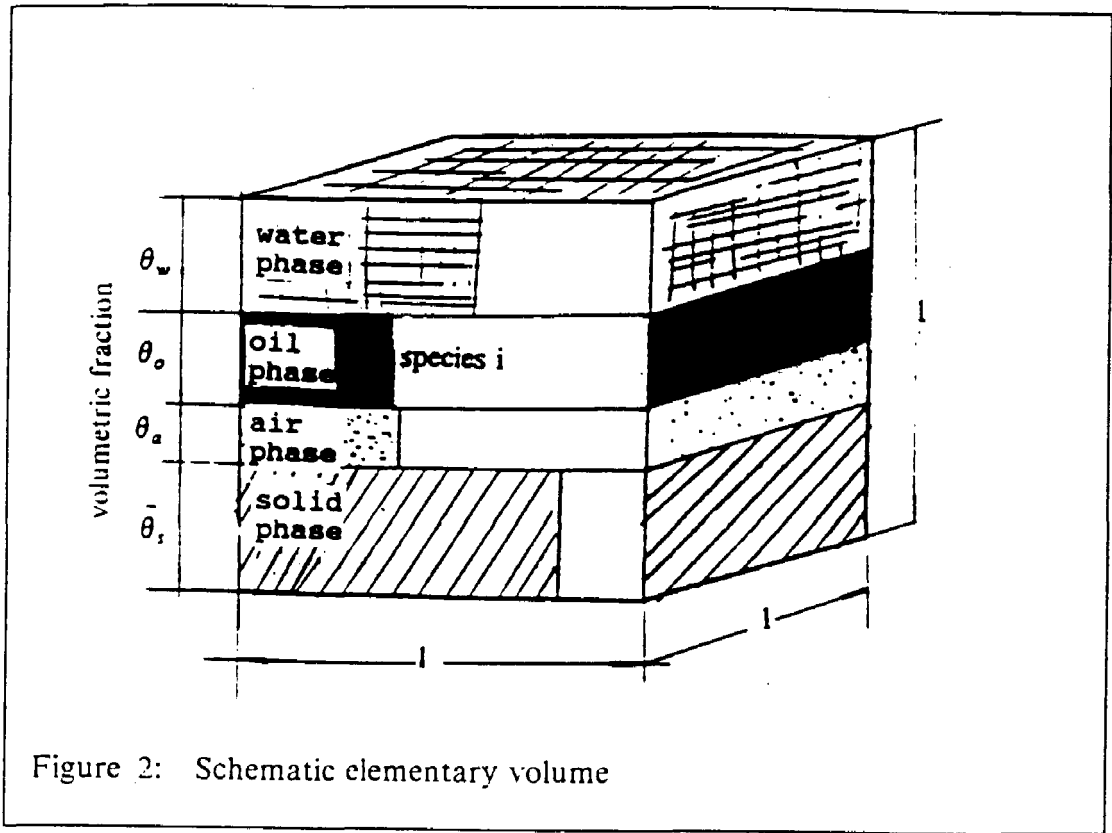


Figure 2: Schematic elementary volume

where,

α = index for phases (water, oil, air, solid phase)

n = index for time step

$$m_x^i = \frac{M_x^i}{U} = \frac{U_x M_x^i}{U U_x} = \theta_x C_x^i = \frac{U_x M_x M_x^i}{U U_x M_x} = \theta_x \rho_x w_x^i$$

U = schematic elementary volume = $\Delta x \Delta y \Delta z$

U_x = volume of α phase

U_v = void volume

$$\phi = \text{porosity} = \frac{U_v}{U}$$

$$\theta_x = \text{volumetric fraction of } \alpha \text{ phase} = \frac{U_x}{U}$$

$$S_x = \text{saturation of } \alpha \text{ phase} = \frac{U_x}{U_v} = \frac{U_x}{\phi U} = \frac{\theta_x}{\phi}$$

M_α = mass of α phase

M_α^i = mass of i species in α phase

G_α^i = production rate of i species in α phase

$$g_\alpha^i = \frac{G_\alpha^i}{U}$$

$$\rho_\alpha = \text{density of } \alpha \text{ phase} = \frac{M_\alpha}{U_\alpha}$$

$$w_\alpha^i = \text{mass fraction of } i \text{ species in } \alpha \text{ phase} = \frac{M_\alpha^i}{M_\alpha}$$

$$C_\alpha^i = \text{concentration of } i \text{ species in } \alpha \text{ phase} = \rho_\alpha w_\alpha^i$$

$$C^i = \text{concentration of } i \text{ species over fluid phases} = S_\alpha C_\alpha^i = S_\alpha \rho_\alpha w_\alpha^i = \frac{m_\alpha^i}{\phi}$$

\vec{V}_α^i = microscopic velocity of i species in α phase

\vec{V}_α = velocity of α phase

If $\Delta x \Delta y \Delta z$ becomes very small, then

$$\sum \left[\frac{\partial m_\alpha^i}{\partial t} + \nabla(m_\alpha^i \vec{V}_\alpha^i) - g_\alpha^i \right] = 0 \quad (3-2)$$

The velocity of i species in α phase is rewritten as, $V_\alpha^i = V_\alpha + V_\alpha^i - V_\alpha$

So,

$$\begin{aligned} m_\alpha^i \vec{V}_\alpha^i &= m_\alpha^i \vec{V}_\alpha + m_\alpha^i (\vec{V}_\alpha^i - \vec{V}_\alpha) = \theta_\alpha C_\alpha^i \vec{V}_\alpha + \theta_\alpha C_\alpha^i (\vec{V}_\alpha^i - \vec{V}_\alpha) \\ &= \theta_\alpha C_\alpha^i \vec{V}_\alpha - \theta_\alpha D_\alpha^i \nabla C_\alpha^i = \phi S_\alpha \rho_\alpha w_\alpha^i \vec{V}_\alpha - \phi S_\alpha D_\alpha^i \nabla(\rho_\alpha w_\alpha^i) \end{aligned} \quad (3-3)$$

where,

D_α^i = dispersive coefficient of i species in α phase

As shown above, the flux difference between the species and phase velocity is expressed as a Fickian form. Here, the dispersive coefficient depends on fluid characteristics and property of porous media. Even though Fickian-type equation is used for the dispersive flux, the dispersive flux is different from the diffusive flux in aquatic domain which only depends on the gradient of concentration.

The velocity of α phase is expressed by Darcy's law as

$$\vec{V}_\alpha = -k \frac{k_{r\alpha}}{\theta_\alpha \mu_\alpha} (\nabla P_\alpha - \rho_\alpha \vec{g}) \quad (3-4)$$

where,

k = intrinsic permeability tensor

$k_{r\alpha}$ = relative permeability of α phase

μ_α = dynamic viscosity of α phase

P_α = pressure of α phase

\vec{g} = gravity vector

So, the governing equation is expressed as

$$\sum \left[\frac{\partial(\theta_\alpha \rho_\alpha w_\alpha^i)}{\partial t} - \nabla(\theta_\alpha \rho_\alpha w_\alpha^i k \frac{k_{r\alpha}}{\theta_\alpha \mu_\alpha} (\nabla P_\alpha - \rho_\alpha \vec{g})) - \nabla(\theta_\alpha D_\alpha^i \nabla(\rho_\alpha w_\alpha^i)) - g_\alpha^i = 0 \right] \quad (3-5)$$

To express the above equation in terms of equivalent water head

$$\begin{aligned}\vec{V}_x &= -\frac{k}{\theta_x \mu_x} \frac{k_{rx}}{\theta_x \mu_x} (\nabla P_x - \rho_x \vec{g}) = -\frac{k}{\theta_x \mu_x} \frac{k_{rx}}{\theta_x \mu_x} (\nabla P_x + \rho_x g \vec{j}) = -\frac{k}{\theta_x \mu_x} \frac{k_{rx} \rho_w g}{\rho_w g} \left(\nabla \frac{P_x}{\rho_w g} + \frac{\rho_x}{\rho_w} \vec{j} \right) \\ &= -\frac{K_x}{\theta_x} \left(\nabla h_x + \frac{\rho_x}{\rho_w} \vec{j} \right) = -\frac{K_x}{\theta_x} \nabla \left(h_x + \frac{\rho_x}{\rho_w} z \right) = -\frac{K_x}{\theta_x} \nabla H_x \quad (3-6)\end{aligned}$$

where,

$$h_x = \frac{P_x}{\rho_w g} = \text{fluid pressure head in equivalent water height}$$

$$H_x = h_x + z \frac{\rho_x}{\rho_w} = \text{total piezometric head in equivalent water height}$$

$$K_x = \frac{k k_{rx}}{\mu_x} \frac{\rho_w g}{\mu_x} = \text{fluid conductivity of } \alpha \text{ phase, } K_\beta = \frac{K_x \mu_x k_{r\beta}}{\mu_\beta k_{rx}}$$

The governing equation is

$$\begin{aligned}\sum_{\alpha=1}^3 \left\{ \frac{\partial(\phi S_\alpha \rho_x w_x^i)}{\partial t} - \nabla(\phi S_\alpha \rho_x w_x^i \frac{K_x}{\phi S_\alpha} (\nabla h_x + \frac{\rho_x}{\rho_w} \vec{j})) \right. \\ \left. - \nabla(\phi S_\alpha D_\alpha^i \nabla(\rho_x w_x^i)) - g_\alpha^i \right\} + \frac{\partial((1-\phi)\rho_s w_s^i)}{\partial t} = 0 \quad (3-7)\end{aligned}$$

Using velocity notation

$$\begin{aligned}\sum_{\alpha=1}^3 \left\{ \frac{\partial(\phi S_\alpha \rho_x w_x^i)}{\partial t} + \nabla(\phi S_\alpha \rho_x w_\alpha^i \vec{V}_\alpha^i) - \nabla(\phi S_\alpha D_\alpha^i \nabla(\rho_x w_x^i)) - g_\alpha^i \right\} \\ + \frac{\partial((1-\phi)\rho_s w_s^i)}{\partial t} = 0 \quad (3-8)\end{aligned}$$

To evaluate the derivative with respect to time

$$\frac{\partial(\phi S_x \rho_x w_x^i)}{\partial t} = w_x^i S_x \frac{\partial(\phi \rho_x)}{\partial t} + \phi \rho_x S_x \frac{\partial w_x^i}{\partial t} + \phi \rho_x w_x^i \frac{\partial S_x}{\partial t} \quad (3-9)$$

The first term of the above equation describes the storage capacity of the given system and is evaluated as follows:

$$\begin{aligned} w_x^i S_x \frac{\partial(\phi \rho_x)}{\partial t} &= w_x^i S_x \left(\rho_x \frac{\partial \phi}{\partial t} + \phi \frac{\partial \rho_x}{\partial t} \right) \\ &= \rho_x w_x^i S_x \sum_{\beta=1}^3 \left(\frac{\partial \phi}{\partial h_\beta} + \frac{\phi}{\rho_x} \frac{\partial \rho_x}{\partial h_\beta} \right) \frac{\partial h_\beta}{\partial t} \end{aligned} \quad (3-10)$$

where,

$$\begin{aligned} \frac{\partial \phi}{\partial h_\beta} &= \text{compressibility of the porosity by } \beta \text{ pressure head} = \phi_\beta \\ \frac{1}{\rho_x} \frac{\partial \rho_x}{\partial h_\beta} &= \text{compressibility of } \alpha \text{ phase by } \beta \text{ pressure head} = \alpha_\beta \\ S_{\alpha\beta} &= \phi_\beta + \phi \alpha_\beta = \text{specific storativity of the porous matrix and fluid} \end{aligned}$$

Therefore, the governing equation becomes,

$$\begin{aligned} \sum_{\alpha=1}^3 \left\{ \sum_{\beta=1}^3 (\rho_\beta w_\beta^i S_\beta S_{\beta\alpha}) \frac{\partial h_\alpha}{\partial t} + \phi \rho_\alpha w_\alpha^i \frac{\partial S_\alpha}{\partial t} + \phi \rho_\alpha S_\alpha \frac{\partial w_\alpha^i}{\partial t} \right\} + \frac{\partial((1-\phi)\rho_s w_s^i)}{\partial t} \\ = \sum_{\alpha=1}^3 \left\{ \nabla(\rho_\alpha w_\alpha^i K_\alpha (\nabla h_\alpha + \frac{\rho_\alpha}{\rho_w} \vec{j})) + \nabla(\theta_\alpha D_\alpha^i \nabla(\rho_\alpha w_\alpha^i)) + g_\alpha^i \right\} \end{aligned} \quad (3-11)$$

The second term of the time derivative is the saturation change which depends on the convective flux and capillary pressure. The expansion of this term can be implemented in two ways. The first case occurs when the dominant driving force of the system is convective flux of fluid phase. The situation occurs in

oil recovery problems, often called as Buckley-Leverett problems (1942). In this case, even though there is no capillary pressure, the saturation is changed by the flux of each phase and the saturation can not be expanded with respect to the capillary pressure. The saturation becomes one primary variable of the system which should be solved simultaneously with the pressure term by the given boundary and initial conditions.

The other case occurs when the dominant driving force is the pressure difference of the each fluid phase. This condition occurs during contaminant migration in a groundwater basin. The time derivative of saturation can be expanded with respect to the capillary pressure, and simultaneous solution of the equation about the saturation term is not required. This assures convergence of the nonlinearity and short computation time. The saturation and its derivative are determined by the constitutive relation of the saturation and capillary pressure. Even when there is no actual capillary phenomena in regions where there is only one phase, a hypothetical saturation derivative is required to avoid singularity of the system of equations. This hypothetical value does not precisely match a real field problem and increases parameter requirements.

In the first case, the governing equation becomes

$$\sum_{\alpha=1}^3 \left\{ \sum_{\beta=1}^3 (\rho_{\beta} w_{\beta}^i S_{\beta} S_{\beta\alpha}) \frac{\partial h_{\alpha}}{\partial t} + \phi \rho_{\alpha} w_{\alpha}^i \frac{\partial S_{\alpha}}{\partial t} + \phi \rho_{\alpha} S_{\alpha} \frac{\partial w_{\alpha}^i}{\partial t} \right\} + \frac{\partial((1-\phi)\rho_s w_s^i)}{\partial t}$$

$$\begin{aligned}
&= \sum_{x=1}^3 \left\{ \nabla(\rho_x w_x^i K_x (\nabla h_x + \frac{\rho_x \vec{j}}{\rho_w})) + \nabla(\rho_x w_x^i K_x \nabla h_{\beta x}) \right. \\
&\quad \left. + \nabla(\theta_x D_x^i \nabla(\rho_x w_x^i)) + g_x^i \right\} \quad (3-12)
\end{aligned}$$

where,

$$h_{\beta\alpha} = h_\beta - h_\alpha = f(S_\alpha) = \text{capillary pressure between } \beta \text{ and } \alpha \text{ phase}$$

In the second case, the governing equation becomes

$$\begin{aligned}
&\sum_{\alpha=1}^3 \left\{ \sum_{\beta=1}^3 (\rho_\beta w_\beta^i S_\beta S_{\beta\alpha} + \phi \rho_\beta w_\beta^i \frac{\partial S_\beta}{\partial h_\alpha}) \frac{\partial h_\alpha}{\partial t} + \phi S_\alpha \rho_x \frac{\partial w_\alpha^i}{\partial t} \right\} + \frac{\partial((1-\phi)\rho_s w_s^i)}{\partial t} \\
&= \sum_{\alpha=1}^3 \left\{ \nabla(\rho_x w_x^i K_x (\nabla h_x + \frac{\rho_x \vec{j}}{\rho_w})) + \nabla(\theta_x D_x^i \nabla(\rho_x w_x^i)) + g_x^i \right\} \quad (3-13)
\end{aligned}$$

As we can see in the above two equations, the primary variables are pressure head, mass fraction and saturation. Even if the second equation is used, we have at least two sets of primary variables. To solve these primary variables we need additional constitutive equations, boundary and initial conditions for each variable. In the case of the saturation variable, the resulting governing equation will be of the hyperbolic type with respect to saturation, and characteristic method gives the best result (Huyakorn and Pinder, 1983), but we can use the upstream weighted finite difference or finite element method to reduce numerical oscillation. Many papers have been written on reducing the instability in Leverett problem in petrochemical engineering. In the case of mass

fraction, the resulting governing equation will be of the parabolic type, which also requires upstream weighting when an advective flux dominates. The simplified form of this mass transport equation is solute transport equation. This equation can be defined as multispecies mass transport equation.

The related parameters of the governing equation are

ρ_α = density of α phase

$k_{r\alpha}$ = relative permeability

μ_α = viscosity of α phase

\underline{D}_α^i = the dispersion coefficient of i species in α phase

g_α^i = internal production rate of i species in α phase

The density, relative permeability, viscosity and dispersion coefficient are computed using the constitutive equations. The internal production rate can be considered as the source term and evaluated using chemical or biological reaction equations.

Specific groundwater systems are explained by the above composite multiphase equation in following sections.

3.1.1 Saturated Groundwater Flow

In a saturated aquifer, only the water phase exists in the void fraction of the porous media, therefore $S_w = 1$, $k_{rw} = 1$.

Only water is present, so $w_w^w = 1$, $\nabla w_w^i = 0$.

The source and sink terms are pumping and recharging flow rates, so

$g_w^w = g_w =$ water pumping (- sign) or recharging (+ sign) rate.

Only water pressure head exists, so the specific storativity or storage coefficient is $S_{ww} = \alpha_w + \phi\beta_w$.

The resulting governing equation is

$$\rho_w S_{ww} \frac{\partial h_w}{\partial t} = \nabla(\rho_w \tilde{K}_w (\nabla h_w + \vec{j})) + g_w \quad (3-14)$$

where,

\vec{j} = upward directional vector in y direction,

ρ_w = density of water phase.

With no change of density in spatial direction, the governing equation simplifies to

$$S_{ww} \frac{\partial h_w}{\partial t} = \nabla(\tilde{K}_w (\nabla h_w + \vec{j})) + \frac{g_w}{\rho_w} \quad (3-15)$$

3.1.2 Unsaturated Groundwater Flow

For simplification, the air phase pressure head is assumed to be atmospheric pressure ($h_a = 0$), and immobile air phase eliminates the need to calculate the air pressure. Therefore, $w_w^w = 1$, $w_a^a = 1$, $\nabla w_a^i = 0$, $S_w + S_a = 1$, $h_{aw} = f(S_w)$

If capillary pressure is the dominant force of the system, then from the second governing equation,

$$(S_w S_{ww} + \phi \frac{\partial S_w}{\partial h_w}) \frac{\partial h_w}{\partial t} = \nabla(K_w(\nabla h_w + \vec{j})) + \frac{g_w}{\rho_w} \quad (3-16-1)$$

$$S_w = f(h_w, h_a) \quad (3-16-2)$$

If convective flux is the dominant force of the system, then from the first governing equation,

$$-S_w S_{ww} \frac{\partial h_{aw}}{\partial t} + \phi \frac{\partial S_w}{\partial t} = -\nabla(K_w(\nabla h_{aw} - \vec{j})) + \frac{g_w}{\rho_w} \quad (3-17-1)$$

$$h_{aw} = f(S_w) \quad (3-17-2)$$

3.1.3 Solute Transport in Unsaturated System

This case exists when all chemical species are totally dissolved in the water phase and are transported by advective water phase motion with the dispersion in water and air phases. All these aspects are included in next solute transport equation. Usually water flow is computed on the first step and the solute migration is computed using water phase velocity. For water flow, the

unsaturated equation can be used. If there is only one species, denoted as o , then we need two equations as follows:

(1). Water Phase Flow Equation (Equation for Water Species)

Assuming that the dissolved chemical species do not affect the flow of the water phase and an immobile air phase, the second approach for the governing equation can be as follows:

$$(S_w S_{ww} + n \frac{\partial S_w}{\partial h_w}) \frac{\partial h_w}{\partial t} = \nabla(\underline{K}_w(\nabla h_w + \vec{j})) + \frac{g_w}{\rho_w} \quad (3-18-1)$$

$$S_w = f(h_{av}) \quad (3-18-2)$$

(2). Solute Transport Equation (Equation for Oil Species)

In the saturated region, the solute migration is expressed by

$$\begin{aligned} \frac{\partial}{\partial t}(\theta_w \rho_w w_w^o + \theta_a \rho_a w_a^o + \theta_s \rho_s w_s^o) &= \nabla(\rho_w w_w^o \underline{K}_w(\nabla h_w + \vec{j})) + \\ &\nabla(\theta_w \underline{D}_w^o \nabla(\rho_w w_w^o)) + \nabla(\theta_a \underline{D}_a^o \nabla(\rho_a w_a^o)) + g_w^o + g_a^o \end{aligned} \quad (3-19)$$

Combining the partitioning concept explained in the section (3.2.2),

$$\begin{aligned} \frac{\partial}{\partial t}(w_w^o(\phi S_w \rho_w + \phi S_a \rho_a H_{av}^o + (1 - \phi) \rho_s H_{sv}^o)) &= \nabla(\rho_w w_w^o \underline{K}_w(\nabla h_w + \vec{j})) + \\ &\nabla(\phi S_w \underline{D}_w^o \nabla(\rho_w w_w^o)) + \nabla(\phi S_a \underline{D}_a^o \nabla(\rho_a H_{av}^o w_w^o)) + g_w^o + g_a^o \end{aligned} \quad (3-20)$$

3.1.4 Multiphase Flow System

This is the case when there is simultaneous immiscible flow of water, air and oil. The governing equations are similar to the equations used in black oil simulation. The primary dependent variables are pressure head and saturation of each phase. The mass fractions in composite multiphase equation become $w_w^w = 1$, $w_o^o = 1$, $w_a^a = 1$ and the derivative of mass fraction is $\nabla w_a^i = 0$. The summation of saturation of each phase is $S_w + S_o + S_a = 1$. The composite multiphase equation is simplified to multiphase equations as follows:

$$\frac{\partial}{\partial t}(\phi S_w \rho_w) = \nabla(\rho_w K_w (\nabla h_w + \vec{j})) + g_w \quad (3-21-1)$$

$$\frac{\partial}{\partial t}(\phi S_o \rho_o) = \nabla(\rho_o K_o (\nabla h_o + \frac{\rho_o}{\rho_w} \vec{j})) + g_o \quad (3-21-2)$$

$$\frac{\partial}{\partial t}(\phi S_a \rho_a) = \nabla(\rho_w K_a (\nabla h_a + \frac{\rho_a}{\rho_w} \vec{j})) + g_a \quad (3-21-3)$$

The last equation is eliminated by assuming an immobile air phase. If the first approach of the governing equation is used, then the resulting governing equation is same as Faust (1985).

$$S_w S_{ww} \frac{\partial h_w}{\partial t} + \phi \rho_w \frac{\partial S_w}{\partial t} = \nabla(\rho_w K_w (\nabla h_o + \vec{j})) - \nabla(\rho_w K_w \nabla h_{ow}) + g_w \quad (3-22-1)$$

$$S_o S_{oo} \frac{\partial h_o}{\partial t} + \phi \rho_o \frac{\partial S_o}{\partial t} = \nabla(\rho_o K_o (\nabla h_o + \frac{\rho_o}{\rho_w} \vec{j})) + g_o \quad (3-22-2)$$

$$S_a = f(h_{aw}), \quad h_{ow} = f(S_w), \quad S_o = 1 - S_w - S_o \quad (3-22-3)$$

The primary variables are S_w , h_o .

The Buckley-Leverett (1942) problem is a simplified form of the above system, which is explained as follows. Assuming no capillary pressure and constant total flow,

$$\frac{\partial}{\partial t}(\phi S_w \rho_w) = -\nabla(\rho_w f_w \vec{q}) \quad (3-23-1)$$

$$f_w = \left(1 + \frac{k_{ro} \mu_w}{k_{rw} \mu_o}\right)^{-1}, \quad k_{rw} = f(S_w), \quad k_{ro} = f(S_w) \quad (3-23-2)$$

where,

\vec{q} = constant total flow $\equiv \vec{q}_w + \vec{q}_o$

f_w = fractional flow function $\rightarrow f(S_w)$

$$k_{rw} = \frac{k_{ro}(S_w)}{1 + \frac{k_{ro} \mu_w}{k_{rw} \mu_o}}$$

As shown above, the convective flux of each phase is decided by the relative permeability and viscosity of each phase, which means the saturation dependency of convective flux.

If capillary pressure is dominant, then the governing equation is same as Kaularachchi and Parker (1989) as follows:

$$(S_w S_{ww} + \phi \rho_w \frac{\partial S_w}{\partial h_{ow}}) \frac{\partial h_w}{\partial t} = \nabla(\rho_w K_w (\nabla h_w + \vec{j})) + g_w \quad (3-24-1)$$

$$(S_o S_{oo} + \phi \rho_o \frac{\partial S_o}{\partial h_{ow}}) \frac{\partial h_o}{\partial t} = \nabla(\rho_o K_o (\nabla h_o + \frac{\rho_o}{\rho_w} \vec{j})) + g_o \quad (3-24-2)$$

3.1.5 Composite Multiphase Contaminant Migration

This is the case when there is simultaneous flow of water, oil and air. The chemical species of each composite phase is changing over time and space. Therefore, the chemical constituent becomes another dependent variable, denoted as mass fraction. The composite multiphase equation cannot be simplified in this case. The pressure head, saturation and mass fraction should be solved simultaneously. However, the governing equation is usually rearranged into two sub-problems; composite multiphase flow and contaminant migration. The multiphase portion is computed first and then the component of each phase is computed.

(1) Composite Multiphase Flow

Primal variables are pressure head and saturation.

(a) 1st approach

$$\sum_{\alpha=1}^3 \left\{ \sum_{\beta=1}^3 (\rho_{\beta} w_{\beta}^i S_{\beta} S_{\beta\alpha}) \frac{\partial h_{\alpha}}{\partial t} + \phi \rho_{\alpha} w_{\alpha}^i \frac{\partial S_{\alpha}}{\partial t} + \phi \rho_{\alpha} S_{\alpha} \frac{\partial w_{\alpha}^i}{\partial t} \right\} + \frac{\partial}{\partial t} ((1 - \phi) \rho_s w_s^i) = \sum_{\alpha=1}^3 \left\{ \nabla(\rho_{\alpha} w_{\alpha}^i K_{\alpha} (\nabla h_{\beta} + \frac{\rho_{\alpha} \vec{j}}{\rho_w})) - \nabla(\rho_{\alpha} w_{\alpha}^i K_{\alpha} \nabla h_{\beta\alpha}) + \nabla(\phi S_{\alpha} D_{\alpha}^i \nabla(\rho_{\alpha} w_{\alpha}^i)) + g_{\alpha}^i \right\} \quad (3-25-1)$$

$$h_{\beta\alpha} = h_{\beta} - h_{\alpha} = f(S_{\alpha}) \quad (3-25-2)$$

(b) 2nd Approach

$$\sum_{\alpha=1}^3 \left\{ \sum_{\beta=1}^3 (\rho_{\beta} w_{\beta}^i S_{\beta} S_{\beta x} + \phi \rho_{\beta} w_{\beta}^i \frac{\partial S_{\beta}}{\partial h_x}) \frac{\partial h_x}{\partial t} + \phi S_{\alpha} \rho_x \frac{\partial w_x^i}{\partial t} \right\} + \frac{\partial}{\partial t} ((1 - \phi) \rho_s w_s^i)$$

$$= \sum_{\alpha=1}^3 \left\{ \nabla(\rho_x w_x^i K_x (\nabla h_x + \frac{\rho_x \bar{v}_j}{\rho_w})) + \nabla(\phi S_{\alpha} D_{\alpha}^i \nabla(\rho_x w_x^i)) + g_{\alpha}^i \right\} \quad (3-26-1)$$

$$S_x = f(h_w, h_o, h_a) \quad (3-26-2)$$

(2) Composite Multiphase Contaminant Transport

The primary variable is mass fraction of species o in water phase. Arranging the governing equation with respect to this primal variable with partitioning concept,

$$\frac{\partial}{\partial t} (w_w^o (\phi S_w \rho_w + \phi S_o \rho_o H_{ow}^o + \phi S_a \rho_a H_{aw}^o + (1 - \phi) \rho_s H_{sw}^o))$$

$$= - \left\{ \nabla(\phi S_w \rho_w w_w^o \bar{V}_w) + \nabla(\phi S_o \rho_o H_{ow}^o w_w^o \bar{V}_o) + \nabla(\phi S_a \rho_a H_{aw}^o w_w^o \bar{V}_a) \right\}$$

$$+ \nabla(\phi S_w D_w^o \nabla(\rho_w w_w^o)) + \nabla(\phi S_o D_o^o \nabla(\rho_o H_{ow}^o w_w^o))$$

$$+ \nabla(\phi S_a D_a^o \nabla(\rho_a H_{aw}^o w_w^o)) + g_w^o + g_o^o + g_a^o \quad (3-27)$$

where, H_{ow}^o , H_{aw}^o , H_{sw}^o = partitioning coefficients.

3.2 CONSTITUTIVE AND PARTITIONING EQUATIONS

3.2.1 Constitutive Equations

3.2.1.1 Saturation and Relative Permeability

From Parker et al. (1987), using the Van Genuchten scheme, the effective saturation of each phase is defined as

$$S_w^e = \frac{S_w - S_{rw}}{S_{sw} - S_{rw}} = \text{effective saturation of water phase}$$

$$S_t^e = \frac{S_t - S_{rt}}{S_{st} - S_{rt}} = \text{effective saturation of total phase (water + oil phase)}$$

$$S_t = S_w + S_o = \text{water phase saturation + oil phase saturation}$$

The saturation of each phase is

$$S_w = S_{rw} + (S_{sw} - S_{rw})S_w^e, \quad S_t = S_{rt} + (S_{st} - S_{rt})S_t^e$$

$$S_o = S_t - S_w, \quad S_a = 1 - S_t$$

$$S_{\alpha} = \text{saturated saturation of } \alpha \text{ phase}$$

$$S_{r\alpha} = \text{residual saturation of } \alpha \text{ phase}$$

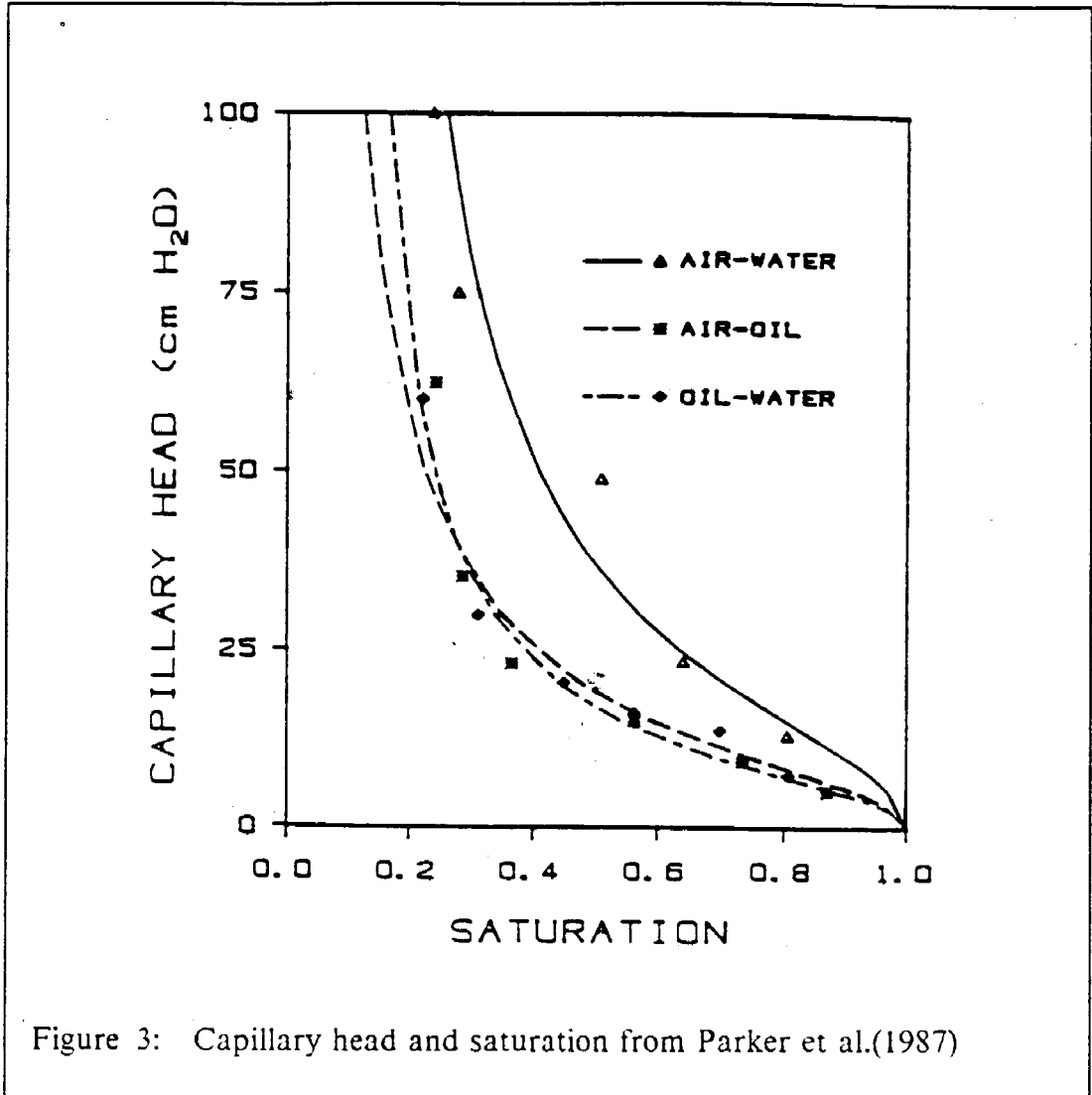
The effective saturations are determined by the capillary heads as,

$$S_w^e = (1 + (\alpha_{ow} h_{ow})^{sn})^{-sm} \quad \text{for } h_{ow} > 0, \quad = 1 \quad \text{for } h_{ow} \leq 0 \quad (3-28-1)$$

$$S_t^e = (1 + (\alpha_{ao} h_{ao})^{sn})^{-sm} \quad \text{for } h_{ao} > 0, \quad = 1 \quad \text{for } h_{ao} \leq 0 \quad (3-28-2)$$

$$\text{where, } sm = 1 - \frac{1}{sn}.$$

In Figure 3, the relation between the capillary head and saturation is plotted.



The relative permeability of each phase is defined from the effective saturation as follows:

$$k_{rw} = (S_w^e)^{1/2} \{1 - (1 - (S_w^e)^{1/m})^m\}^2 \quad (3 - 29 - 1)$$

$$k_{ro} = (S_t^e - S_w^e)^{1/2} \{ (1 - (S_w^e)^{1/m})^m - (1 - (S_t^e)^{1/m})^m \}^2 \quad (3 - 29 - 2)$$

3.2.1.2 Dispersive Coefficient

The dispersive flux is a result of the extension of species flux to phase flux as $\theta_\alpha \rho_\alpha w_\alpha^i \bar{V}_\alpha^i = \theta_\alpha \rho_\alpha w_\alpha^i (\bar{V}_\alpha + (\bar{V}_\alpha^i - \bar{V}_\alpha)) = \theta_\alpha \rho_\alpha w_\alpha^i \bar{V}_\alpha - \theta_\alpha \underline{D}_\alpha^i \nabla (\rho_\alpha w_\alpha^i)$.

Dispersive flux is the difference between the microscopic species phase flux and the averaged phase flux, which is caused by the species velocity fluctuation from the averaged phase velocity. The result of the dispersion is similar to the diffusion of chemical species and traditional Fickian-type flux is used for the evaluation of the diffusion. The diffusive flux is caused by the difference of solute concentration. The dispersive flux is caused by the microscopic velocity fluctuation, and the difference in chemical concentration. The dispersion coefficient has directional property of velocity of each phase and can be different for each species.

Extending the approach from Sutra (1984), the dispersion coefficient tensor of i species in α phase is expressed as

$$D_{\alpha xx}^i = \frac{1}{V_\alpha^2} (d_{xl}^i V_{\alpha,x}^2 + d_{xt}^i V_{\alpha,y}^2), \quad D_{\alpha yy}^i = \frac{1}{V_\alpha^2} (d_{xl}^i V_{\alpha,x}^2 + d_{xt}^i V_{\alpha,y}^2) \quad (3 - 30 - 1)$$

$$D_{\alpha ij}^i = \frac{1}{V_\alpha^2} (d_{xl}^i - d_{xt}^i) V_{\alpha,i} V_{\alpha,j} \quad \text{for } i = x,y, j = x,y \quad (3 - 30 - 2)$$

$$V_\alpha = \sqrt{V_{\alpha x}^2 + V_{\alpha y}^2} = \text{magnitude of } \alpha \text{ phase velocity}$$

$$d_{xl}^i = \alpha_{\alpha l}^i V_\alpha = \text{longitudinal dispersion coefficient of } i \text{ species in } \alpha \text{ phase}$$

$d_{\alpha t}^i = \alpha_{\alpha t}^i V_{\alpha}$ = transverse dispersion coefficient of i species in α phase

$\alpha_{\alpha t}^i$ = longitudinal dispersivity of i species in α phase

$\alpha_{\alpha t}^i$ = transverse dispersivity of i species in α phase

Corapcioglu and Baehr (1987) proposed an alternative formulation. They suggested that the hydrodynamic coefficient tensor depends on the void fraction (θ), the soil type, and phase velocity. They expressed it as follows:

$$D_{\alpha}^i = d_{\alpha} \lambda_{\alpha} + D_{m\alpha}^i \theta_{\alpha} \xi_{\alpha} \quad (3 - 31)$$

Here,

d_{α} = the matrix of factors depending on fluid velocity

λ_{α} = product of θ_{α} and dispersivity

$D_{m\alpha}^i$ = molecular diffusion constant for ith constituent

ξ_{α} = complexity tensor for fluid

Although many forms of equations have been developed for the dispersive constant, no equation is available for the general case. For the diffusive constant of the aquatic domain, good methods have been developed and if we combine the parameter estimation scheme with the experiment, we can acquire reasonable result. However, in the subsurface domain, the tortuosity of the geometry makes the analysis very difficult to perform and the microscopic phase velocity is almost impossible to postulate. The only method used heretofore relies on soil sample tests and the tracer tests. These results can be combined with parameter estimation techniques. Often only approximate re-

sults can be obtained. Additional research is required to overcome these difficulties.

3.2.2 Partitioning Equations

The partitioning equations are based on the current understanding of chemical and biological processes in the subsurface environment. The partitioning concept is a very useful way of describing the transfer of contaminant from one phase to another. The partitioning concept implies instantaneous mass transfer among phases to obtain equilibrium. If the fugacity of each species in each phase is known, the partition ratio can be calculated. The mass fraction of i species in α phase is computed by $w_{\alpha}^i = w_{\beta}^i \times H_{\alpha\beta}^i$ for known mass fraction of i species in β phase w_{β}^i and partition coefficient of i species between α and β phase $H_{\alpha\beta}^i$. The computation time is reduced by this concept.

3.2.2.1 Partition between water and oil phase

From Pinder and Abriola (1986), in the case of trichloroethylene, the solubility of TCE in to water phase is 1100 mg/L. The value of solubility implies mass fraction of TCE in water phase.

$$H_{wo}^o = \frac{w_w^o}{w_o^o} = \frac{\text{TCE mass fraction in water phase}}{\text{TCE mass fraction in oil phase}}$$

If we assume no water species in oil phase ($w_o^o = 1$), then $H_{wo}^o = 1.1 \times 10^{-3}$.

3.2.2.2 Partition between water and air phase

We can use Henry's law to calculate the equilibrium of volatile species with the aqueous phases. Henry's law is defined as,

(partial pressure of chemical in gas phase) / (solubility of chemical in liquid phase).

$$\text{From Pinder and Abriola (1986), } H_{gw}^o = \frac{P_{vp}^o}{S_w^o} = 10^4 \frac{\text{atm}}{\text{mole/cm}^3}$$

$$P_{vp}^o = \text{vapor pressure of TCE in air phase} = 0.0837 \text{ atm}$$

$$S_w^o = \text{solubility of TCE in water phase} = 0.0837 \times 10^{-4} \text{ mole/cm}^3$$

$$\frac{P_{vp}^o}{P_a} = \frac{m_a^o}{m_a} \times \frac{28.9}{131.4}$$

$$28.9, 131.4 = \text{molecular weight of air and TCE}$$

$$\text{Therefore, } w_a^o = 0.0837 \times \frac{131.4}{28.9} = 0.38$$

3.2.2.3 Partition between solid and water phase

In the sorption process, both adsorption and absorption are included. The lab measurement of sorption is implemented by the determination of the sorption isotherm by using the batch method. Figure 4 shows the relation between the sorbed concentration and the equilibrium concentration (Lyman et al., 1982). The partition coefficient of adsorption is $H_{sw}^o = \frac{w_s^o}{w_w^o}$

In general, sorption process depends on the solute, sorbent, solution condition (pH, temperature, ionic strength, specific species, organic and inorganic solute, collide, etc.), and time. Because sorption is dominated by the organic carbon, the partition coefficient is computed by $H_{sw}^o = f_{oc} \times H_{oc}$

$$f_{oc} = \text{fraction of organic carbon, measured from field data}$$

H_{oc} = partition coefficient based on organic carbon, computed by

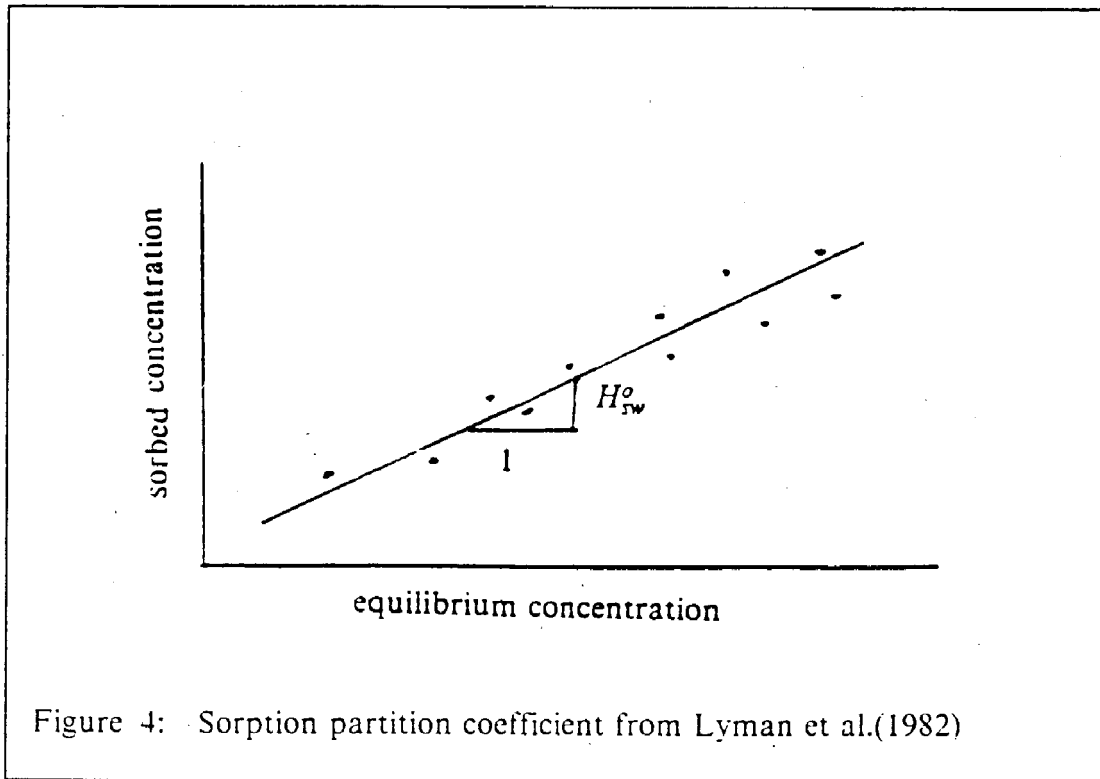
$$\log H_{oc} = a \log(H_{ow}, S, BCF, \dots) + b$$

H_{ow} = octanol water partition coefficient

S = solubility

BCF = bioconcentration factor

Figure 5 is from Karickhoff (1984).



Sorption is greatly influenced by the soil characteristics and cosolvent, requiring the above equation needs to be modified for site specific conditions. When sorptive reactions are occurring slowly, a kinetic model should be used. The simple finite difference approach for the given time step (time step is decided by the main program) is adequate for this evaluation.

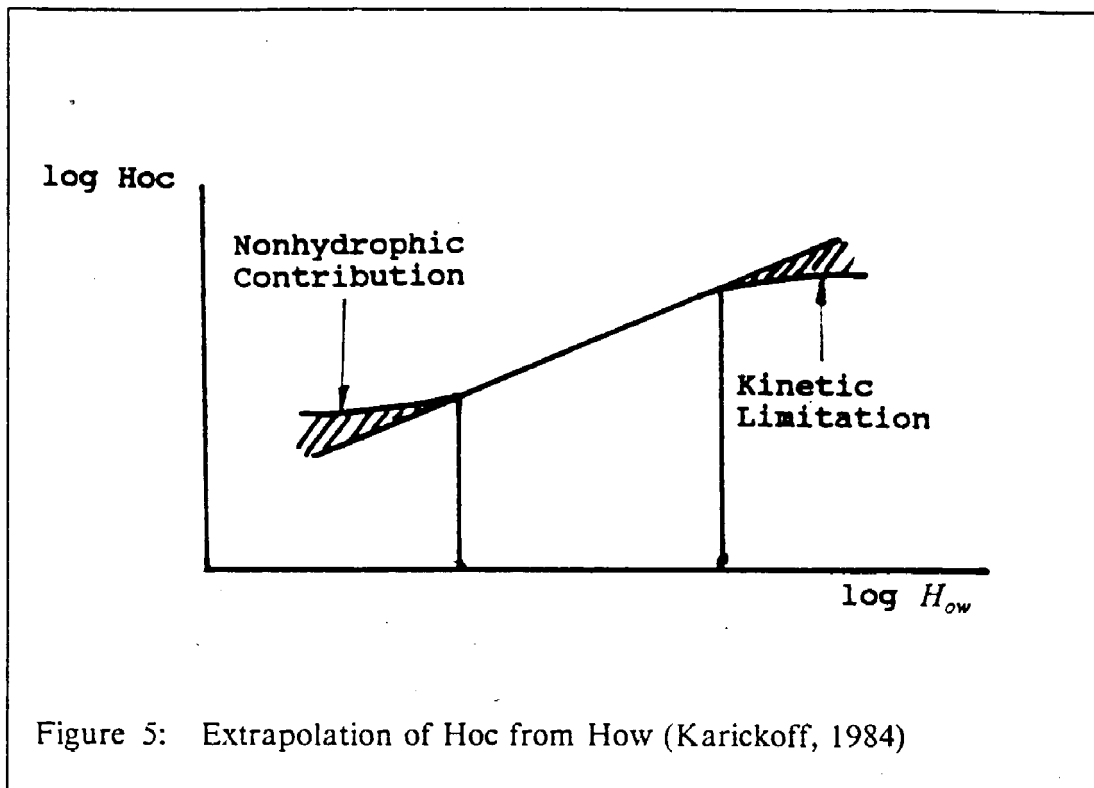


Figure 5: Extrapolation of H_{oc} from H_{ow} (Karickhoff, 1984)

The sorption process is especially important in the case of the solute transport (when the chemical substance is totally dissolved in water phase). Traditionally, solvent "holdup" has been calculated through an empirical "retardation factor", which is calculated from the sorption coefficient. The velocity of the water flow is divided by the value of the retardation factor. In saturated flow, retardation by the sorption process is the cause of the difference between the advancing concentration and the groundwater fronts.

3.3 CHEMICAL AND BIOLOGICAL REACTIONS

The approach for chemical reactions in the aqueous phase is applied to the problem of the subsurface region by assuming that the chemical reactions only occur in the liquid phase. Estimating reaction rates in groundwater is complicated by the effect of buffer catalysis, dissolved metals, dissolved organic materials and cosolvents. The first-order reactions can be expressed by :

$$\frac{dC_x^i}{dt} = -k_\alpha^i C_x^i \quad (3 - 32)$$

Where k_α^i is reaction rate constant of i species in α phase.

To determine the reaction rate, we need the formulation of the reactions first, then the parameter values such as k_α^i . For the evaluation of the parameters, if no published formula is available, we can use the parameter estimation scheme.

Stenstrom and the author did this parameter estimation evaluation of the chemical reactions in the chlorination process in the lab reactor. The governing equation was a very similar form of the solute transport equation. Even though we could get reasonable reaction rate constant values, the physical relation between the reaction rate constant and the surrounding conditions is not easy to postulate because of the experimental error and also ambiguity of the dynamic parameters of the flow.

The biological reaction depends on the following factors :

- a. the quantity of the microorganisms capable of biotransformations

- b. the concentration of the target chemicals
- c. the quantity of the substrate
- d. temperature
- e. the upper and lower limit of the quantity of oxygen
- f. the toxicants

In the subsurface domain, most of the microorganisms are attached to the surface of the solids making the biofilm. So, sloughing and endogenous reactions should be considered. The Monod-type kinetics for the substrate oxidation is commonly used, as follows:

$$\frac{dS_f}{dt} = - \frac{k_m X S_f}{K_s + S_f} \quad (3 - 33)$$

where,

S_f = substrate concentration

X = concentration of active bacteria

k_m = coefficient of maximum rate of utilization

K_s = half velocity coefficient

Chapter IV

DIFFICULTIES IN EACH PROCESS

4.1 PHYSICAL PROCESS

Advection, hydrodynamic dispersion, molecular diffusion and density stratification are physical phenomena which is important for this model. For advection, laboratory tests for porosity and hydraulic conductivity are not reliable for the unconsolidated samples such as sand, gravel or clay. A pumping test for the hydraulic conductivity can be used to obtain average values over the entire domain, so we need the system identification procedure. The dispersion is very dependent on the velocity distribution, and it is difficult to conduct a tracer test properly. For the nonpoint source e.g. rainfall, leakage from other aquifer, it is difficult to estimate and calibration is required.

4.2 CHEMICAL PROCESS

Oxidation, radionuclide decay, ion-exchange, complexation, co-solvation, sorption and immiscible phase partitioning are chemical processes.

The oxidation/reduction or electronic alterations is important in inorganic compounds, but it is difficult to determine the redox state of the aquifer zone and to identify and quantify the redox active reactants. Hydrolysis, elimination and substitution reactions belong to this process. For radionuclide decay, a well defined decay constant for each radionuclide is available.

For the other phenomena, little is known and approximations are usually required. Additional research is required to better determine these phenomena.

4.3 BIOLOGICAL PROCESS

Microbial population dynamics, substrate utilization, biotransformation, adoption, co-metabolism need to be studied in this process.

This process is more important when the contamination residence time is long. Usually the biotransformation rate increases after the exposure to the contaminant. Most microbes are attached to the solid surface (less than 1% of total population is truly planktonic). The parameters from other processes (e.g. contaminant concentration, oxygen, redox state, pH, toxicity of the contaminant) are quite important to this process. Knowledge in this area is still embryonic.

Chapter V

NUMERICAL METHOD

5.1 FINITE ELEMENT METHOD

The finite element method can be interpreted as the minimization of the residual or error between the numerical solution and the true solution over the whole spatial and time domain. Usually a finite difference algorithm is used along the time domain instead of the finite element method, because there is no large difference in accuracy, and the time domain is regular shape as compared to the irregular spatial domain.

To apply a finite element method and to develop a code for the solution of the previously defined different equations, it is necessary to invent several process modules in general fashion. These modules are explained as below:

1. module for inputing data and printing results
2. module related to spatial domain
 - a. module for basis and weighting functions
 - b. module for automatic element generation and nodal connectivities
 - c. module for element-wise evaluation of nodal parameters
3. module for evaluation of element matrices
4. module for assembling the element matrices
5. module for solving the nonlinear system
6. module for empirical equations of parameters and source terms

The remainder of this section discusses the general requirements for solution. Later sections describe the specifics of the solution technique, such as upstream weighting. From the previously defined governing equation,

$$\begin{aligned} & \sum_{\alpha=1}^3 \left[\sum_{\beta=1}^3 (\rho_{\beta} w_{\beta}^i S_{\beta} S_{\beta\alpha}) \frac{\partial h_{\alpha}}{\partial t} + \phi \rho_{\alpha} w_{\alpha}^i \frac{\partial S_{\alpha}}{\partial t} + \phi S_{\alpha} \rho_{\alpha} \frac{\partial w_{\alpha}^i}{\partial t} \right] + \frac{\partial}{\partial t} ((1 - \phi) \rho_s w_s^i) \\ & = \sum_{\alpha=1}^3 \left[\nabla(\rho_{\alpha} w_{\alpha}^i K_{\alpha} (\nabla h_{\alpha} + \frac{\rho_{\alpha} \vec{r}}{\rho_w j})) + \nabla(\phi S_{\alpha} D_{\alpha}^i \nabla(\rho_{\alpha} w_{\alpha}^i)) + g_{\alpha}^i \right] \quad (5-1) \end{aligned}$$

Applying the asymmetric weighting function W_i and integrating over the spatial domain, equation (5-2) is obtained.

$$\begin{aligned} & \sum_{\alpha=1}^3 \left[\int W_i \sum_{\beta=1}^3 (\rho_{\beta} w_{\beta}^i S_{\beta\alpha}) \frac{\partial h_{\alpha}}{\partial t} dR + \int W_i \rho_{\alpha} w_{\alpha}^i \frac{\partial S_{\alpha}}{\partial t} dR + \int W_i \rho_{\alpha} S_{\alpha} \frac{\partial w_{\alpha}^i}{\partial t} dR \right] \\ & + \int W_i \frac{\partial}{\partial t} ((1 - \phi) \rho_s w_s^i) dR = \sum_{\alpha=1}^3 \left[\int W_i \nabla(\rho_{\alpha} w_{\alpha}^i K_{\alpha} (\nabla h_{\alpha} + \frac{\rho_{\alpha} \vec{r}}{\rho_w j})) dR \right. \\ & \left. + \int W_i \nabla(\phi S_{\alpha} D_{\alpha}^i \nabla(\rho_{\alpha} w_{\alpha}^i)) dR + \int W_i g_{\alpha}^i dR \right] \quad (5-2) \end{aligned}$$

Defining the following element matrices as shown below,

$$[ET_{ij}] = \langle W_i, N_j \rangle \quad (5-3-1)$$

$$[EV_{jd,ij}] = \langle W_i, \frac{dN_j}{dx_{jd}} \rangle \quad (5-3-2)$$

$$[EK_{id,i,j}] = \left\langle \frac{dW_i}{dx_{id}}, N_j \right\rangle \quad (5-3-3)$$

$$[ED_{id,jd,i,j}] = \left\langle \frac{dW_i}{dx_{id}}, \frac{dN_j}{dx_{jd}} \right\rangle \quad (5-3-4)$$

where N_j is basis function of node j and W_i is weighting function of node i .

Using the bilinear basis function $N_{i,ig}$, the nodal parameter values are changed to element-wise parameter values as follows :

$$(\rho_\beta w_\beta^i S_{\beta\alpha})_i = \sum_{ig=1}^{nn} (\rho_\beta w_\beta^i S_{\beta\alpha})_{ig} N_{i,ig} \quad (5-4-1)$$

$$(\rho_\alpha w_\alpha^i K_\alpha)_i = \sum (\rho_\alpha w_\alpha^i K_\alpha)_{ig} N_{i,ig} \quad (5-4-2)$$

$$(\phi S_\alpha \rho_\alpha)_i = \sum (\phi S_\alpha \rho_\alpha)_{ig} N_{i,ig} \quad (5-4-3)$$

Using the above element matrices and parameter notation and applying Green's theorem, the finite element scheme of the governing equation is expressed as :

$$\begin{aligned} & \sum_{e=1}^{nel} \sum_{j=1}^{nn} \sum_{\alpha=1}^{na} [[ET_{i,j}] \{ (\sum_{\beta=1}^{na} (\rho_\beta w_\beta^i S_{\beta\alpha})_j) \frac{\partial h_{\alpha j}}{\partial t} + (\phi \rho_\alpha w_\alpha^i)_j \frac{\partial S_{\alpha j}}{\partial t} + (\phi \rho_\alpha S_\alpha)_j \frac{\partial w_{\alpha j}^i}{\partial t} \}} \\ & + \sum_{id=1}^{nd} \sum_{jd=1}^{nd} [ED_{jd,i,j}] \{ (\rho_\alpha w_\alpha^i K_\alpha)_j h_{\alpha j} + (\phi S_\alpha \rho_\alpha)_j w_{\alpha j}^i \}] + [ET_{i,j}] \frac{\partial}{\partial t} ((1-\phi) \rho_s w_s^i)_j \end{aligned}$$

$$\begin{aligned}
&= \sum_{j=1}^{nn} \sum_{x=1}^{na} \left[- \sum_{jd=1}^{nd} ([EK_{jd,ij}](\rho_x w_x^i K_x \frac{\rho_x}{\rho_w})_j) + [ET_{ij}]g_{xj}^i \right. \\
&\quad \left. + \int W_i \rho_x w_x^i K_x (\nabla h_x + \frac{\rho_x}{\rho_w} \vec{j}) \cdot \vec{n} dB + \int W_i \phi S_x D_x^i \nabla(\rho_x w_x^i) \cdot \vec{n} dB \right] \quad (5-5)
\end{aligned}$$

Here,

nel = total number of elements

nd = dimension of the problem

nn = total number of nodes in each element

na = total number of phases excluding soil phase

id = index for direction (1 = x, 2 = y, 3 = z)

The integral at the boundary is evaluated by defining the fluxes as follows :

Let

$$q_{x,h}^i = - \rho_x w_x^i K_x (\nabla h_x + \frac{\rho_x}{\rho_w} \vec{j}) \cdot \vec{n} = \text{outward normal advective flux}$$

$$q_{x,d}^i = - \phi S_x D_x^i \nabla(\rho_x w_x^i) \cdot \vec{n} = \text{outward normal dispersive flux}$$

Using these definitions, the boundary conditions can be written as

$$\int W_i \rho_x w_x^i K_x (\nabla h_x + \frac{\rho_x}{\rho_w} \vec{j}) \cdot \vec{n} dB = - \int W_i q_{x,h}^i dB \quad (5-6-1)$$

$$\int W_i \phi S_x D_x^i \nabla(\rho_x w_x^i) \cdot \vec{n} dB = - \int W_i q_{x,d}^i dB \quad (5-6-2)$$

Combining the generalized finite difference algorithm for time domain, results in the following equation :

$$\begin{aligned}
& \sum_{e=1}^{nel} \sum_{j=1}^{nn} \sum_{\alpha=1}^{na} \left[\frac{1}{\Delta t} [ET_{ij}] \sum_{\beta=1}^{na} \{ (\rho_{\beta} w_{\beta}^i S_{\beta\alpha}) (h_{\alpha j}^{n+1} - h_{\alpha j}^n) \right. \\
& \quad \left. + (\phi \rho_{\alpha} w_{\alpha}^i) (S_{\alpha j}^{n+1} - S_{\alpha j}^n) + (\phi \rho_{\alpha} S_{\alpha}) (w_{\alpha j}^{n+1,i} - w_{\alpha j}^n) \right\} \\
& + \sum_{id=1}^{nd} \sum_{jd=1}^{nd} \{ \varepsilon [ED_{id,jd,ij}]^{n+1} \{ (\rho_{\alpha} w_{\alpha}^i K_{\alpha}^i)^{n+1} h_{\alpha j}^{n+1} + (\phi S_{\alpha} \rho_{\alpha} D_{\alpha}^i)^{n+1} (w_{\alpha j}^i)^{n+1} \} \\
& \quad + (1 - \varepsilon) [ED_{id,jd,ij}]^n \{ (\rho_{\alpha} w_{\alpha}^i K_{\alpha}^i)^n h_{\alpha j}^n + (\phi S_{\alpha} \rho_{\alpha} D_{\alpha}^i)^n (w_{\alpha j}^i)^n \} \} \\
& \quad + \frac{1}{\Delta t} [ET_{ij}] \{ ((1 - \phi) \rho_s w_s^i)^{n+1} - ((1 - \phi) \rho_s w_s^i)^n \} \\
& = \sum_{\alpha=1}^{na} \sum_{j=1}^{nn} \left[- \sum_{jd=1}^{nd} ([EK_{jd,ij}] (\rho_{\alpha} w_{\alpha}^i K_{\alpha}^i \frac{\rho_{\alpha}}{\rho_w})_j) + [ET_{ij}] g_{\alpha j}^i \right. \\
& \quad \left. - \int W_{\alpha,h}^i dB - \int W_{\alpha,d}^i dB \right] \quad (5-7)
\end{aligned}$$

To solve the above equation with respect to the pressure head, saturation and mass fraction terms, let the assembled global matrix and load vector of the i species in α phase be

$$\begin{aligned}
[Ah_{\alpha}^i] &= \sum_e \sum_j \left[\frac{1}{\Delta t} [ET_{ij}] \sum_{\beta} (\rho_{\beta} w_{\beta}^i S_{\beta\alpha})^{n+1} \right. \\
& \quad \left. + \sum_{id} \sum_{jd} \{ \varepsilon [ED_{id,jd,ij}]^{n+1} (\rho_{\alpha} w_{\alpha}^i K_{\alpha}^i)^{n+1} \} \right] \quad (5-8-1)
\end{aligned}$$

$$[As_x^i] = \sum_e \sum_j \frac{1}{\Delta t} [ET_{ij}] (\phi \rho_x w_x^i)_j^{n+1} \quad (5-8-2)$$

$$[Am_x^i] = \sum_e \sum_j \left[\frac{1}{\Delta t} [ET_{ij}] (\phi \rho_x S_x)_j^{n+1} \right. \\ \left. + \sum_{id} \sum_{jd} \{ \varepsilon [ED_{id,jd,ij}]^{n+1} (\phi \rho_x S_x D_x^i)_j^{n+1} \} \right] \quad (5-8-3)$$

$$\{fh_x^i\} = \sum_e \sum_j \left[\left\{ \frac{1}{\Delta t} [ET_{ij}] \sum_{\beta} (\rho_{\beta} w_{\beta}^i S_{\beta x})_j^n \right. \right. \\ \left. \left. + \sum_{id} \sum_{jd} (\varepsilon - 1) [ED_{id,jd,ij}]^n (\rho_x w_x^i K_x)_j^n \right\} (h_{x,j})^n - \sum_j \sum_{jd} ([EK_{jd,ij}] (\rho_x w_x^i K_x \frac{\rho_x}{\rho_w})_j) \right. \\ \left. + [ET_{ij}] g_{x,j}^i - \int W_{f_{x,h}}^i dB - \int W_{f_{x,d}}^i dB \right] \quad (5-8-4)$$

$$\{fs_x^i\} = \sum_e \sum_j \left\{ \frac{1}{\Delta t} [ET_{ij}] (\phi \rho_x w_x^i)_j^n (S_{x,j})^n \right\} \quad (5-8-5)$$

$$\{fm_x^i\} = \sum_e \sum_j \left[\frac{1}{\Delta t} [ET_{ij}] (\phi \rho_x S_x)_j^n \right. \\ \left. + \sum_{id} \sum_{jd} (\varepsilon - 1) [ED_{id,jd,ij}]^n (\phi \rho_x S_x D_x^i)_j^n \right] (w_{x,j}^i)^n \quad (5-8-6)$$

Using the above notation, the equation (5-7) becomes

$$\sum_{x=1}^{na} [Ah_x^i]\{h_x\} + [As_x^i]\{S_x\} + [Am_x^i]\{w_x^i\} = \{fh_x^i\} + \{fs_x^i\} + \{fm_x^i\} \quad (5-9)$$

The above system can be solved in a coupled or decoupled way. To reduce the computation time and to minimize the difficulties associated with the nonlinear terms, the decoupling technique is used. This depends on the property of each primary variable and is case specific. As an example, consider composite multiphase contaminant migration. Algorithms for specific problems are explained later. By using second approach of the governing equation, with the immobile air phase assumption, the governing equations are reduced to the following set:

Equation for water phase pressure head

$$\begin{aligned} [Ah_w^w]\{h_w\}^{n+1/2} = & -([Ah_o^w]\{h_o\}^n + [Ah_a^w]\{h_a\}^n) \\ & + \sum_x \{ -[As_x^i]\{S_x\}^n - [Am_x^i]\{w_x^i\}^n + \{fh_x^i\}^n + \{fs_x^i\}^n + \{fm_x^i\}^n \} \quad (5-10-1) \end{aligned}$$

Equation for oil phase pressure head

$$\begin{aligned} [Ah_o^o]\{h_o\}^{n+1} = & -([Ah_w^o]\{h_w\}^{n+1/2} + [Ah_a^o]\{h_a\}^{n+1/2}) \\ & + \sum_x \{ -[As_x^i]\{S_x\}^n - [Am_x^i]\{w_x^i\}^n + \{fh_x^i\}^n + \{fs_x^i\}^n + \{fm_x^i\}^n \} \quad (5-10-2) \end{aligned}$$

Equation for dissolved species in the water phase

$$\sum_x [Am_x^o]\{w_x^o\}^{n+1} = \sum_x \{ -[Ah_x^i]\{h_x\}^n - [As_x^i]\{S_x\}^n$$

$$+ \{fh_x^i\}^n + \{fs_x^i\}^n + \{fm_x^i\}^n \quad (5 - 10 - 3)$$

The assembling procedure is performed carefully to make the resulting global matrix banded form so that an asymmetric banded matrix solver can be used and storage requirement for the variables can be reduced. Iteration continues until the error criteria (difference of h_α^{n+1} , h_α^n and of $w_\alpha^{i,n+1}$, $w_\alpha^{i,n}$) is satisfied. After each iteration for nonlinearity and decoupling, the computation proceeds to next time level.

5.1.1 Basis Function

Because the basis function only depends on the geometry of each element, the evaluation of the basis function needs to be performed only once which greatly reduces computation time. Therefore, all the basis functions are evaluated at all Gaussian points and later assembled when the integration of element matrices are performed. By mapping, the integration is performed in the local domain. Figure 6 shows transformation of global coordinates to local ones in one, two, three dimensional elements.

The basis functions and asymmetric weighting functions can be expressed by the notation of the linear basis function in each direction. Figure 7 shows the basis and weighting functions.

The basis functions are :

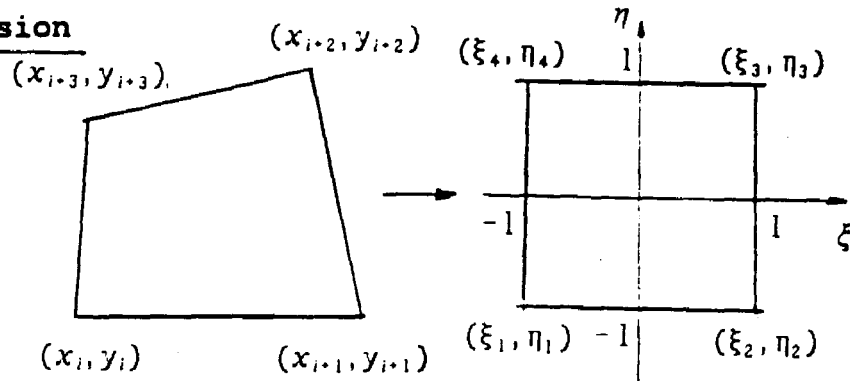
$$N_{\xi i, ig} = 0.5(1 + \xi \xi_{ig}), \quad N_{\eta i, ig} = 0.5(1 + \eta \eta_{ig})$$

$$N_{\zeta i, ig} = 0.5(1 + \zeta \zeta_{ig}) \quad (5 - 11 - 1)$$

One Dimension



Two Dimension



Three Dimension

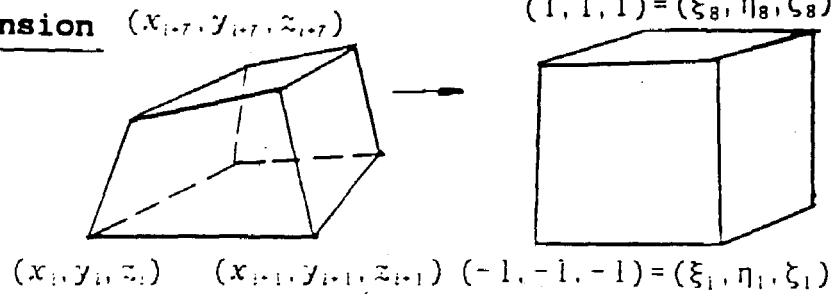


Figure 6: Transformation of global coordinates into local ones

$$N_{i,ig} = N_{\xi_i,ig} N_{\eta_i,ig} N_{\zeta_i,ig} \quad (5-11-2)$$

The derivatives of basis functions are :

$$\frac{\partial N_{\xi_i}}{\partial \xi} = 0.5 \xi_i, \quad \frac{\partial N_{\eta_i}}{\partial \eta} = 0.5 \eta_i, \quad \frac{\partial N_{\zeta_i}}{\partial \zeta} = 0.5 \zeta_i \quad (5-12-1)$$

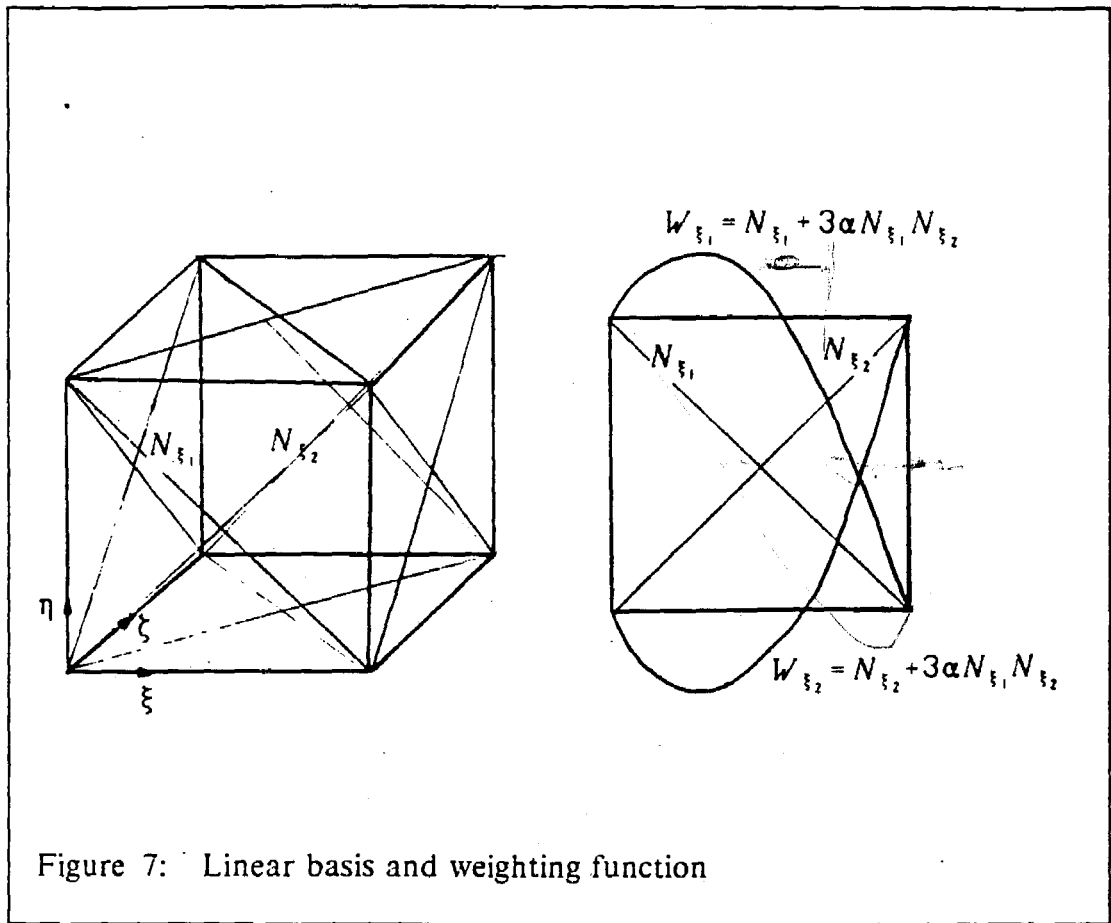


Figure 7: Linear basis and weighting function

$$\frac{\partial N_{i,ig}}{\partial \xi} = \frac{\partial N_{\xi i}}{\partial \xi} N_{\eta i,ig} N_{\zeta i,ig} \quad \frac{\partial N_{i,ig}}{\partial \eta} = \frac{\partial N_{\eta i}}{\partial \eta} N_{\xi i,ig} N_{\zeta i,ig}$$

*

$$\frac{\partial N_{i,ig}}{\partial \zeta} = \frac{\partial N_{\zeta i}}{\partial \zeta} N_{\xi i,ig} N_{\eta i,ig} \quad (5-12-2)$$

The local coordinates of nodal points in each element are :

$$\xi_i = -1, 1, 1, -1, -1, 1, 1, -1$$

$$\eta_i = -1, -1, 1, 1, -1, -1, 1, 1$$

$$\zeta_i = 1, 1, 1, 1, -1, -1, -1, -1$$

The Gaussian points are :

$$\xi_{ig} = -0.577, 0.577, 0.577, -0.577, -0.577, 0.577, 0.577, -0.577$$

$$\eta_{ig} = -0.577, -0.577, 0.577, 0.577, -0.577, -0.577, 0.577, 0.577$$

$$\zeta_{ig} = 0.577, 0.577, 0.577, 0.577, -0.577, -0.577, -0.577, -0.577$$

The derivatives of global basis functions are derived as follows:

$$\begin{bmatrix} \frac{\partial N_{i,ig}}{\partial \xi} \\ \frac{\partial N_{i,ig}}{\partial \eta} \\ \frac{\partial N_{i,ig}}{\partial \zeta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x_{ig}}{\partial \xi} & \frac{\partial y_{ig}}{\partial \xi} & \frac{\partial z_{ig}}{\partial \xi} \\ \frac{\partial x_{ig}}{\partial \eta} & \frac{\partial y_{ig}}{\partial \eta} & \frac{\partial z_{ig}}{\partial \eta} \\ \frac{\partial x_{ig}}{\partial \zeta} & \frac{\partial y_{ig}}{\partial \zeta} & \frac{\partial z_{ig}}{\partial \zeta} \end{bmatrix} \begin{bmatrix} \frac{\partial N_{i,ig}}{\partial x} \\ \frac{\partial N_{i,ig}}{\partial y} \\ \frac{\partial N_{i,ig}}{\partial z} \end{bmatrix} \quad (5-13)$$

The above matrix is called as Jacobian matrix $[J]_{ig}$.

$$x_i = \sum_{i=1}^{nn} N_{i,ig} x_i \quad \frac{\partial x_i}{\partial \xi} = \sum_{i=1}^{nn} \frac{\partial N_{i,ig}}{\partial \xi} x_i \quad (5-14)$$

$$[J]_{ig} = \begin{bmatrix} \sum \frac{\partial N_{i,ig}}{\partial \xi} x_i & \sum \frac{\partial N_{i,ig}}{\partial \xi} y_i & \sum \frac{\partial N_{i,ig}}{\partial \xi} z_i \\ \sum \frac{\partial N_{i,ig}}{\partial \eta} x_i & \sum \frac{\partial N_{i,ig}}{\partial \eta} y_i & \sum \frac{\partial N_{i,ig}}{\partial \eta} z_i \\ \sum \frac{\partial N_{i,ig}}{\partial \zeta} x_i & \sum \frac{\partial N_{i,ig}}{\partial \zeta} y_i & \sum \frac{\partial N_{i,ig}}{\partial \zeta} z_i \end{bmatrix} \quad (5-15)$$

$$dx dy = \det([J])_{ig} d\xi d\eta \quad (5-16)$$

$$\begin{bmatrix} \frac{\partial N_{i,ig}}{\partial x} \\ \frac{\partial N_{i,ig}}{\partial y} \\ \frac{\partial N_{i,ig}}{\partial z} \end{bmatrix} = [J]_{ig}^{-1} \begin{bmatrix} \frac{\partial N_{i,ig}}{\partial \xi} \\ \frac{\partial N_{i,ig}}{\partial \eta} \\ \frac{\partial N_{i,ig}}{\partial \zeta} \end{bmatrix} \quad (5-17)$$

The components of the element matrices are:

$$\begin{aligned} [ET_{ij}] &= \iiint W_i N_j dx dy dz = \iiint W_i N_j |J| d\xi d\eta d\zeta \\ &= \sum_{ig=1}^{nn} W_{i,ig} N_{j,ig} |J| \quad (5-18-1) \end{aligned}$$

$$\begin{aligned} [EV_{jd,i,j}] &= \iiint W_i \frac{dN_j}{dx_{jd}} dx dy dz = \iiint W_i \frac{dN_j}{dx_{jd}} |J| d\xi d\eta d\zeta \\ &= \sum_{ig=1}^{nn} W_{i,ig} \frac{dN_{j,ig}}{dx_{jd}} |J| \quad (5-18-2) \end{aligned}$$

$$[ED_{id,jd,i,j}] = \iiint \frac{dW_i}{dx_{id}} \frac{dN_j}{dx_{jd}} dx dy dz = \sum_{ig=1}^{nn} \frac{dW_{i,ig}}{dx_{id}} \frac{dN_{j,ig}}{dx_{jd}} |J| \quad (5-18-3)$$

5.1.2 Upstream Weighting Function

As indicated previously, upstream weighting is used to reduce numerical oscillation. Using the technique of Huyakorn and Nikuha (1979), asymmetric weighting functions are developed from the basis function by adding a asymmetric weighting term. In Figure 7, the additional weighting terms were shown.

The weighting functions are :

$$W_{\xi i,ig} = N_{\xi i,ig} + 3\alpha_{\xi i} \xi_i N_{\xi 1,ig} N_{\xi 2,ig}, \quad W_{\eta i,ig} = N_{\eta i,ig} + 3\beta_{\eta i} \eta_i N_{\eta 1,ig} N_{\eta 2,ig}$$

$$W_{\zeta i,ig} = N_{\zeta i,ig} + 3\gamma_{\zeta i} \zeta_i N_{\zeta 1,ig} N_{\zeta 2,ig} \quad (5-19-1)$$

$$W_{i,ig} = W_{\xi i,ig} W_{\eta i,ig} W_{\zeta i,ig} \quad (5-19-2)$$

The derivatives of weighting functions are :

$$\frac{\partial W_{\xi i,ig}}{\partial \xi} = \frac{\partial N_{\xi i}}{\partial \xi} + 3\alpha_{\xi i} \xi_i 0.25(0 - 2\xi_{ig}), \quad \frac{\partial W_{\eta i,ig}}{\partial \eta} = \frac{\partial N_{\eta i}}{\partial \eta} + 3\beta_{\eta i} \eta_i 0.25(0 - 2\eta_{ig})$$

$$\frac{\partial W_{\zeta i,ig}}{\partial \zeta} = \frac{\partial N_{\zeta i}}{\partial \zeta} + 3\gamma_{\zeta i} \zeta_i 0.25(0 - 2\zeta_{ig}) \quad (5-20-1)$$

$$\frac{\partial W_{i,ig}}{\partial \xi} = \frac{\partial W_{\xi i}}{\partial \xi} W_{\eta i,ig} W_{\zeta i,ig}, \quad \frac{\partial W_{i,ig}}{\partial \eta} = \frac{\partial W_{\eta i,ig}}{\partial \eta} W_{\xi i,ig} W_{\zeta i,ig}$$

$$\frac{\partial W_{i,ig}}{\partial \zeta} = \frac{\partial W_{\zeta i,ig}}{\partial \zeta} W_{\xi i,ig} W_{\eta i,ig} \quad (5-20-2)$$

By using the Jacobian matrix [J]

$$\left(\frac{\partial W_{i,ig}}{\partial x}, \frac{\partial W_{i,ig}}{\partial y}, \frac{\partial W_{i,ig}}{\partial z} \right)^T = [J]^{-1} \left(\frac{\partial W_{i,ig}}{\partial \xi}, \frac{\partial W_{i,ig}}{\partial \eta}, \frac{\partial W_{i,ig}}{\partial \zeta} \right)^T \quad (5-21)$$

The weighting factors are evaluated from the nodal velocity as follows:

$$\alpha_1 = \alpha_2 = \frac{V_{\xi 1} + V_{\xi 2}}{|V_{\xi 1}| + |V_{\xi 2}|}, \quad \alpha_3 = \alpha_4 = \frac{V_{\xi 3} + V_{\xi 4}}{|V_{\xi 3}| + |V_{\xi 4}|} \quad (5-22-1)$$

$$\beta_1 = \beta_4 = \frac{V_{\eta 1} + V_{\eta 2}}{|V_{\eta 1}| + |V_{\eta 2}|}, \quad \beta_2 = \beta_3 = \frac{V_{\eta 3} + V_{\eta 4}}{|V_{\eta 3}| + |V_{\eta 4}|} \quad (5-22-2)$$

$$\gamma_1 = \gamma_5 = \frac{V_{\zeta 1} + V_{\zeta 5}}{|V_{\zeta 1}| + |V_{\zeta 5}|}, \quad \gamma_2 = \gamma_6 = \frac{V_{\zeta 2} + V_{\zeta 6}}{|V_{\zeta 2}| + |V_{\zeta 6}|} \quad (5-22-3)$$

Where, $\alpha_i, \beta_i, \gamma_i$ = weighting factor along ξ, η, ζ , direction.

5.1.3 Boundary Conditions

In the finite element analogue of the governing equation, as shown previously in equation (5-7), the term related to the boundary conditions is :

$$- \int W^i q_{x,h}^i dB - \int W^i q_{x,d}^i dB \quad (5-23)$$

The above equation quantifies the overall influx at the boundaries. Even though the finite element analogue is applied to each element over the whole domain, the values of boundary fluxes at inside elements cancel each other, so only evaluations at the overall outside boundaries are needed. There can be three types of boundary conditions, described as follows:

1. Specified primary variable values

2. Specified flux values

3. Mixed type

The integrations of the boundary fluxes are also dependent on the dimension of the problem; therefore, the algorithm must be derived for multidimensional cases for the above three type boundary conditions.

5.1.3.1 1st type boundary (known primary variables)

By adding the large penalty values to the right and left hand side of the final system of equations, the known boundary values are maintained through the simulation.

$$h = h_B$$

At this boundary point, if we denote this boundary node as ib , then

$$h_{ib}(A_{ib} + penalty) + \dots = f_{ib} + h_B \times penalty \quad (5 - 24)$$

From the above equation, if the value of penalty is very large, then the result is $h_{ib} = h_B$.

5.1.3.2 2nd type boundary (known flux at node)

This case exists when fluxes are known. By using the basis and weighting functions, the integration of normal fluxes can be performed as follows :

(a) One Dimensional Problem

$$f_{b,1} = -q_{b,1}, f_{b,np} = -q_{b,np} \quad (5 - 25)$$

The functions $f_{b,1}$ and $f_{b,np}$ are added to the right hand side global vector.

(b) Two Dimensional Problem

$$\int W_i q_n dB = \sum_{j=1}^2 \langle W_i, N_j \rangle q_{bj} \quad (5-26-1)$$

If basis function N_i is used for weighting function W_i , then

$$\begin{aligned} \begin{bmatrix} f_{b,i} \\ f_{b,i+1} \end{bmatrix} &= - \int \begin{bmatrix} N_1 N_1 & N_1 N_2 \\ N_2 N_1 & N_2 N_2 \end{bmatrix} dB \begin{bmatrix} q_{b,i} \\ q_{b,i+1} \end{bmatrix} \\ &= \frac{\Delta B}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} q_{b,i} \\ q_{b,i+1} \end{bmatrix} \end{aligned} \quad (5-26-2)$$

where $q_{b,i}$ and $q_{b,i+1}$ is known normal flux for node i and $i+1$.

The functions $f_{b,i}$ and $f_{b,i+1}$ are added to the right hand side assembled global vector.

(c) Three Dimensional Problem

$$f_{b,i} = - \iiint W_i q_n dB = \sum_{j=1}^4 \langle W_i, N_j \rangle q_{bj} \quad (5-27)$$

The function $f_{b,i}$ is added to the right hand side assembled global vector.

5.1.3.3 3rd type boundary (mixed type)

This is the case when the outflux of the concentration or phase is proportional to the difference between the boundary concentration or pressure head and the concentration of the surrounded area. That is :

$$q_n = k_b(C - C_b) \quad (5 - 28 - 1)$$

where k_b is a proportionality coefficient.

This expression is incorporated into a generalized finite difference scheme as follows:

$$q_n = k_b(\varepsilon C^{n+1} + (1 - \varepsilon)C^n - C_b) \quad (5 - 28 - 2)$$

This results in the following equations:

$$\begin{aligned} f_{b,i} &= - \int W_i q_n dB = \int W_i k_b (\varepsilon C^{n+1} + (1 - \varepsilon)C^n - C_b) dB \\ &= - \int W_i k_b \varepsilon C^{n+1} dB - \int W_i k_b (1 - \varepsilon)(C^n - C_b) dB \\ &= \sum_{ig=1}^{nb} \{ - \langle W_i, N_j \rangle (k_b \varepsilon C^{n+1})_j - \langle W_i, N_j \rangle (k_b (1 - \varepsilon)(C^n - C_b))_j \} \quad (5 - 29) \end{aligned}$$

The matrix $\sum_{ig=1}^{nb} \langle W_i, N_j \rangle (k_b \varepsilon)_j$ is added to left assembled global matrix and the vector $-\sum_{ig=1}^{nb} \langle W_i, N_j \rangle (k_b (1 - \varepsilon)(C^n - C_b))_j$ is added to right side load vector.

5.2 NONLINEARITY TREATMENT

As shown in the section 3.2, the system of equations is very dependent on the component equations. To avoid a system of nonlinear algebraic equations, the nonlinear terms at the new (unknown) time level must be estimated. It is common practice to use the values of the nonlinear terms from the old time level. To insure minimum error, some sort of iteration and error criteria are required. If the approximate value of the unknowns cannot be estimated, or if the nonlinearities are severe, a large number of iterations with excessive computation time will result. Very careful selection of the step size for space and time, and proper initial condition are required to reduce the instability problems related to nonlinearity.

In the case of the dispersive transport, the stability problem is not so severe as compared to the advective transport case, when the governing equations are approaching hyperbolic type. For the latter case, it will be necessary to use a very small time step in addition to the upstream weighting scheme. This becomes a big problem in the highly nonlinear case. To overcome this problem, it was necessary to use the variable time and spatial step size. The variable time step can be determined from the truncation error. The variable spatial step size requires the algorithm to find the sharp advancing front. If we compute the gradient of the concentration, then an abrupt change of the gradient indicates the presence of a sharp advancing front. Near this location, smaller step size is required.

To overcome the difficulty of the nonlinearity of the system, mass lumping, upstream weighting and iteration techniques are used.

5.2.1 Mass Lumping

In the evaluation of the time derivative of the mass matrix, diagonalization is performed to reduce the instability of the solution. Therefore,

$$\langle N_i, N_j \rangle = 0 \text{ for } i \neq j \quad (5-30)$$

5.2.2 Iteration Scheme

To minimize the iteration and to use the variable time step, the method proposed by Cooley (1983) is used as follows:

$$dh = h^{k+1} - h^k \quad dh_{\max} = \max |dh|_i \quad h^{k+1} = w \times dh + h^k$$

where,

dh = difference of the pressure head between new and old iteration level

k = index for iteration level

dh_{\max} = maximum value of dh

$$S = \frac{dh_{\max}^{k+1}}{w^k dh_{\max}^k} \text{ for } k > 0, \quad = 1 \text{ for } k = 0$$

$$w^e = \frac{3 + S}{3 + |S|} \text{ for } S \geq -1, \quad = \frac{1}{2|S|} \text{ for } S < -1$$

$$w^{k+1} = w^e \text{ for } w^e \leq \frac{dha}{|dh_{\max}|}, \quad = \frac{dha}{dh_{\max}} \text{ for } w^e > \frac{dha}{dh_{\max}}$$

where dha = maximum allowable dh in each iteration.

The convergence criteria is expressed by :

$$|dh| \leq \varepsilon_a + \varepsilon_r |h^r| \quad (5-31)$$

ε_a = absolute convergence error ε_r = relative convergence error

The time step is controlled by :

$$dt = dt \times (1 + f_t) \quad \text{for } i \leq I_{\min} \quad (5-32-1)$$

$$dt = \frac{dt}{1 + f_t} \quad \text{for } i \geq I_{\max} \quad (5-32-2)$$

where, f is the time changing factor.

5.2.3 *Element-wise Parameter Evaluation*

Because the element matrix is computed by the integration of the basis functions at Gaussian points over the elementary domain, the parameters in each element matrix, such as relative permeability, fluid capacity, density etc., should be evaluated over the elementary domain using the Gaussian points, as shown in equations (5-4). Generally, this evaluation scheme is expressed as follows :

$$K_i = \sum_{g=1}^{nn} K_{ig} N_{i,ig} \quad (5-33)$$

where,

K_i = element-wise parameter at node i

K_{ig} = parameter at node ig

$N_{i,jg}$ = basis function

5.3 STABILITY AND ERROR ANALYSIS OF THE ALGORITHM

In digital simulation, the continuous time and spatial domain should be discretized and the variables evaluated at the nodal points along the time and the spatial domain. All the algorithms of the governing equations are related to the nodal points which are adjacent in spatial and time domain. Values of the dependent variables not denoted at nodal points must be determined by interpolation.

There is no difference in the above sense whether the algorithm is finite element or finite difference. Even though in the finite element method the basis function is evaluated at each element and assembled later, the resulting global matrix indicates that the basis function can be interpreted as a scheme choosing different nodal points comparable to the finite difference method. If we can construct the nodal points relations in a fashion similar to the FDM, we can apply the above method for stability and error analysis. Pinder and Gray (1977) provided a detailed explanation of the stability and errors in the finite element method of the simple solute transport equation by using the Fourier Series representation for the numerical and analytical solution. In this study, the methodology developed by Karplus (1958) is used to insure a stable solution because of its simplicity and clearness.

In FEM, even though the basis function is evaluated at each element, because of the geometrical connectivity of the each nodal point, the resulting global matrix is banded for the one, two and three dimensional cases. This results because the governing equation is only concerned about the points of the adjacent nodes.

To provide a simple explanation, a one dimensional case is used. Two and three dimensional cases and different basis functions, such as quadratic or hermitian polynomials can be implemented by expanding the following analysis. For illustrative purposes, the simple solute transport is used as follows:

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x} (D \frac{\partial C}{\partial x}) - V \frac{\partial C}{\partial x} \quad (5-34)$$

If a linear basis function is used, then the resulting finite element scheme is

$$\sum_{e=1}^{nel} \frac{\Delta x}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} \frac{\partial C_1}{\partial t} \\ \frac{\partial C_2}{\partial t} \end{bmatrix} + \left(\frac{V}{2} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} + \frac{D}{\Delta x} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \right) \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = 0 \quad (5-35)$$

For the one dimensional case, to see the relation between a nodal point and its adjacent points, assembling the two element matrices as follows:

$$\frac{\Delta x}{6} \begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} \frac{\partial C_1}{\partial t} \\ \frac{\partial C_2}{\partial t} \\ \frac{\partial C_3}{\partial t} \end{bmatrix} + \left(\frac{V}{2} \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix} + \frac{D}{\Delta x} \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \right) \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = 0 \quad (5-36)$$

From the above finite element analogue,

$$\begin{aligned} \frac{1}{6} \left[\frac{\partial C_{i-1}}{\partial t} + 4 \frac{\partial C_i}{\partial t} + \frac{\partial C_{i+1}}{\partial t} \right] + \frac{V}{2\Delta x} [C_{i+1} - C_{i-1}] \\ = \frac{D}{\Delta x^2} [C_{i+1} - 2C_i + C_{i-1}] \quad (5-37) \end{aligned}$$

The time derivatives are evaluated as follows, using the θ method :

$$\begin{aligned} \frac{1}{6\Delta t} [(C_{i+1}^{n+1} - C_{i+1}^n) + 4(C_i^{n+1} - C_i^n) + (C_{i-1}^{n+1} - C_{i-1}^n)] + \\ \frac{V}{2\Delta x} [\theta(C_{i+1} - C_{i-1})^{n+1} + (1-\theta)(C_{i+1} - C_{i-1})^n] = \\ \frac{D}{\Delta x^2} [\theta(C_{i+1} - 2C_i + C_{i-1})^{n+1} + (1-\theta)(C_{i+1} - 2C_i + C_{i-1})^n] \quad (5-38) \end{aligned}$$

After rearranging each term of the above equation in the form of $C_i^{n+1} - C_i^n$, the summation of all the coefficients should be less than 0 to satisfy the stability criteria. Therefore,

$$\frac{D}{\Delta x^2} 2(1-\theta) - \frac{4}{6\Delta t} < 0 \quad \frac{D\Delta t}{\Delta x^2} < \frac{1}{3(1-\theta)} \quad \frac{C_r}{P_e} < \frac{1}{3(1-\theta)} \quad (5-39)$$

where,

$$C_r = V \frac{\Delta t}{\Delta x} = \text{Courant Number}$$

$$P_e = V \frac{\Delta x}{D} = \text{Local Peclet Number}$$

In using the above stability criteria, we should consider the Courant condition also, because the system can be interpreted as the moving coordinate by the

velocity component of the total derivative. The velocity should be within the already defined analogue. Therefore, $V\Delta t < \Delta x$, which is same as $C_r < 1$.

The above FEM algorithm is very similar to the finite difference algorithm except for the time derivative which is evaluated at three different spatial points with the weighting factors 1, 4, 1.

Applying the above technique to generalized finite difference analogue, the stability condition of generalized FDM is :

$$\frac{D\Delta t}{\Delta x^2} < \frac{1}{2(1-\theta)} \quad (5-40)$$

The analogue of forward FEM and FDM is shown in Figure 8.

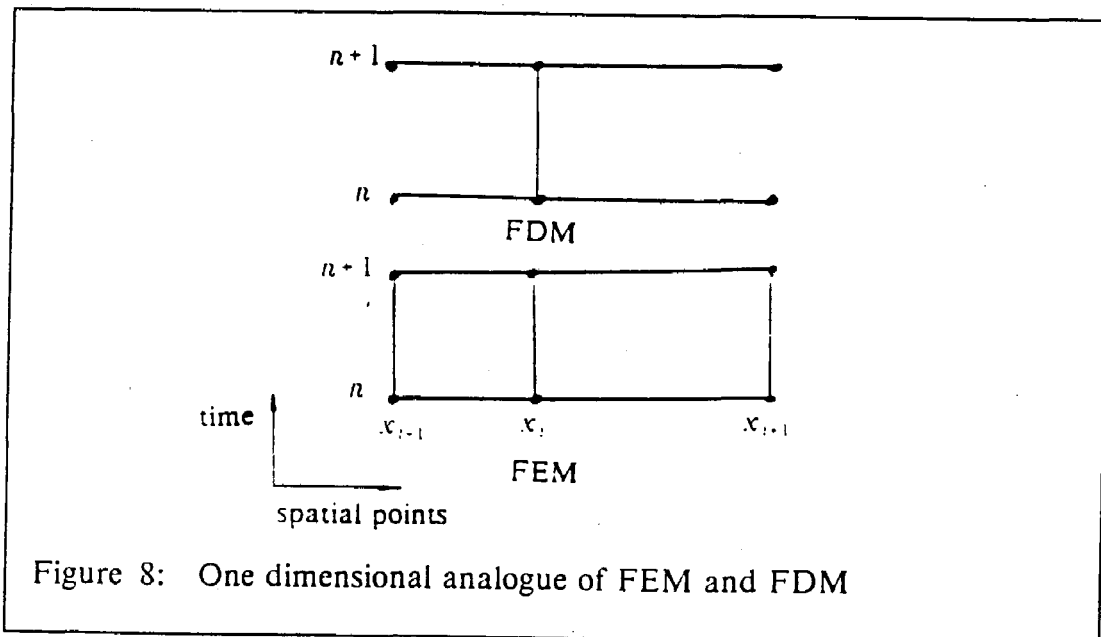


Figure 8: One dimensional analogue of FEM and FDM

The above approach for the stability analysis is not only simple but also provides physical insight of the algorithm by constructing the finite difference-like analogue.

Chapter VI

SIMULATION OF SPECIFIC PROBLEMS

6.1 UNSATURATED GROUNDWATER FLOW SYSTEM

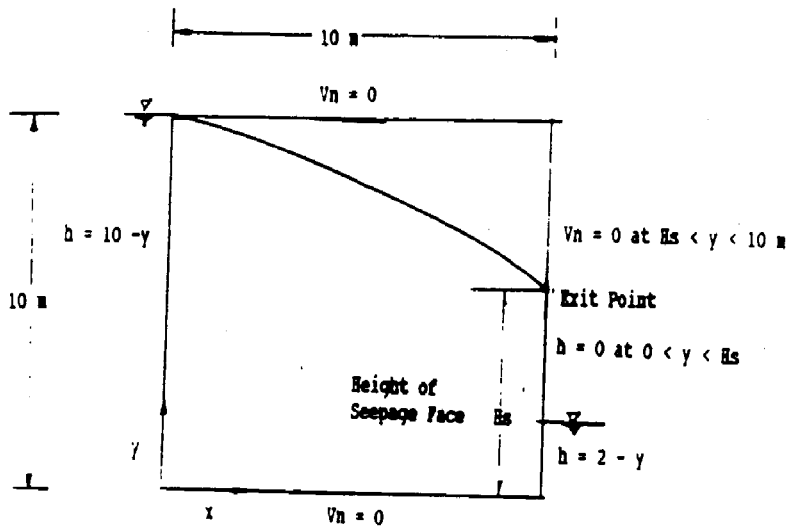
To test the highly nonlinear property of the governing equation, the steady state drainage through an embankment problem by Cooley (1983) was used. If capillary pressure is the dominant force of the system, then from the second governing equation,

$$(S_w S_{ww} + \phi \frac{\partial S_w}{\partial h_w}) \frac{\partial h_w}{\partial t} = \nabla \cdot (K_w (\nabla h_w + \vec{j})) + \frac{g_w}{\rho_w} \quad (6-1-1)$$

$$S_w = f(h_w, h_a) \quad (6-1-2)$$

The problem domain and data are shown in Figure 9.

To find the height of seepage face, the method suggested by Cooley (1983) was used. As shown in Figure 10, the result matches well with the comparison by Huyakorn et al. (1986).



Physical Parameters

Saturated hydraulic conductivity $K_w = 0.01$ m/day

Air pressure head $h_a = 0.0$ m

Residual water saturation $S_{wr} = 0.2$

Effective saturation $S_e = \frac{1}{1 + (\alpha_{sw}/h_{sw})^n}$ for $h_{sw} > 0$
 $\alpha_{sw} = 0.5623$ $n = 4$

Relative permeability $k_{rw} = S_e^m$ $m = 4$

Numerical Data

Spatial step size $\Delta x = 1$ m, $\Delta y = 1$ m

Error Criteria error = 0.01 m

Figure 9: Problem definition and data from Huyakorn et al.(1986)

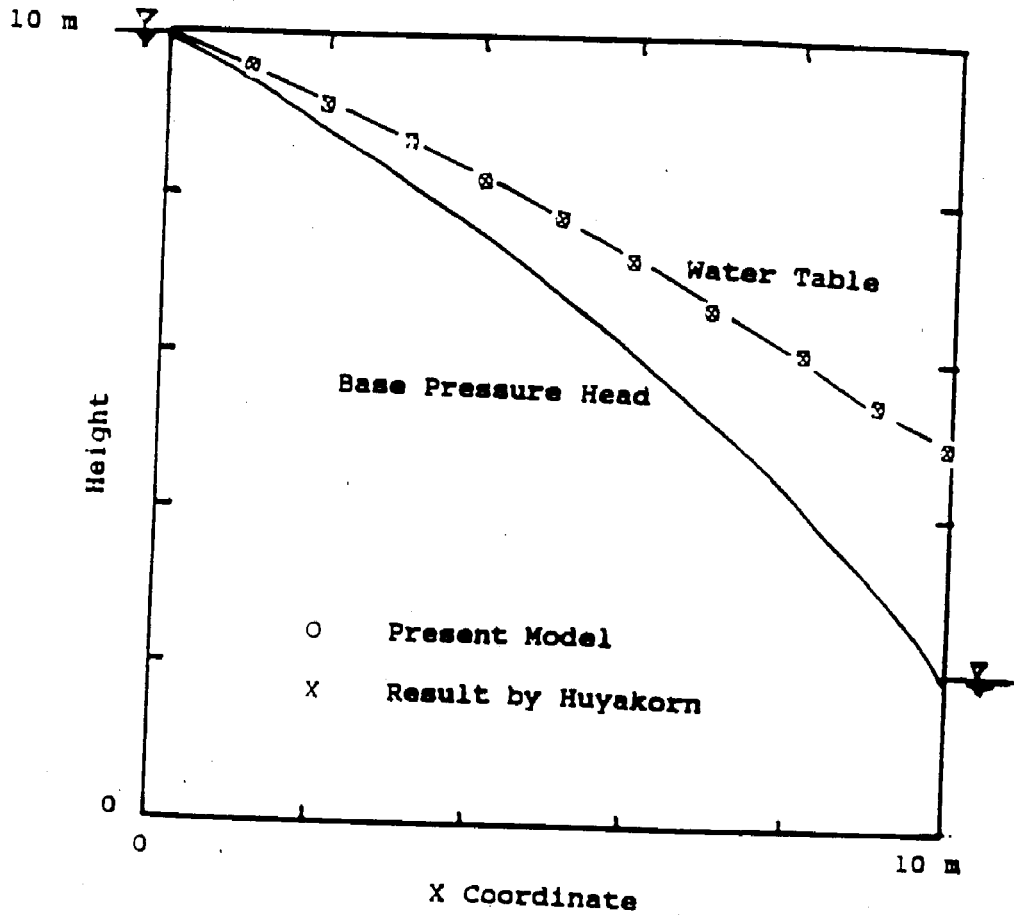


Figure 10: Water table and base pressure head from Huyakorn et al.(1986)

6.2 MASS TRANSFER SYSTEM

6.2.1 One Dimensional Upstream Weighted FEM

By using the partitioning concepts in section 3-2-2,

$$w_w^i = H_{wo}^i w_o^i, \quad w_a^i = H_{ao}^i w_o^i, \quad w_s^i = H_{so}^i w_o^i \quad (6-2)$$

Replacing the mass fraction of other phases by the partitioned values, the governing equation becomes :

$$\frac{\partial}{\partial t}(\kappa_1 w_o^i) = -\nabla(\bar{\kappa}_2 w_o^i) + \nabla(\underline{\kappa}_3 \nabla(w_o^i)) + G^i \quad (6-3)$$

where,

$$\kappa_1 = \theta_w \rho_w H_{wo}^i + \theta_o \rho_o + \theta_a \rho_a H_{ao}^i + \rho_s \rho_s H_{so}^i$$

$$\bar{\kappa}_2 = \theta_w \rho_w H_{wo}^i \bar{V}_w + \theta_o \rho_o \bar{V}_o + \theta_a \rho_a H_{ao}^i \bar{V}_a$$

$$\underline{\kappa}_3 = \theta_w \rho_w H_{wo}^i \underline{D}_w^i + \theta_o \rho_o \underline{D}_o^i + \theta_a \rho_a H_{ao}^i \underline{D}_a^i \quad G^i = \sum_{\alpha} g_{\alpha}^i$$

Even though the derivation of the governing equation is from the composite multiphase approach, the resulting equation is very similar to the solute transport equation for the one chemical case, which is :

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x} \left(D \frac{\partial C}{\partial x} \right) - V \frac{\partial C}{\partial x} + k \quad (6-4)$$

Where k indicates the reaction rate.

Even though there is a slight difference between the mass transfer equation and the solute transport equation, the solute transport equation can be used

as the basic structure of the governing equation. Excluding the source term, the solute transport equation becomes same form as equation (5-34). Differential operator $L(C)$ is defined as follow:

$$L(C) = \frac{\partial C}{\partial t} - \frac{\partial}{\partial x} \left(D \frac{\partial C}{\partial x} \right) + V \frac{\partial C}{\partial x} \quad (6-5)$$

The independent variable, in this case concentration value C , is the summation of the nodal concentration value multiplied by the basis function of each node. That is $C(t,x) = \sum_{j=1}^{nn} N_j(x) C_j(t)$. Because the finite difference scheme will be used for the time domain, the basis function N_j is only the function of x . If one wants to also use the finite element method for the time domain, the basis function should be a function of time and space (e.g. $N_j(x,t)$).

To minimize the weighted residual over the whole spatial domain, let the integral of the weighted residual be zero. $\int R(x) \times W_i(x) = 0$ for $i=1 \dots np$, where residual $R(x) = L(C) - 0$.

The above equation is expressed as $\langle R(x), W_i(x) \rangle = 0$

Integrating the above equation using Galerkin method and the rule of the integration by parts, the finite element analogue of solute transport equation is expressed as follow:

$$\sum \left[\left\langle W_i, \frac{\partial C_j}{\partial t} N_j \right\rangle + D \left\langle \frac{\partial W_i}{\partial x}, C_j \frac{\partial N_j}{\partial x} \right\rangle + V \left\langle W_i, C_j \frac{\partial N_j}{\partial x} \right\rangle = D W_i \frac{\partial C}{\partial x} \right] \quad (6-6)$$

In each element, there are n_p sets of equations depending on the total nodal number of the each element, because the given function is evaluated at each node by using each nodal weighting function. If a linear basis function is used (that is when total nodal number of each element is two), then the element matrix is 2 by 2. For quadratic basis function the element matrix is 3 by 3 and if a hermitian basis function is used (this is the case when the first derivative of independent variable is also considered), the matrix is 4 by 2.

After computing the element matrices with respect to the time derivative term (matrix ET), dispersive flux term (matrix ED), advective flux term (matrix EV), the element matrices are assembled for the whole elements. These matrices are derived in next sections. The attained simultaneous equations are solved at each time step. In a one dimensional case, the global matrix is tri-diagonal form, so the Thomas algorithm is used.

Along the time domain, the generalized Crank-Nicholson method is used to reduce the oscillation and smearing effect in the case of the convective transport. To solve this problem, the upstream weighting technique is introduced to the hermitian basis function.

Four finite element methods have been used in this dissertation. They are different by the basis functions: linear, quadratic, hermitian, upstream weighted hermitian.

6.2.1.1 FEM with Linear Basis Function

The linear basis functions for element i and $i-1$ are shown in Figure 7. Considering the Figure 7, from the linear interpolation,

$$C(x) = C_{i-1} + (C_i - C_{i-1}) \frac{X - X_{i-1}}{X_i - X_{i-1}} = C_{i-1} \frac{X_i - X}{\Delta X_i} + C_i \frac{X - X_i}{\Delta X_i} \quad (6-7)$$

$$\text{So, } N_{i-1} = \frac{X_i - X}{\Delta X_i} \quad N_i = \frac{X - X_{i-1}}{\Delta X_i}$$

By using the concept of linear mapping, the original global coordinates are converted to the local coordinates. This transformation simplifies the computation procedure. The basis functions in the local domain are:

$$N_1 = \frac{1 - \xi}{2} \quad N_2 = \frac{1 + \xi}{2} \quad (6-8)$$

Here, node one in local domain corresponds to the inlet node in each element.

So, in element i , $N_1 = N_{i-1}$

Evaluating the element matrices by using the local basis functions :

$$[ET] = \int \begin{bmatrix} N_1 N_1 & N_2 N_1 \\ N_1 N_2 & N_2 N_2 \end{bmatrix} dx = \frac{\Delta x}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad (6-9-1)$$

$$[EV] = \int \begin{bmatrix} \frac{dN_1}{dx} N_1 & \frac{dN_2}{dx} N_1 \\ \frac{dN_1}{dx} N_2 & \frac{dN_2}{dx} N_2 \end{bmatrix} dx = \frac{1}{2} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \quad (6-9-2)$$

$$[ED] = \int \begin{bmatrix} \frac{dN_1}{dx} \frac{dN_1}{dx} & \frac{dN_2}{dx} \frac{dN_1}{dx} \\ \frac{dN_1}{dx} \frac{dN_2}{dx} & \frac{dN_2}{dx} \frac{dN_2}{dx} \end{bmatrix} dx = \frac{1}{\Delta x} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad (6-9-3)$$

The elements of the matrices can be easily evaluated by using the local coordinates.

$$\int N_1 N_1 dx = \int N_1 N_1 \frac{dx}{d\xi} d\xi = \frac{\Delta x}{3} \quad (6-10-1)$$

$$\int \frac{dN_1}{dx} N_1 dx = \int \frac{dN_1}{d\xi} \frac{d\xi}{dx} N_1 \frac{dx}{d\xi} d\xi = -\frac{1}{2} \quad (6-10-2)$$

$$\int \frac{dN_1}{dx} \frac{dN_1}{dx} dx = \int \frac{dN_1}{d\xi} \frac{d\xi}{dx} \frac{dN_1}{d\xi} d\xi = \frac{1}{\Delta x} \quad (6-10-3)$$

After assembling all the element matrices and using the generalized Crank-Nicholson scheme for the time derivative, the global matrix becomes:

$$[A]\{C\} = \{f\} \quad (6-11-1)$$

where,

$$[A] = \sum \left(\frac{1}{\Delta t} [ET] + \varepsilon (V[EV] + D[ED]) \right)^{n+1} \quad (6-11-2)$$

$$\{f\} = \sum \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}^{n+1}$$

$$+ \left(\frac{1}{\Delta t} [ET] + (\varepsilon - 1)(V[EV] + D[ED]) \right) \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}^n \quad (6-11-3)$$

ε = weighting factor in generalized Crank-Nicholson scheme

6.2.1.2 FEM with Quadratic Basis Function

The derivation of the Galerkin form is very similar to the case of the linear basis function except for the element matrices of the basis function. Thus, the same algorithm is applicable to the boundary conditions. But if the problem

is in two dimensional space, then the algorithm for the boundary condition is different. The global matrix is a banded matrix with a maximum band of 5, so tridiagonal algorithm cannot be used. An asymmetric banded matrix solver is provided. To evaluate the integration of the basis function, the Lagrangian polynomial and the character of the Beta function are used.

From Huyakorn and Pinder (1983), the Lagrangian basis function for node i is expressed as:

$$N_i(L_1, L_2) = \psi_p(L_1) \phi_q(L_2) \quad (6-12)$$

where,

$$L_1 = \frac{x - x_{m+1}}{x_1 - x_{m+1}} \quad L_2 = \frac{x - x_1}{x_{m+1} - x_1}$$

$$\psi_p = \Pi\left(\frac{mL_1 - k + 1}{k}\right) \quad \phi_q = \Pi\left(\frac{mL_2 - k + 1}{k}\right)$$

p : the number of nodes on the right side

q : the number of the nodes on the left side

So, for the quadratic element, the Lagrangian basis functions are:

$$N_1 = L_1(2L_1 - 1) \quad N_2 = 4L_1L_2 \quad N_3 = L_2(2L_2 - 1) \quad (6-13)$$

$$\frac{dN_1}{dx} = \frac{1 - 4L_1}{\Delta x} \quad \frac{dN_2}{dx} = \frac{4(L_1 - L_2)}{\Delta x} \quad \frac{dN_3}{dx} = \frac{4L_2 - 1}{\Delta x} \quad (6-14)$$

Figure 11 shows L_1 , L_2 and the quadratic basis function.

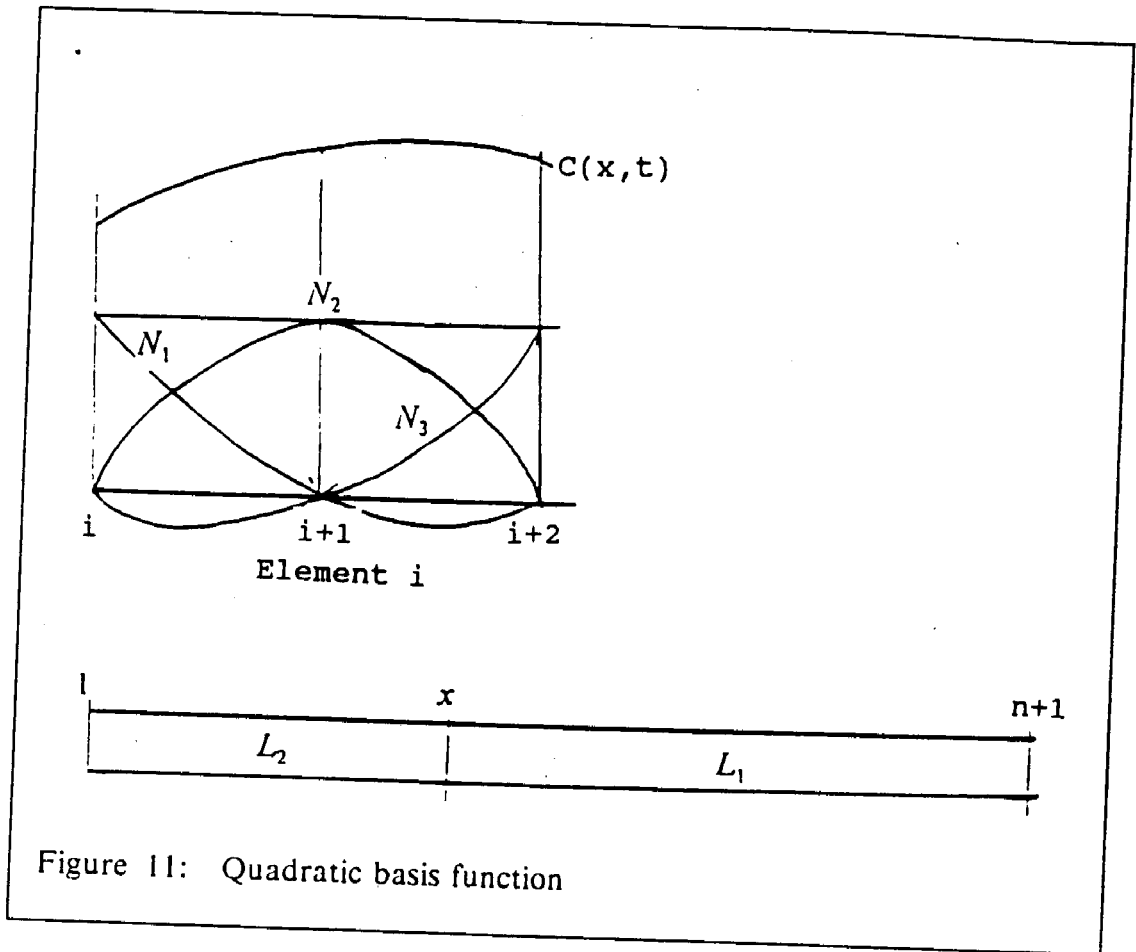


Figure 11: Quadratic basis function

By the concept of the standard Beta function,

$$\int L_1^a L_2^b dx = \frac{\Delta x a! b!}{(a+b+1)!}$$

By using the above formula, the evaluation of the quadratic basis functions can be easily implemented.

$$\langle N_1, N_1 \rangle = \langle (L_1(2L_1 - 1))^2 \rangle = \Delta x \frac{2}{15} \quad (6-15-1)$$

$$\left\langle \frac{dN_1}{dx}, N_2 \right\rangle = \langle (1 - 4L_1)4L_1, L_2 \rangle = -\frac{2}{3} \quad (6-15-2)$$

$$\left\langle \frac{dN_1}{dx}, \frac{dN_1}{dx} \right\rangle = \frac{1}{\Delta x} \langle (1-4L_1), (1-4L_1) \rangle = \frac{7}{3} \quad (6-15-3)$$

$$[ET] = \int \begin{bmatrix} N_1 N_1 & N_2 N_1 & N_3 N_1 \\ N_1 N_2 & N_2 N_2 & N_3 N_2 \\ N_1 N_3 & N_2 N_3 & N_3 N_3 \end{bmatrix} dx = \frac{\Delta x}{30} \begin{bmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{bmatrix} \quad (6-16-1)$$

$$[EV] = \int \begin{bmatrix} \frac{dN_1}{dx} N_1 & \frac{dN_2}{dx} N_1 & \frac{dN_1}{dx} N_1 \\ \frac{dN_1}{dx} N_2 & \frac{dN_2}{dx} N_2 & \frac{dN_1}{dx} N_2 \\ \frac{dN_1}{dx} N_3 & \frac{dN_2}{dx} N_3 & \frac{dN_1}{dx} N_3 \end{bmatrix} dx$$

$$= \frac{1}{6} \begin{bmatrix} -3 & 4 & -1 \\ -4 & 0 & 4 \\ 1 & -4 & 3 \end{bmatrix} \quad (6-16-2)$$

$$[ED] = \int \begin{bmatrix} \frac{dN_1}{dx} \frac{dN_1}{dx} & \frac{dN_2}{dx} \frac{dN_1}{dx} & \frac{dN_3}{dx} \frac{dN_1}{dx} \\ \frac{dN_1}{dx} \frac{dN_2}{dx} & \frac{dN_2}{dx} \frac{dN_2}{dx} & \frac{dN_3}{dx} \frac{dN_2}{dx} \\ \frac{dN_1}{dx} \frac{dN_3}{dx} & \frac{dN_2}{dx} \frac{dN_3}{dx} & \frac{dN_3}{dx} \frac{dN_3}{dx} \end{bmatrix} dx$$

$$= \frac{1}{3\Delta x} \begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix} \quad (6-16-3)$$

6.2.1.3 FEM with Hermitian Basis Function

The approximating function is given by:

$$C = C_j N_{0j} + \frac{dC_j}{dx} N_{1j} \quad (6-17)$$

The basis function must be chosen to satisfy the continuity of itself (C_0 continuity) and the derivative at each node (C_1 continuity).

The finite element scheme is expressed by:

$$\begin{aligned} \frac{d}{dt} \langle C_j N_{0j} + \frac{dC_j}{dx} N_{1j}, \delta(x - x_i) \rangle + V \langle C_j \frac{dN_{0j}}{dx} + \frac{dC_j}{dx} \frac{dN_{1j}}{dx}, \delta(x - x_i) \rangle \\ - D \langle C_j \frac{d^2 N_{0j}}{dx^2} + \frac{dC_j}{dx} \frac{d^2 N_{1j}}{dx^2}, \delta(x - x_i) \rangle = 0 \quad (6-18) \end{aligned}$$

Owing to the property of the delta function $\delta(x - x_i)$, the basis functions are evaluated at the Gaussian points.

$$[ET] = \begin{bmatrix} N_{01}(x_1) & N_{11}(x_1) & N_{02}(x_1) & N_{12}(x_1) \\ N_{01}(x_2) & N_{11}(x_2) & N_{02}(x_2) & N_{12}(x_2) \end{bmatrix} \quad (6-19-1)$$

$$[EV] = \begin{bmatrix} \frac{dN_{01}}{dx}(x_1) & \frac{dN_{11}}{dx}(x_1) & \frac{dN_{02}}{dx}(x_1) & \frac{dN_{12}}{dx}(x_1) \\ \frac{dN_{01}}{dx}(x_2) & \frac{dN_{11}}{dx}(x_2) & \frac{dN_{02}}{dx}(x_2) & \frac{dN_{12}}{dx}(x_2) \end{bmatrix} \quad (6-19-2)$$

$$[ED] = - \begin{bmatrix} \frac{d^2 N_{01}}{dx^2}(x_1) & \frac{d^2 N_{11}}{dx^2}(x_1) & \frac{d^2 N_{02}}{dx^2}(x_1) & \frac{d^2 N_{12}}{dx^2}(x_1) \\ \frac{d^2 N_{01}}{dx^2}(x_2) & \frac{d^2 N_{11}}{dx^2}(x_2) & \frac{d^2 N_{02}}{dx^2}(x_2) & \frac{d^2 N_{12}}{dx^2}(x_2) \end{bmatrix} \quad (6-19-3)$$

Figure 12 shows the Hermitian basis functions in the local coordinate.

$$N_{01} = 0.25(\xi - 1)^2(\xi + 2), \quad N_{02} = -0.25(\xi + 1)^2(\xi - 2) \quad (6-20-1)$$

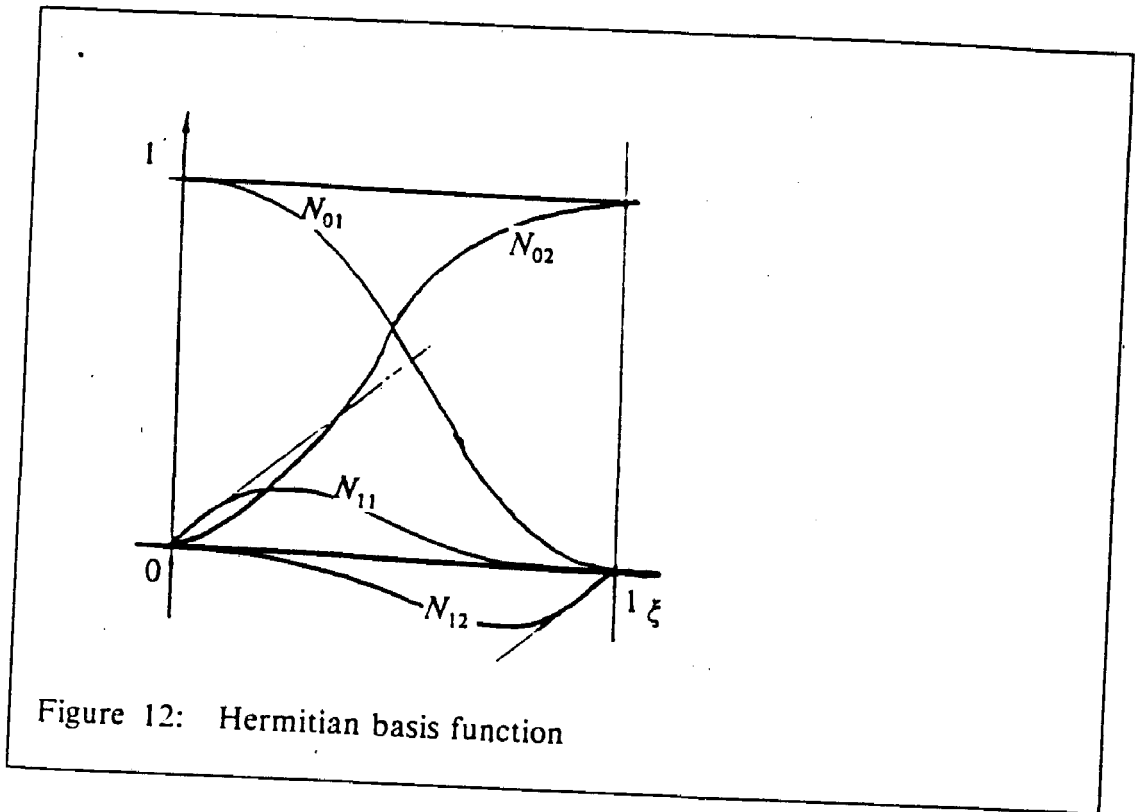


Figure 12: Hermitian basis function

$$N_{11} = 0.25(\xi - 1)^2(\xi + 2)\frac{\Delta x}{2}, \quad N_{01} = 0.25(\xi + 1)^2(\xi - 1)\frac{\Delta x}{2} \quad (6-20-2)$$

$$\frac{dN_{01}}{dx} = \frac{1}{2\Delta x}(\xi - 1)(3\xi + 3), \quad \frac{dN_{02}}{dx} = \frac{-1}{2\Delta x}(\xi + 1)(3\xi - 3) \quad (6-21-1)$$

$$\frac{dN_{11}}{dx} = \frac{1}{4}(\xi - 1)(3\xi + 1), \quad \frac{dN_{12}}{dx} = \frac{1}{4}(\xi + 1)(3\xi - 1) \quad (6-21-2)$$

$$\frac{d^2N_{01}}{dx^2} = \frac{1}{\Delta x^2}(6\xi), \quad \frac{d^2N_{02}}{dx^2} = \frac{-1}{\Delta x^2}(6\xi) \quad (6-22-1)$$

$$\frac{d^2N_{11}}{dx^2} = \frac{1}{2\Delta x}(6\xi - 2), \quad \frac{d^2N_{12}}{dx^2} = \frac{1}{2\Delta x}(6\xi + 2) \quad (6-22-2)$$

The Gaussian points are $x_1 = -0.57735$ $x_2 = 0.57735$. Therefore,

$$[ET] = \begin{bmatrix} 0.8849 & 0.1314\Delta x & 0.1151 & -0.03522\Delta x \\ 0.1151 & 0.03522\Delta x & 0.8849 & -0.1314\Delta x \end{bmatrix} \quad (6-23-1)$$

$$[EV] = \begin{bmatrix} \frac{-1}{\Delta x} & 0.28867 & \frac{1}{\Delta x} & -0.28867 \\ \frac{-1}{\Delta x} & -0.28867 & \frac{1}{\Delta x} & -0.28867 \end{bmatrix} \quad (6-23-2)$$

$$[ED] = \frac{-1}{\Delta x^2} \begin{bmatrix} -3.4641 & -2.7321\Delta x & 3.4641 & -0.7321\Delta x \\ 3.4641 & 0.7321\Delta x & -3.4641 & 2.7321\Delta x \end{bmatrix} \quad (6-23-3)$$

As shown above, Hermite basis FEM requires no integration of the basis functions. The assembling process is the same as the linear basis FEM except for the shape of the element matrices, which is 4 by 2. Because of the two sets of continuity, resulting assembled global matrix requires two more equations to solve the system of the algebraic equations.

If we denote the global matrix as $[A]$,

and let $[ET] = \frac{1}{\Delta t}[ET]$, $[EV] = V [EV]$, $[ED] = D [ED]$, then

$$A_{1,1} = ET_{1,1} + \varepsilon(EV_{1,1} + ED_{1,1}) \quad (6-24-1)$$

$$A_{1,2} = ET_{1,2} + \varepsilon(EV_{1,2} + ED_{1,2}) \quad (6-24-2)$$

$$A_{2ne+2,2ne+1} = ET_{2,3} + \varepsilon(EV_{2,3} + ED_{2,3}) \quad (6-24-3)$$

$$A_{2ne+2,2ne+2} = ET_{2,4} + \varepsilon(EV_{2,4} + ED_{2,4}) \quad (6-24-4)$$

where ne is total element number.

6.2.1.4 FEM with Modified Hermitian Basis Function

The Hermitian basis function can be modified by adding the asymmetric upstream weighting term to the standard Hermitian basis function. Except for the basis functions, the entire procedure is exactly the same as the case of the FEM with Hermitian basis function. To evaluate the basis functions and their first and second derivatives, the following equations are used.

$$N_{01} = 0.25(\xi - 1)^2(\xi + 2) + F(\xi) \quad (6 - 25 - 1)$$

$$N_{02} = -0.25(\xi + 1)^2(\xi - 2) - F(\xi) \quad (6 - 25 - 2)$$

$$N_{11} = (0.25(\xi - 1)^2(\xi + 2) + G(\xi))\frac{\Delta x}{2} \quad (6 - 26 - 1)$$

$$N_{12} = (0.25(\xi + 1)^2(\xi - 2) - G(\xi))\frac{\Delta x}{2} \quad (6 - 26 - 2)$$

The asymmetric weighting term F and G are subject to the following constraints :

$$F(\xi) = G(\xi) = \frac{dF(\xi)}{d\xi} = \frac{dG(\xi)}{d\xi} = 0 \quad \text{for } \xi = -1 \text{ and } 1 \quad (6 - 27 - 1)$$

So, F and G are evaluated by the fourth order polynomials as follow :

$$F(\xi) = \alpha_0(\xi^2 - 1)^2 \quad G(\xi) = \alpha_1(\xi^2 - 1)^2 \quad (6 - 27 - 2)$$

The first derivatives are :

$$\frac{dN_{01}}{dx} = \frac{1}{2\Delta x}((\xi - 1)(3\xi + 3) + 16\alpha_0\xi(\xi^2 - 1)) \quad (6 - 28 - 1)$$

$$\frac{dN_{02}}{dx} = \frac{-1}{2\Delta x}((\xi + 1)(3\xi - 3) - 16\alpha_0\xi(\xi^2 - 1)) \quad (6-28-2)$$

$$\frac{dN_{11}}{dx} = \frac{1}{4}(\xi - 1)(3\xi + 1) + 4\alpha_1\xi(\xi^2 - 1) \quad (6-29-1)$$

$$\frac{dN_{12}}{dx} = \frac{1}{4}(\xi + 1)(3\xi - 1) - 4\alpha_1\xi(\xi^2 - 1) \quad (6-29-2)$$

The second derivatives are :

$$\frac{d^2N_{01}}{dx^2} = \frac{1}{\Delta x^2}((6\xi) + 16\alpha_0(3\xi^2 - 1)) \quad (6-30-1)$$

$$\frac{d^2N_{02}}{dx^2} = \frac{-1}{\Delta x^2}((6\xi) + 16\alpha_0(3\xi^2 - 1)) \quad (6-30-2)$$

$$\frac{d^2N_{11}}{dx^2} = \frac{1}{\Delta x}((3\xi - 1) + 8\alpha_1(3\xi^2 - 1)) \quad (6-31-1)$$

$$\frac{d^2N_{12}}{dx^2} = \frac{1}{\Delta x}((3\xi + 1) - 8\alpha_1(3\xi^2 - 1)) \quad (6-31-2)$$

When $\alpha_0 = \alpha_1 = 0$, this case is same as the standard Hermite basis FEM. The degree of upstream weighting depends upon the magnitude of the factor α_0 and α_1 .

6.2.1.5 Finite Difference Method

To compare the solution of FEM, the generalized Crank-Nicholson method has been used for the FDM scheme. The derivation of the algorithm is not

explained here, because of its simplicity. The algorithm for the boundary conditions is explained below. Generally speaking, if Thomas algorithm can be used for the solution of the system of equations, then boundary conditions are usually handled by the modification of the coefficient of the tridiagonal term. Denoting the tridiagonal term as a_i, b_i, c_i and the load vector as d_i , the boundary conditions are incorporated as follows :

(a) 1st type boundary

$$b_1 = 1 \quad c_1 = 0 \quad d_1 = C_L \quad (6-40-1)$$

$$a_{np} = 0 \quad b_{np} = 1 \quad d_{np} = C_R \quad (6-40-2)$$

(b) 2nd type boundary

This is the case when $\frac{\partial C}{\partial x} = C_L$ or C_R

Using the concept of imaginary node (node 0 or node $np + 1$),

$$\frac{\partial C}{\partial x} = \frac{C_2 - C_0}{2\Delta X_1} = C_L, \quad \frac{\partial C}{\partial x} = \frac{C_{np+1} - C_{np-1}}{2\Delta X_{np-1}} = C_R$$

$$C_0 = C_2 - 2\Delta x_1 C_L, \quad C_{np+1} = C_{np-1} + 2\Delta_{np-1} C_R$$

Therefore,

$$c_1 = c_1 + a_1, \quad d_1 = d_1 + 2\Delta x_1 a_1 C_L \quad (6-41-1)$$

$$a_{np} = a_{np} + c_{np}, \quad d_{np} = d_{np} - 2\Delta x_{np-1} c_{np} C_R \quad (6-41-2)$$

(c) 3rd type boundary

This is the case when $\frac{\partial C}{\partial x} = k_L(C_1 - C_L)$

$$\frac{C_2 - C_0}{2\Delta x_1} = k_L(C_1 - C_L), \quad \frac{C_{np+1} - C_{np-1}}{2\Delta x_{np-1}} = k_R(C_{np} - C_R)$$

So,

$$b_1 = b_1 - 2\Delta x_1 k_L a_1, \quad c_1 = c_1 + a_1, \quad d_1 = d_1 - 2\Delta x_1 a_1 k_L C_L \quad (6-42-1)$$

$$b_{np} = b_{np} - 2\Delta x_{np-1} k_R c_{np}, \quad a_{np} = a_{np} + c_{np}$$

$$d_{np} = d_{np} - 2\Delta x_{np-1} c_{np} k_R C_R \quad (6-42-2)$$

where, np = number of total nodal points.

6.2.2 Two Dimensional Upstream Weighted FEM

The formulation of the finite element scheme is explained as follows. Let

the differential operator be $L(c) = \frac{\partial C}{\partial t} + \bar{V}\nabla C - \underline{D}\nabla^2 C$.

Applying finite element scheme, $\int L(C)W_i(x,y)dR = 0$

where $C = \sum_{j=1}^{nn} C_j(t)N_j(x,y)$, W_i = weighting function.

$$\int \frac{\partial C}{\partial t} N_i dR + \int (\bar{V}\nabla C - \underline{D}\nabla^2 C) W_i dR = 0 \quad (6-43)$$

Using integration by parts and summing over the whole discretized domain,

the equation (6-43) becomes:

$$\sum_{e=1}^{nel} \sum_{j=1}^{nn} [\langle W_i, N_j \rangle \frac{dC_j}{dt} + \vec{V} \langle W_i, \nabla N_j \rangle C_j + \underline{D} \langle \nabla W_i, \nabla N_j \rangle C_j] \\ = \int W_i \underline{D} \nabla C \cdot \vec{n} dB \quad (6-44)$$

To handle the irregular geometry, coordinate transformation is performed by the concept of the linear mapping. Figure 6 showed the transformation of the global coordinate to the local one. The linear basis function is used along the x and y axis direction. The assembling procedure is the same as the one dimensional FEM except the element matrices.

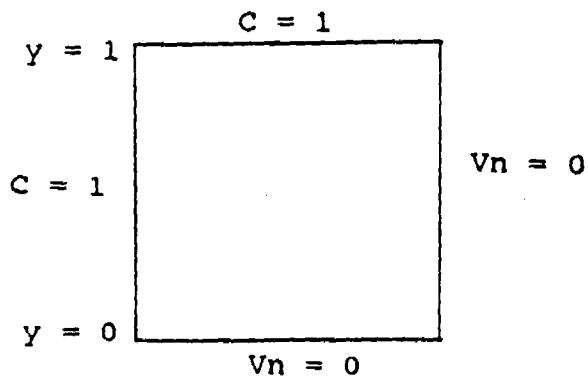
To test numerical oscillation in the case of the advective flux dominant system, the simple solute transport equation is solved in one and two dimensions. The one dimensional result is same as the result by Pinder and Gray (1977). Problem domain and data are shown in Figure 13. The comparison has been made for several basis functions in one and two dimensions in Figure 14.

One Dimension

$$C = 1 \quad \text{-----} \quad V_n = 0$$

$x = 0$ $x = 1$

Two Dimension



Dispersive Coefficient	$D = 0.0001725$
Velocity	$V = 0.369$
Spatial Step Size	$dx = 0.025$ $dy = 0.025$
Time Step Size	$dt = 0.025$

Figure 13: Problem data in solute transport system

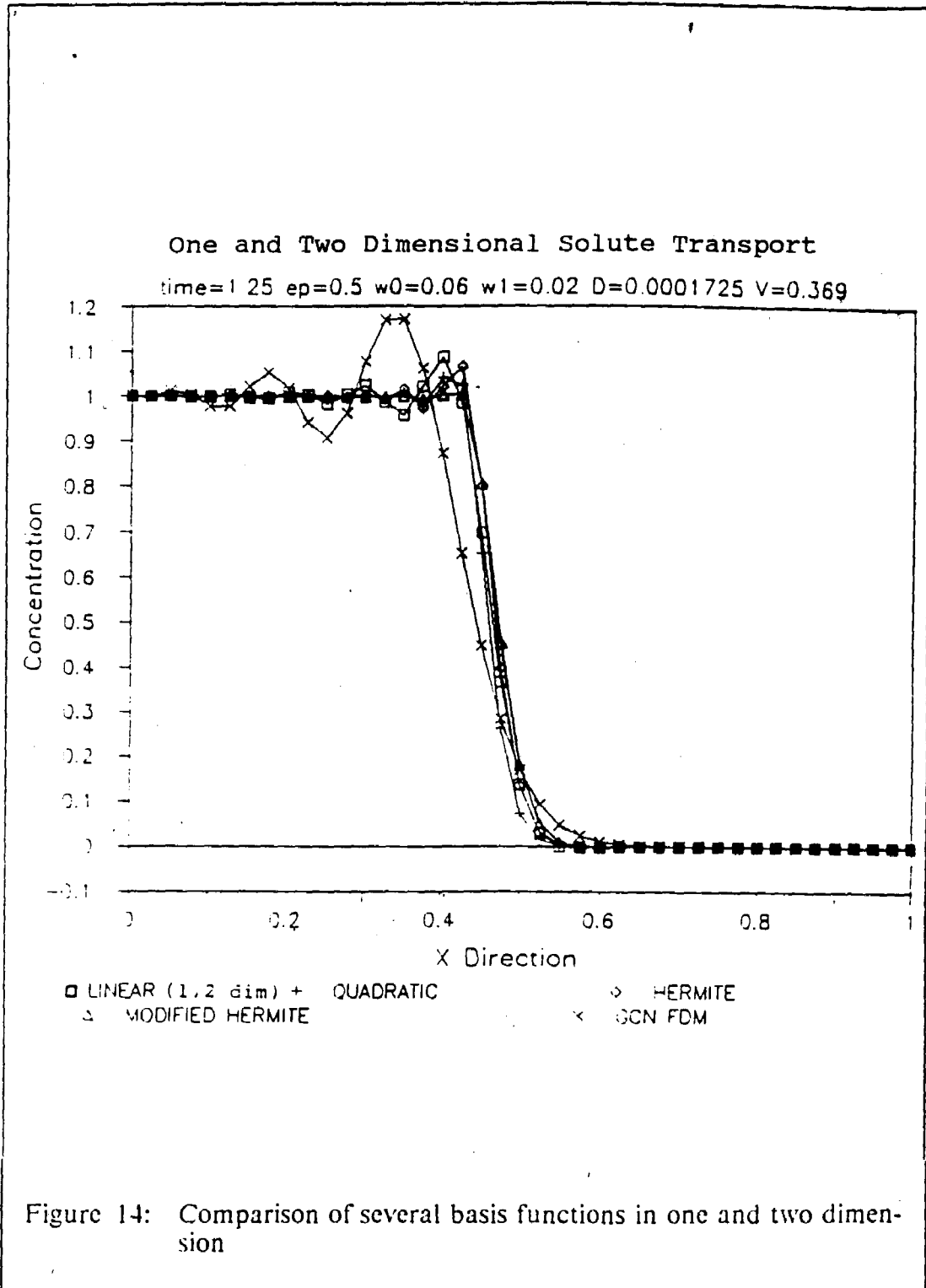


Figure 14: Comparison of several basis functions in one and two dimension

6.3 MULTIPHASE FLOW SYSTEM

If capillary pressure is dominant, then the governing equations are :

$$(S_w S_{ww} + \phi \rho_w \frac{\partial S_w}{\partial h_{ow}}) \frac{\partial h_w}{\partial t} = \nabla(\rho_w \underline{K}_w (\nabla h_w + \vec{j})) + g_w \quad (6-45-1)$$

$$(S_o S_{oo} + \phi \rho_o \frac{\partial S_o}{\partial h_{ow}}) \frac{\partial h_o}{\partial t} = \nabla(\rho_o \underline{K}_o (\nabla h_o + \vec{j})) + g_o \quad (6-45-2)$$

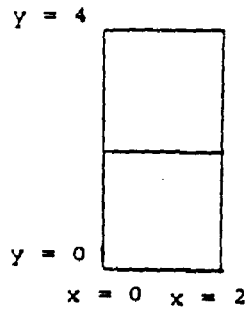
Using the above system of equations, the multidimensional capability of the code has been tested by simulating the small one, two and three dimensional problems. Problem domain and data are shown in Figure 15. Results of this multidimensional problem are shown in Figure 16. Figure 17 shows the parameter dependency in a large two dimensional problem.

Problem Domain

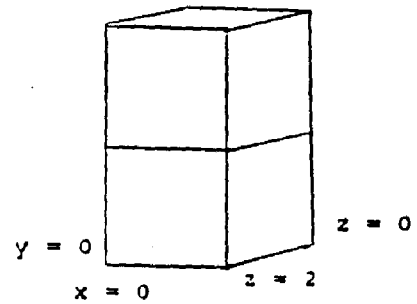
One Dimension



Two Dimension



Three Dimension



Boundary Conditions

$$h_w = 2, \quad \frac{\partial h_w}{\partial n} = 0 \text{ at } x = 0$$

$$h_w = 0.01, \quad \frac{\partial h_w}{\partial n} = 0 \text{ at } x = 4$$

$$h_w = 2, \quad \frac{\partial h_w}{\partial n} = 0 \text{ at } y = 0$$

$$h_w = 0.01, \quad \frac{\partial h_w}{\partial n} = 0 \text{ at } y = 4$$

Initial Conditions

$$h_w = 4 - x$$

$$h_o = h_w - 0.3$$

$$h_w = 4 - y$$

$$h_o = h_w - 0.3$$

Physical Parameters

Saturated conductivity of water phase $K_{sw} = 1.008 \text{ m/day}$

Ratio of viscosity between oil and water phase $\mu_m = 0.5$

Ratio of density between oil and water phase $\rho_m = 1.2$

Figure 15: Data of multidimensional multiphase flow problem

One Dimension

	TIME=	0.0100		TIME=	0.0541		TIME=	0.1078		
IZ=1	Y/X	0.00	2.00	4.00	IZ=1	Y/X	0.00	2.00	4.00	
	Y/X	0.00	0.20	0.20	0.77	0.00	0.20	0.21	0.77	
					IZ=1	Y/X	0.00	2.00	4.00	
						Y/X	0.00	0.20	0.26	0.77

Two Dimension

	TIME=	0.0100		TIME=	0.0520		TIME=	0.1061		
IZ=1	Y/X	0.00	2.00	4.00	0.77	0.77	IZ=1	Y/X	0.00	2.00
	Y/X	4.00	0.77	0.77	2.00	0.23	0.23	4.00	0.77	0.77
		2.00	0.20	0.20	0.00	0.20	0.20	2.00	0.33	0.33
		0.00	0.20	0.20				0.00	0.20	0.20

Three Dimension

	TIME=	0.0100		TIME=	0.0559		TIME=	0.1065		
IZ=1	Y/X	0.00	2.00	4.00	0.77	0.77	IZ=1	Y/X	0.00	2.00
	Y/X	4.00	0.77	0.77	2.00	0.27	0.27	4.00	0.77	0.77
		2.00	0.20	0.20	0.00	0.20	0.20	2.00	0.40	0.40
		0.00	0.20	0.20				0.00	0.20	0.20
IZ=2	Y/X	0.00	2.00	4.00	0.77	0.77	IZ=2	Y/X	0.00	2.00
	Y/X	4.00	0.77	0.77	2.00	0.27	0.27	4.00	0.77	0.77
		2.00	0.20	0.20	0.00	0.20	0.20	2.00	0.40	0.40
		0.00	0.20	0.20				0.00	0.20	0.20

Figure 16: Result of multidimensional multiphase flow problem

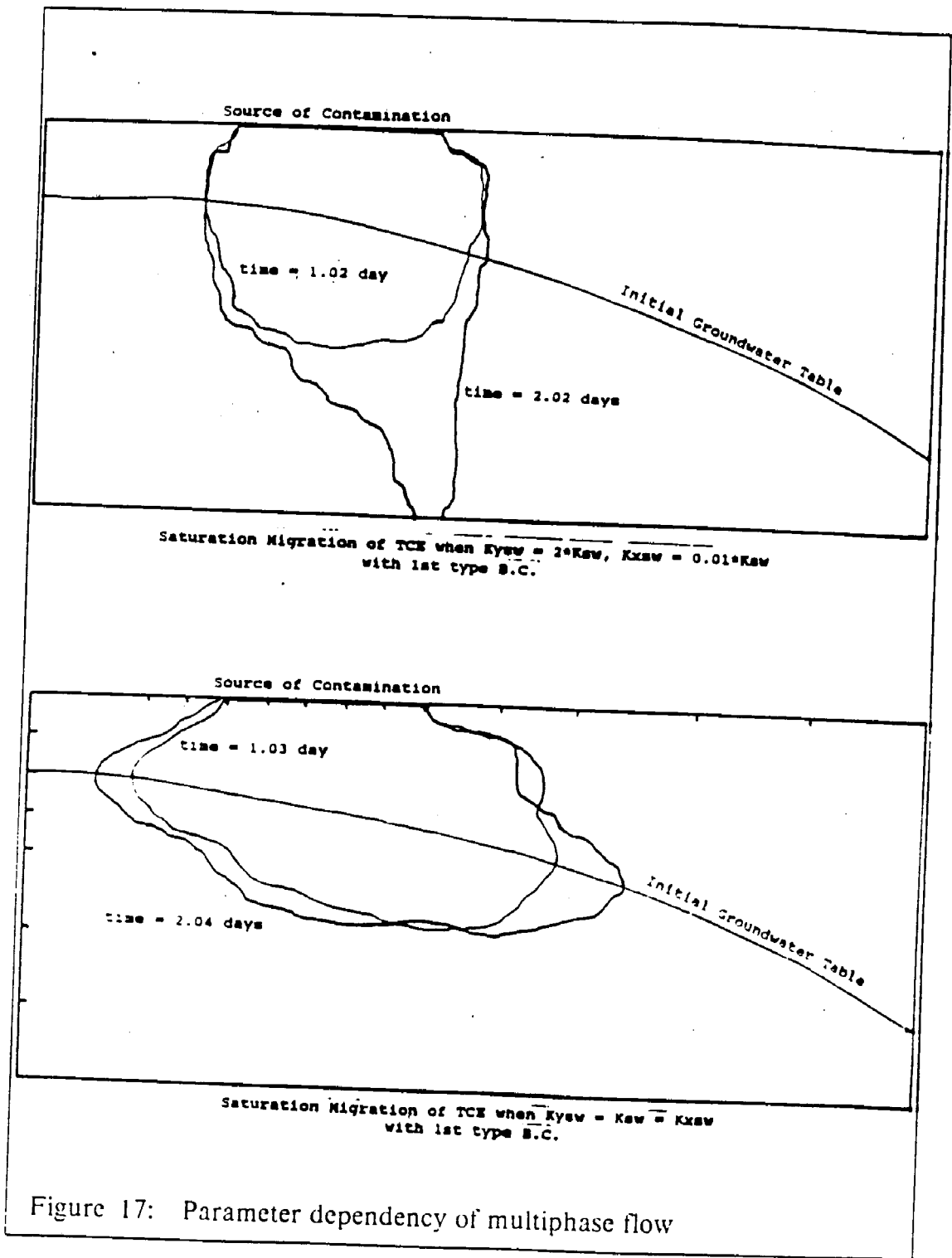


Figure 17: Parameter dependency of multiphase flow

6.4 COMPOSITE MULTIPHASE CONTAMINANT MIGRATION

Contaminant migration is relatively slower than the case of oil recovery.

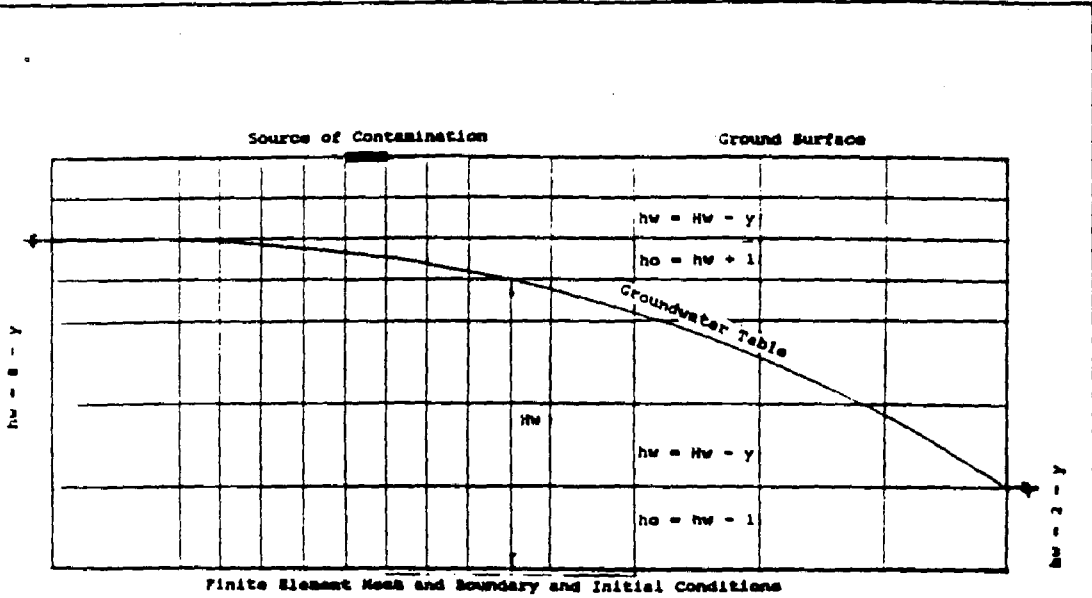
Thus, the second approach (capillary dominant) is used.

6.4.1 Composite Multiphase Flow

Primal variables are pressure head and saturation, as presented in the following equations :

$$\begin{aligned} & \sum_{\alpha=1}^3 \left\{ \left(\sum_{\beta=1}^3 (\rho_{\beta} w_{\beta}^i S_{\beta} S_{\beta\alpha}) + \phi \rho_{\alpha} w_{\alpha}^i \frac{\partial S_{\alpha}}{\partial h_{\alpha}} \right) \frac{\partial h_{\alpha}}{\partial t} + \nabla (\rho_{\alpha} w_{\alpha}^i K_{\alpha} (\nabla h_{\alpha} + \frac{\rho_{\alpha} \vec{j}}{\rho_w})) \right\} \\ & = \sum_{\alpha=1}^3 \left\{ \nabla (\phi S_{\alpha} D_{\alpha}^i \nabla (\rho_{\alpha} w_{\alpha}^i)) + g_{\alpha}^i - \phi S_{\alpha} \rho_{\alpha} \frac{\partial w_{\alpha}^i}{\partial t} \right\} - \frac{\partial}{\partial t} ((1 - \phi) \rho_s w_s^i) \quad (6-46) \end{aligned}$$

Problem domain and data are shown in Figure 18. The result of saturation migration is shown in Figure 19.



Physical Parameters

- Saturated conductivity of water phase = 0.504 m/day
- Ratio of viscosity between oil and water phase = 0.5
- Ratio of density between oil and water phase = 1.2
- Solubility of TCE into water phase = 1100 ppm
- Residual Saturation = 0.2
- Capillary perssure and saturation relation
 $n = 1.98$ $\lambda_{ow} = 5.2$ $\lambda_{so} = 11.0$ $\lambda_{sw} = 9.9$
- Dispersive Coefficient
 $D_w = 0.5574$ $D_o = 0.2784$ $D_a = 1.0034$

Numerical data

- Error criteria = 0.1 m
- Time increasing factor = 0.1
- Weighting factor of FDM = 0.68

Figure 18: Data of composite multiphase contaminant migration problem

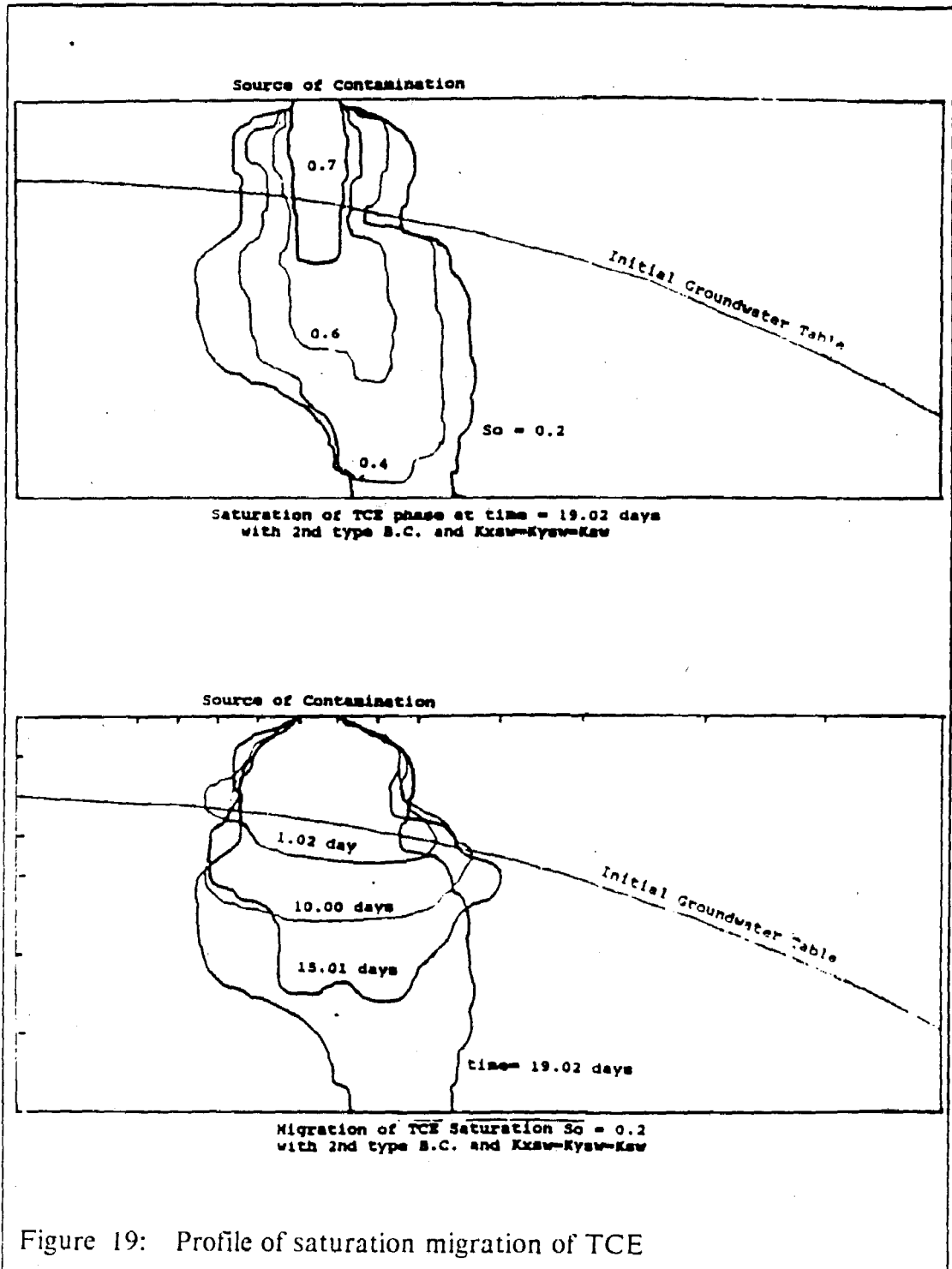


Figure 19: Profile of saturation migration of TCE

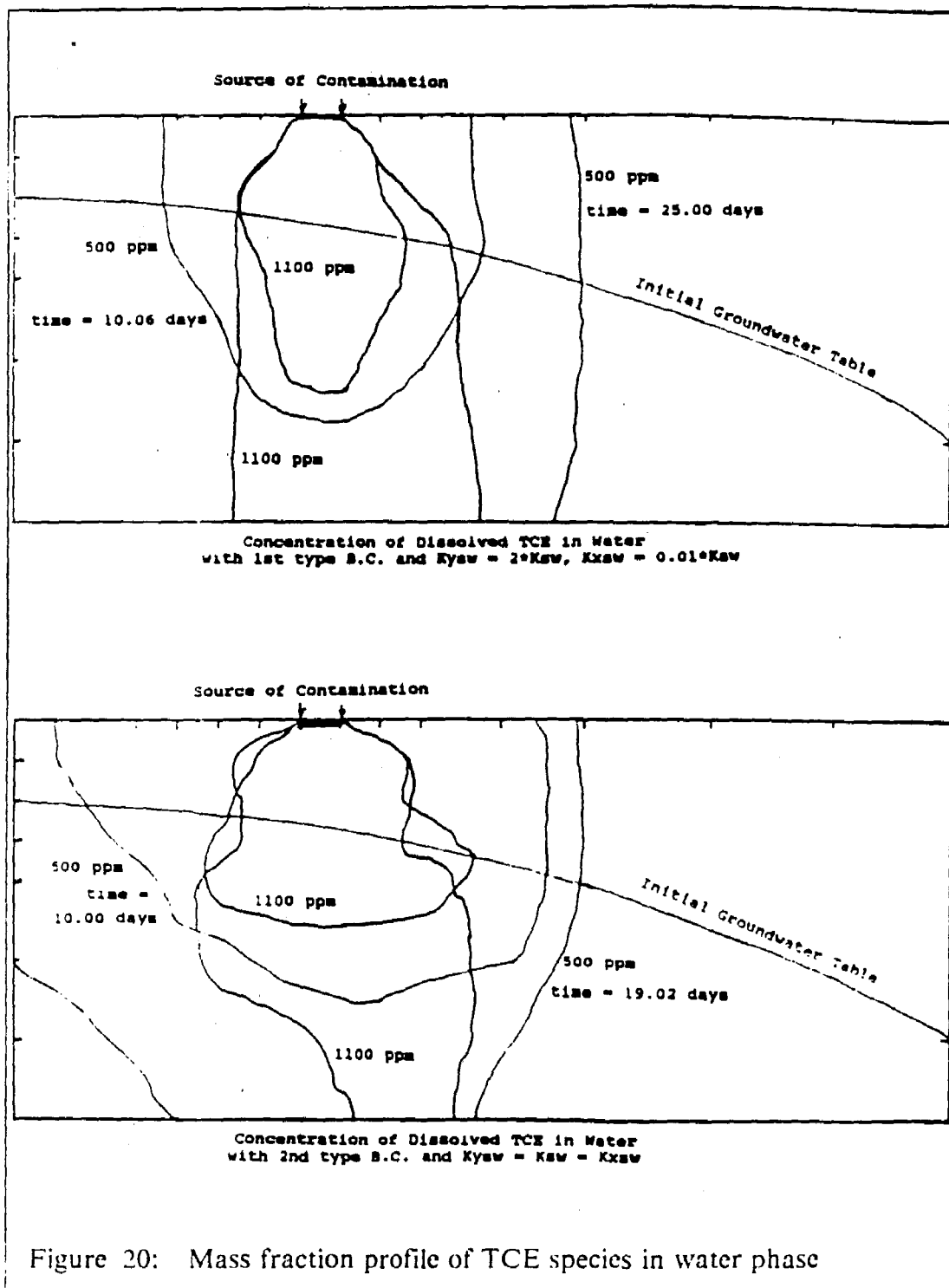
6.4.2 Composite Multiphase Contaminant Transport

The equations for saturation and pressure head terms are the same as in section 6.4.1. An additional primary variable is the mass fraction of species o in the water phase. Arranging the governing equation with partitioning concept with respect to this primal variable, the contaminant transport equation is derived as follows:

$$\begin{aligned}
 & \frac{\partial}{\partial t} (w_w^o (\phi S_w \rho_w + \phi S_o \rho_o H_{ow}^o + \phi S_a \rho_a H_{aw}^o + (1 - \phi) \rho_s H_{sw}^o)) \\
 & = - \nabla (w_w^o (\phi S_w \vec{V}_w + \phi S_o \rho_o H_{ow}^o \vec{V}_o + \phi S_a \rho_a H_{aw}^o \vec{V}_a)) \\
 & \quad + \nabla (\phi S_w \rho_w \underline{D}_w^o \nabla w_w^o) + \nabla (\phi S_o \rho_o \underline{D}_o^o \nabla (H_{ow}^o w_w^o)) \\
 & \quad + \nabla (\phi S_a \rho_a \underline{D}_a^o \nabla (H_{aw}^o w_w^o)) + g_w^o + g_o^o + g_a^o \quad (6-47)
 \end{aligned}$$

where, H_{ow}^o , H_{aw}^o , H_{sw}^o = partitioning coefficients.

The transport of mass fraction is shown in Figure 20 for two cases; (1) first type boundary condition and large vertical fluid conductivity case; (2) second type boundary condition and isotropic fluid conductivity case.



Chapter VII

CONCLUSIONS AND FUTURE RESEARCH

In this study, traditional groundwater flow and contaminant transport equations have been rearranged to explain many different problems by the general comprehensive governing equation.

The governing equations have been developed from averaged parameters which are not derived from a microscopic physical relation, and in this study the groundwater system is very dependent on these average parameters. Therefore, sensitivity analysis, simplification and estimation of these parameters are important. The parameters, constitutive equations and partition concepts of previous research have been addressed as broadly as possible, and some of the evaluation techniques have been improved.

A multidimensional computer code was developed using the upstream weighted quadrilateral element. The numerical difficulties such as instability and nonlinearity which results in extreme computation time have been overcome by using upstream weighting, mass lumping, modified Picard iteration, variable time steps, decoupling and efficient coding using vector and parallel processing.

Hypothetical contaminant migration was simulated in a saturated and unsaturated system. The plots of these simulations are shown in previous section.

The specifics of each output of this dissertation are summarized in the following section.

7.1 UNSATURATED GROUNDWATER FLOW SYSTEM

To verify the procedure and approach presented here, Huyakorn's (1986) results were simulated, as shown in Figure 10. The results of the simulation match Huyakorn's (1986) results very well. The main difficulty of this simulation was the location of the unknown seepage height, which was determined using a moving first-type boundary condition. Cooley's scheme (1983) was used to find the location. 175 iterations were required to satisfy the error criteria of 0.01 m.

7.2 MASS TRANSFER SYSTEM

The model was used to simulate a sharp advancing concentration front in one and two dimensions. As shown in Figure 14, the upstream weighting technique reduced the numerical oscillation of the sharp advancing front, but introduced the numerical dissipation. The solute transport model works well and is the basis of the more advanced system such as composite multiphase mass transfer. When the composite contaminant migration was simulated, the time and spatial step size were first determined by the stability criteria of this system.

7.3 MULTIPHASE FLOW SYSTEM

Symmetrical problems were simulated in one, two and three dimensions. The results were all identical. This provides verification of the multidimensional aspects of the code. The verification of two dimensional multiphase contaminant transport was implemented in field scale. To observe the parameter dependency of the model, the fluid conductivity was changed for several cases. The increased fluid conductivity enhanced the mobility of the phases in roughly one-to-one rate. The derivative of this fluid conductivity acts like an advective flux and numerical oscillation becomes a problem when the derivative term is large, which causes convergence problems in the nonlinear iteration.

The relative permeability depends on the saturation; therefore, if the changing ratio of saturation is large, the relative permeability also causes a convergence problem. Therefore, to insure rapid convergence, the residual saturation must be restricted. A value of 0.2 was used in this simulation.

The constant head pressure boundary acts as a forcing pressure from the boundary. As shown in Figure 17, the river boundary of the water phase slowed the downward migration of the oil phase, and the resulting saturation profile of the oil phase is different from the conceptual model. If the second type boundary condition is used in contaminant migration, the result is identical to conceptual model. Figure 19 shows the above aspect.

The assembled global matrix of three dimensional FEM is at least two orders of magnitude larger than two dimensional case; so efficient matrix storage scheme and matrix solver are imperative. Either asymmetric band matrix sol-

ver or the methodology suggested by Pelka and Peters (1986) for vector processing can be used for this huge matrix. The simulation domain can be reduced by a front tracking method.

7.4 COMPOSITE MULTIPHASE CONTAMINANT MIGRATION

As shown in Figure 20, the dissolved mass fraction of TCE species is large, which confirms empirical field observations that suggest transport of such compounds must be solved using a composite multiphase approach. Both the dissolved and immiscible TCE pose environmental problems. The movement of mass fraction depends on the magnitude of the dispersive coefficients. Because the water phase velocity is small compared to the velocity of the oil phase, the advective flux of mass fraction does not cause stability problem.

7.5 PARAMETER SENSITIVITY IN SIMULATION

The techniques presented in this dissertation are very dependent upon the range of parameters used in the simulation. Since, many of the required parameters have never been measured in an actual groundwater basin, it is difficult to postulate the appropriate value and range of parameters. The pitfalls associated with certain parameters are discussed in this section.

7.5.1 *Fluid Conductivity*

Large values of fluid conductivity and relative permeability cause greater numerical error, which results in more iterations, smaller time steps and spatial increments and more computation time.

7.5.2 *Saturation*

The system of equations becomes singular at the region before the interface. To overcome this problem, hypothetical non-zero values of the saturation derivative and relative permeability are required. In this research, saturation derivative ranging from 10^{-4} to 10^{-5} was required.

7.5.3 *Density*

As in the conceptual model, heavy oil migrates downward. And if density is lighter than water phase, the oil phase is floating over the water phase.

7.5.4 *Initial Conditions*

The system is especially sensitive to the initial conditions of pressure head, saturation and mass fraction. The rate of convergence is highly dependent upon the proper set of initial conditions. Because of the dependency of the saturation to capillary pressure head, the constitutive equation between saturation and capillary head also effects the rate of convergence.

7.5.5 *Boundary Conditions*

Contrary to many existing hypothetical simulation results, first type boundary conditions cannot be used to obtain results similar to conceptual model. For the field problems, it is appropriate to use the second type boundary conditions if there is no first type boundary such as river near the source of contaminant.

7.6 OPTIMIZATION OF CODE WITH VECTOR AND PARALLEL PROCESSING

Even though it was difficult to thoroughly optimize the code for vector processing because of the severe nonlinearity of the given problem and of the recursive relations between processes, the vectorization gave excellent results. The computation time after vectorization was only one third of the time required for scalar processing. The following programming techniques were used to facilitate vectorization.

1. The dimension of all arrays was defined in descending order. ($A(n_p, n_e, n_a)$, where $n_p > n_e > n_a$)
2. About 80% of the computation time was spent in the evaluation of the element matrices. Therefore, the element matrices were evaluated over the whole domain, not over the each element, and assembled later. This facilitates vectorization, but increases the storage requirements.
3. In the case of multiple do-loops, the inner-most do-loop was for the largest array.

4. The recursive variables were evaluated outside of the do-loops by using redundant variables.

7.7 FUTURE RESEARCH

This model is built upon the most current understanding of the multiphase, multicomponent transport process and uses the best available techniques. Nevertheless it should be considered as a development model which is useful to gain practical insight of ground contamination process, as opposed to a model for producing exact answers to specific problems.

Further research is necessary to better define the mechanisms and parameters for the following specific functions of the model:

1. Constitutive equations
 - a. Fluid conductivity
 - b. Relation between saturation and capillary pressure
 - c. Dispersion coefficient
 - d. Chemical and biological equations
2. Numerical method
 - a. Front tracking procedure and moving boundary
 - b. Adoptive mesh generation
 - c. Boundary element algorithm

Verification of the model with field data using a system identification technique is also required.

REFERENCES

1. Abriola, L. M., and G. F. Pinder, A Multiphase Approach to the Modeling of Porous Media Contamination by Organic Compounds, *Water Resour. Res.*, 21(1), 11-26, 1985
2. Allen, M. B., Collocation Techniques for Modeling Compositional Flows in Oil Reservoirs, Lecture Notes in Engineering, Springer-Verlag, 1984.
3. Bachmat, Y., and J. Bear, On the Concept and Size of A Representative Elementary Volume (REV), NATO ASI Series, 5-20, 1986.
4. Bear, J., Hydraulics of Groundwater, McGraw-Hill, Ind., New York, 1979.
5. Borden, R. C., and P. B. Bedient, Transport of Dissolved Hydrocarbons Influenced by Oxygen-Limited Biodegradation, *Water Resour. Res.*, 22(13), 1973-1990, 1986.
6. Buckley, S. E., and M. C. Leverett, Mechanism of Fluid Displacement in Sands, *AIME*, 146, 107-116, 1942.
7. Cooley, R. L., Some New Procedure for Numerical Solution of Variably Saturated Flow Problems, *Water Resour. Res.*, 19(5), 1271-1285, 1983.
8. Corapcioglu, M. Y., and A. L. Baehr, A Compositional Multiphase Model for Groundwater Contamination by Petroleum Products, *Water Resour. Res.*, 23(1), 191-213, 1987.
9. Environmental Protection Agency, Handbook Ground Water, U.S. E.P.A., 1987

10. Faust, C. R., Transport of Immiscible Fluids Within and Below the Unsaturated Zone: A Numerical Model, *Water Resour. Res.*, 21(4), 587-596, 1985.
11. Hassanizadeh, M. and W. G. Gray, General Conservation Equations for Multi-Phase Systems: 1. Averaging Procedure, *Adv. in Water Res.*, 2, Sept., 131-144, 1979a.
12. Hassanizadeh, M. and W. G. Gray, General Conservation Equations for Multi-Phase Systems: 1. Mass, Momenta, Energy, and Entropy Equations, *Adv. in Water Res.*, 2, Dec., 191-203, 1979b.
13. Henry, R. L., and R. S. Metcalfe, Multiple-Phase Generation for Carbon Dioxide Flooding, *Soc. Pet. Eng. J.*, 23, 595-601, 1983.
14. Huyakorn, P. S., and K. Nikuha, Solution of transient transport equation using an upstream finite element scheme, *Appl. Math. Modeling*, 3, 7-17, 1979.
15. Huyakorn, P. S., and G. F. Pinder, Computational Methods in Subsurface Flow, Academic Press, New York., 1983.
16. Huyakorn, P. S., P. F. Anderson, O. Guven, F. J. Molz, A Curvilinear Finite Element Model for Simulating Two-Well Tracer Tests and Transport in Stratified Aquifers, *Water Resour. Res.*, 22(5), 663-678, 1986.
17. Huyakorn, P. S., E. P. Springer, V. Guvanasen, and T. D. Wadsworth, A Three-Dimensional Finite-Element Model for Simulating Water Flow in Variably Saturated Porous Media, *Water Resour. Res.*, 22(13), 1790-1808, 1986.
18. Kaluarachchi, J. J., and J. C. Parker, An Efficient Finite Element Method for Modeling Multiphase Flow, *Water Resour. Res.*, 25(1), 43-54, 1989.

19. Karickhoff, S. W., Organic Pollutant Sorption in Aquatic Systems, *Journal of Hydraulic Engineering*, 110(6), 707-735, 1984.
20. Karplus, W. J., An Electric Circuit Theory Approach to Finite Difference Stability, *Trans. AIEE*, 77, Pt. 1, 210-213, 1958.
21. Kuppusamy, T., J. Sheng, J. C. Parker, and R. J. Lenhard, Finite-Element Analysis of Multiphase Immiscible Flow Through Soils, *Water Resour. Res.*, 23(4), 625-631, 1987.
22. Labrid, J. C., Oil Displacement Mechanisms by Winsor's Type I Micellar Systems, SPE Annual Technical Conference and Exhibition, 23-26, Las Vegas, 1979.
23. Lenhard, R. J., and J. C. Parker Experimental Validation of the Theory of Extending Two-Phase Saturation-Pressure Relations to Three-Fluid Phase Systems, *Water Resour. Res.*, 24(3), 373-380, 1988.
24. Lenhard, R. J., J. H. Dane, J. C. Parker, and J. J. Kaluarachchi, Measurement and Simulation of One-Dimensional Transient Three-Phase Flow for Monotonic Liquid Drainage, *Water Resour. Res.*, 24(6), 853-863, 1988.
25. Leverett, M. C., W. B. Lewis, and M. E. True, Dimensional-model Studies of Oil-field Behavior, *AIME*, 146, 175-193, 1942.
26. Lin, C., G. F. Pinder, E. F. Wood, Water and Trichloroethylene as Immiscible Fluids in Porous Media, Water Resour. Program, 83-WR-2, Princeton University, 1982.
27. Lyman, W. J., W. F. Reehl, and D. H. Rosenblatt, Handbook of chemical property estimation methods, Environmental Behavior of Organic Compounds, McGraw Hill, N. Y., 1982.

28. Molz, F. J., M. A. Widdowson, and L. D. Benefield, Simulation of Microbial Growth Dynamics Coupled to Nutrient and Oxygen Transport in Porous Media, *Water Resour. Res.*, 22(8), 1207-1216, 1986.
29. Odeh, A. S., Mathematical Modeling of the Behavior of Hydrocarbon Reservoirs - the Present and the Future, NATO ASI Series, 823-848, 1986.
30. Osborne, M., and J Sykes, Numerical Modeling of Immiscible Organic Transport at the Hyde Park Landfill, *Water Resour. Res.*, 22(1), 25-33, 1986.
31. Parker, J. C. , R. J. Lenhard, and T. Kuppusamy, A Parametric Model for Constitutive Properties Governing Multiphase Flow in Porous Media, *Water Resour. Res.*, 23(4), 618-624, 1987.
32. Parker, J. C., and R. J. Lenhard, A Model for Hysteretic Constitutive Relations Governing Multiphase Flow, *Water Resour. Res.*, 23(12), 2187-2206, 1987.
33. Pelka, W., and A. Peters, Implementation of Finite Element Groundwater for Models on Vector and Parallel Computers, VI International Conference on Finite Elements in Water Resources, Lisbon, Portugal, 301-312, 1986.
34. Pinder, G. F., E. M. Frind, and S. S. Papadopoulos, Functional Coefficients in the Analysis of Groundwater Flow, *Water Resour. Res.*, 9(1), 222-226, 1973.
35. Pinder, G. F. and W. G. Gray, Finite Element Simulation in Surface and Subsurface Hydrology, Academic Press, New York, 1977.
36. Pinder, G. F., and A. Shapiro, A New Collocation Method for the Solution of the Convection-Dominated Transport Equation, *Water Resour. Res.*, 15(5), 1177-1182, 1979.

37. Pinder, G. F., and L. M. Abriola, On the Simulation of Nonaqueous Phase Organic Compounds in the Subsurface, *Water Resour. Res.*, 22(9), 1095-1195, 1986.
38. Plumb, O. A., and S. Whitaker, Dispersion in Heterogeneous Porous Media, 1. Local Volume Averaging and Large Scale Averaging, 2. Predictions for Stratified and Two-Dimensional Spatially Periodic Systems, *Water Resour. Res.*, 24(7), 913-938, 1988.
39. Quy, N. V., and J. Labrid, A Numerical Study of Chemical Flooding - Comparison With Experiments, *Soc. Pet. Eng. J.*, 23, 461-474, 1983.
40. Voss I. C., A Finite-element Simulation Model for Saturated-Unsaturated, Fluid-Density Dependent Groundwater Flow with Energy Transport or Chemically Reactive Single-Species Solute Transport, U.S. Geological Survey, Water-Resources Investigations Report 84-4369, 1984.
41. Yeh, G. T., and V. S. Tripathi, A Critical Evaluation of Recent Developments in Hydrochemical Transport Models of Reactive Multichemical Components, *Water Resour. Res.*, 25(1), 93-108, 1989.

Appendix A

ALL SYMBOLS AND DEFINITIONS

C = concentration of a one species reaction

C^i = concentration of i species over fluid phases = $\frac{M_\alpha^i}{U_\nu} = S_\alpha C_\alpha^i = \frac{m_\alpha^i}{\phi}$

C_α^i = concentration of i species in α phase = $\frac{M_\alpha^i}{U_\alpha} = \rho_\alpha w_\alpha^i$

C_r = Courant Number = $\frac{V\Delta t}{\Delta x}$

$\underline{\underline{D}}_\alpha^i$ = dispersive coefficient tensor of i species in α phase

$\underline{\underline{D}}_{\alpha,ij}^i$ = dispersive coefficient tensor in direction i, j

$d_{\alpha,t}^i$ = longitudinal dispersion coefficient of i species in α phase = $\alpha_{\alpha t}^i V_\alpha$

$d_{\alpha,t}^i$ = transverse dispersion coefficient of i species in α phase = $\alpha_{\alpha t}^i V_\alpha$

f_t = changing factor for time step size

f_w = fractional flow function in Buckley-Leverett problem

f_{oc} = fraction of organic carbon

\vec{g} = gravity vector

g_w = water pumping(-sign) or recharging(+ sign) rate

g_α^i = unit production rate of i species in α phase = $\frac{G_\alpha^i}{U}$

G_α^i = production rate of i species in α phase

h_B = known pressure head at boundary

h_α = pressure head of α phase in equivalent water height

$h_{\beta\alpha}$ = capillary pressure head between β and α phase = $\frac{P_\alpha}{\rho_w g}$

H_α = total pressure head of α phase = $h_\alpha + \frac{\rho_\alpha}{\rho_w} y$

- H_{oc} = partition coefficient of sorption based on organic carbon
 \dot{H}_{ow} = octanol water partition coefficient
 $H_{\alpha\beta}^i$ = partition coefficient of i species between α and β phase = $\frac{w_{\alpha}^i}{w_{\beta}^i}$
 id = index for direction (1 = x, 2 = y, 3 = z)
 \vec{j} = unit vector in y direction
 $[J]_{ig}$ = Jacobian matrix at ig Gaussian point
 k_b = proportional coefficient in third-type boundary conditions
 k_m = coefficient of maximum rate of utilization in Monod equation
 K_s = half velocity coefficient in Monod equation
 k_{α}^i = reaction rate constant of i species in α phase
 \underline{k} = intrinsic permeability tensor
 k_{ra} = relative permeability of α phase
 \underline{K}_{α} = fluid conductivity of α phase = $\underline{k} k_{ra} \frac{\rho_w g}{\mu_{\alpha}}$
 M_{α} = mass of α phase
 m_{α}^i = mass of i species in α phase over unit volume
 M_{α}^i = mass of i species in α phase
 n = time step size
 nel = total number of elements
 nd = dimension of the problem
 nn = total number of nodes in each element
 na = total number of phases excluding soil phase
 \vec{n} = unit normal vector
 N_j = basis function at node j
 $N_{\xi,i,ig}$ = local basis function in ξ direction at i node and ig Gaussian point

- $N_{\eta,i,ig}$ = local basis function in η direction at i node and ig Gaussian point
 $N_{\zeta,i,ig}$ = local basis function in ζ direction at i node and ig Gaussian point
 $N_{i,ig}$ = basis function at i node and ig Gaussian point = $N_{\xi,i,ig}N_{\eta,i,ig}N_{\zeta,i,ig}$
 P_{α} = pressure of α phase
 P_{vp}^o = vapor pressure of trichloroethylene in air phase
 P_e = Peclet Number = $\frac{V\Delta x}{D}$
 \bar{q} = constant total flow in Buckley-Leverett problem
 S_f = substrate concentration in Monod equation
 S_{α} = saturation of α phase = $\frac{U_{\alpha}}{U_v} = \frac{U_{\alpha}}{\phi U} = \frac{\theta_{\alpha}}{\phi}$
 S_{α}^e = effective saturation of α phase = $\frac{S_{\alpha} - S_{ra}}{S_{sa} - S_{ra}}$
 S_{ra} = residual saturation of α phase
 S_{sa} = saturated saturation of α phase
 $S_{\alpha\beta}$ = specific storativity of porous matrix and α phase by β pressure head
 S_w^i = solubility of i species in water phase
 U = schematic elementary volume
 U_{α} = volume of α phase
 U_v = void volume
 V_{α} = velocity of α phase
 V_{α}^i = microscopic velocity of i species in α phase
 $V_{\xi 1}$ = local velocity in ξ direction at node 1
 $V_{\eta 1}$ = local velocity in η direction at node 1
 $V_{\zeta 1}$ = local velocity in ζ direction at node 1
 w_{α}^i = mass fraction of i species in α phase = $\frac{M_{\alpha}^i}{M_{\alpha}}$
 W_i = weighting function at node i

X = concentration of active bacteria in Monod equation

GREEK LETTERS

- α = index for phases (water, oil, air, solid) pressure head
- α_{ow} = constant in saturation equation between oil and water capillary head
- α_{ao} = constant in saturation equation between air and oil capillary head
- $\alpha_{\alpha,l}^i$ = longitudinal dispersivity of i species in α phase
- $\alpha_{\alpha,t}^i$ = transverse dispersivity of i species in α phase
- α_i = weighting factor in ξ direction
- α_0 = weighting factor in modified Hermitian Basis Function for node 1
- α_1 = weighting factor in modified Hermitian Basis Function for node 2
- α_β = compressibility of α phase by β pressure head
- β_i = weighting factor in η direction
- γ_i = weighting factor in ζ direction
- ε = weighting factor for time domain in generalized FDM
- ε_a = absolute convergence error
- ε_r = relative convergence error
- θ_α = volumetric fraction of α phase = $\frac{U_\alpha}{U}$
- μ_α = dynamic viscosity of α phase
- μ_{ra} = ratio between viscosity of α phase and water phase = $\frac{\mu_\alpha}{\mu_w}$
- ρ_α = density of α phase
- ρ_{ra} = density ratio between α and water phase = $\frac{\rho_\alpha}{\rho_w}$
- ϕ = porosity of porous media = $\frac{U_v}{U}$
- ϕ_β = compressibility of porous matrix by β pressure head

Appendix B

MACROSCOPIC VOLUME AVERAGING TECHNIQUE

Considering the conservation rule of some unit property in the α phase

$e_\alpha = \frac{E_\alpha}{U}$, the microscopic balance equation becomes as follows :

$$\frac{\partial e_x}{\partial t} = -\nabla(e_\alpha \vec{V}^{e_\alpha}) + G^{e_\alpha} \quad (B-1)$$

$$\frac{\partial e_x}{\partial t} = -\nabla(e_\alpha \vec{V}_\alpha + e_\alpha \vec{V}^{e_\alpha} - e_\alpha \vec{V}_\alpha) + G^{e_\alpha} \quad (B-2)$$

We can change the above microscopic equation into the macroscopic one over the entire phase domain by applying the volume averaging technique, so the above equation becomes :

$$\begin{aligned} \frac{\partial(\theta_\alpha \bar{e}_x)}{\partial t} = & -\nabla \theta_\alpha (\bar{e}_\alpha \bar{V}_\alpha + \bar{e}_\alpha \bar{V}_\alpha + \bar{J}^*) - \frac{1}{U_0} \int_{(4)} e_\alpha (\bar{V}_\alpha - \vec{u}_x) \cdot \vec{n}_\alpha d\gamma \\ & + \theta_\alpha \bar{G}^{e_\alpha} \quad (B-3) \end{aligned}$$

Where bar indicates the averaged value.

(1) = advective flux

(2) = mechanical dispersive flux = $-\bar{D}_s \nabla e_\alpha$

(3) = molecular diffusion = $-\bar{D}_m \nabla e_\alpha$

(4) = transfer between phases = f_α

(5) = production rate

$$(2) + (3) = \text{dispersive flux} = -D_{\alpha}^i \nabla e_{\alpha}$$

To derive the governing equation of composite multiphase flow,

1. Replace the property e_{α} by the density of species i in phase α ρ_{α}^i ;
2. Use the concept of mass fraction, $w_{\alpha}^i = \frac{Mass_{\alpha}^i}{Mass_{\alpha}} = \frac{\rho_{\alpha}^i}{\rho_{\alpha}}$, so $\rho_{\alpha}^i = \rho_{\alpha} w_{\alpha}^i$;
3. Use Darcy's law, $\vec{V}_{\alpha} = \rho_{\alpha} k_{r\alpha} \frac{k_{r\alpha}}{\theta_{\alpha} \mu_{\alpha}} (\nabla P_{\alpha} - \rho_{\alpha} \vec{g})$;
4. Assemble all phases for species i .

$$\sum \left[\frac{\partial(\theta_{\alpha} \rho_{\alpha} w_{\alpha}^i)}{\partial t} - \nabla \left(\rho_{\alpha} k_{r\alpha} \frac{k_{r\alpha}}{\mu_{\alpha}} (\nabla P_{\alpha} - \rho_{\alpha} \vec{g}) w_{\alpha}^i \right) - \nabla (\theta_{\alpha} \rho_{\alpha} D_{\alpha}^i \nabla w_{\alpha}^i) \right] + \frac{\partial(\theta_s \rho_s w_s^i)}{\partial t} = \sum [G_{\alpha}^i] \quad (B-4)$$

Appendix C
LIST OF PROGRAM

c generalized finite difference method in time
c modified picard iteration for nonlinearity
c element wise evaluation of parameters
c decoupling of system of equations
c
c

c Structure

- c 1. main : main program
- c 2. input : input data (input,binput)
- c 3. parameter
 - c a. partit : partition coefficients
 - c b. disper : dispersive coefficients
 - c c. densit : density of each phase
 - c d. porsit : porosity of spatial domain
 - c e. dsat : saturation and relative permeability
 - c f. veloc : velocity of each phase
 - c g. coeff : coefficient of mass transport eq'n
 - c h. sourit : point source or sink
- c 4. fem : finite element scheme (femh,femc)
 - c a. asem : assembling element matrices (asemh,asemc)
 - c b. elemen : evaluation of element matrices
 - c c. basis : basis functions (basis1,basis2,basis3)
 - c d. weigh : weighting functions (weigh1,weigh2,weigh3)
 - c e. boun : boundary conditions
(bound1,bounc1,boun12,boun22,boun32)
 - c f. solve : asymmetric band matrix solver
- c 5. output : print out result (outp,output)


```

&rk(npm,nam,ndm,ndm)
dimension rn(npm)
dimension wm(npm,nsm,4),oldwm(npm,nsm,4),c(npm),oldc(npm)
dimension rho(npm,4),rhor(nam)
dimension source(npm,nsm,nam)
c
common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S      ipx,ipy,npx,ipy,ndmp,nsmp,namp,nnmp
common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
S      pltime,dpltim,erc,wold,wnew
common /space/ x,y,z
common /bas/ bn,bnd,bndr,bnr,dnr,dji,det,wdn,dwdn
common /points/ gp,cn
c
common /dens/ rho,rhor
common /part/ wm,oldwm
common /part2/ c,oldc
common /poro/ rn
common /sour/ source
c
c
call input(oldh,btime)
call porsit
call densit
call disper
call sourit
c
c... initialize pressure head ...
c
do 10 ia = 1,namp
do 10 ip = 1,npmax
h(ip,ia) = oldh(ip,ia)
10 continue
c
do 15 ip = 1,npmax
oldc(ip) = 0.d0
c(ip) = 0.d0
15 continue
c
do 17 is = 1,nsmp
do 17 ia = 1,namp
do 17 ip = 1,npmax
oldwm(ip,is,ia) = 0.d0
wm(ip,is,ia) = 0.d0
17 continue
c

```



```

c... evaluate the basis function before time advanced ...
c
    if(ndmp.eq.1) call basis1
    if(ndmp.eq.2) call basis2
    if(ndmp.eq.3) call basis3
c
1    time = time + dt
c
c... debug ..
c
    if(time.le.dt) then
        write(3,*)
        write(3,*) 'basis function'
        write(3,1000) ((bn(in,jn),jn = 1,nnmp),in = 1,nnmp)
1000    format(4(/4e15.5))
        write(3,*)
        write(3,*) 'derivative of basis function'
        do 20 ic = 1,2
        do 20 id = 1,ndmp
            write(3,*) 'ic,id = ',ic,id
            write(3,1000) ((dnr(ic,id,in,jn),jn = 1,nnmp),in = 1,nnmp)
20    continue
        endif
c
    if(time.gt.tmax) goto 2
c
c... change of boundary conditions ...
c
    if(time.ge.btime) call binput(btime)
c
c... finite element scheme ...
c
    call femh(h,oldh,sat,oldsat)
c
    call femc(h,oldh,sat,oldsat)
c
c... print out head and saturation at each printing step ...
c
    if(time.ge.ptime) then
        ptime = ptime + dptime
        call output(h,sat)
    endif
c
c... reset primal variables ...
c
    do 40 ia = 1,namp

```

```
do 40 ip = 1,npmax
  oldh(ip,ia) = h(ip,ia)
40 continue
c
do 45 ip = 1,npmax
  oldc(ip) = c(ip)
45 continue
c
do 50 is = 1,nsmp
do 50 ia = 1,namp
do 50 ip = 1,npmax
  oldwm(ip,is,ia) = wm(ip,is,ia)
50 continue
c
goto 1
c
c
2 stop
end
```

```

c
c
c
  subroutine input(hi,btime)
c
c
c... this subroutine inputs and prints out data...
c
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension hi(npm,nam),x(npm),y(npm),z(npm),gc(3),ijk(npm,nnm)
c
  dimension
  &nbe1(nam),nbe2(nam),nbe3(nam),noeb1(npm,nam),noeb3(npm,nam),
  &noeb2(npm,nam,nnm),hbe1(npm,nam),hbe3(npm,nam),
  &hbe2(npm,nam,nnm),xkbe3(npm,nam)
c
  dimension gp(ndm,nnm),cn(ndm,nnm)
c
  dimension px(41,41),py(41,41)
c
  dimension rn(npm),rn1(npm,nam)
c
  character*80 gtitle,potitl,
  &title,ctitl1,ctitl1,htitl,ttitl,btitle,b1titl,b2titl,b3titl
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
  S ipx,ipy,npx,npj,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
  S pltime,dpltim,erc,wold,wnew
  common /space/ x,y,z
  common /points/ gp,cn
  common /axis/ px,py
c
  common /ib/ indb1,indb2,indb3
  common /ib1/ nbe1,noeb1
  common /b1/ penalt,hbe1
  common /ib2/ nbe2,noeb2
  common /b2/ hbe2
  common /ib3/ nbe3,noeb3
  common /b3/ hbe3,xkbe3
c
  common /poro/ rn
c
c

```

```

    wold = 0.5d0
    wnew = 0.5d0
    axi = 0.6
    acta = 0.6
c
    read(1,1000) title
    write(2,1100) title
    write(3,1100) title
    write(7,1100) title
    read(1,*) ndmp,nsmp,namp,nnmp,nelmax,npmax
    write(2,1200) ndmp,nsmp,namp,nnmp,nelmax,npmax
    write(3,1200) ndmp,nsmp,namp,nnmp,nelmax,npmax
    write(7,1200) ndmp,nsmp,namp,nnmp,nelmax,npmax
c
    read(1,1000) ttitle
    read(1,*) fmax,time,dt,ep,ptime,dptime,pltime,dpltim,
S      ipx,ipy,npx,nty,npz,rgt
    write(3,1900) tmax,time,dt,ep,ptime,dptime,pltime,dpltim,
S      axi,acta,ipx,ipy,npx,nty,npz
c
c31  npp = npmax-1
     npp = npmax-1
c
c... coordinates of gaussian and corner points ...
c
    read(1,1000) gtitle
    write(3,1100) gtitle
    read(1,*) ((gp(id,in),in = 1,nnmp),id = 1,ndmp)
    write(3,2100) ((gp(id,in),in = 1,nnmp),id = 1,ndmp)
    read(1,*) ((cn(id,in),in = 1,nnmp),id = 1,ndmp)
    write(3,2100) ((cn(id,in),in = 1,nnmp),id = 1,ndmp)
c
c... global coordinates of the nodes ...
c
    npz = 1
    ip = 0
    read(1,1000) ctitl1
    write(3,1100) ctitl1
    read(1,*) index,fx,fy,fz
c   write(3,*) index,fx,fy,fz
    if(index.gt.0) then
        do 20 iz = 1,npz
        do 20 ix = 1,npx
        do 20 iy = 1,nty
            ip = ip + 1
            x(ip) = fx*(ix-1)

```

```

        y(ip) = fy*(iy-1)
20      z(ip) = fz*(iz-1)
        else
          do 25 ip = 1,npmax
            read(1,*) ith,(gc(i),i = 1,ndmp)
            x(ip) = gc(1)*fx
            y(ip) = gc(2)*fy
25      z(ip) = gc(3)*fz
          endif
        write(3,1300) (ip,x(ip),y(ip),z(ip),ip = 1,npmax)
c
c... set coordintes of the axis for print out ...
c
        k = 0
        do 30 i = 1,npx
          do 30 j = 1,ncpy
            k = k + 1
            px(i,j) = x(k)
30      py(i,j) = y(k)
c
c... element connectivity of each element ...
c
        read(1,1000) etitl1
        write(3,1100) etitl1
        read(1,*) (j,(ijk(ie,i),i = 1,nnmp),ie = 1,nelmax)
        write(3,1400) (ip,(ijk(ie,i),i = 1,nnmp),ie = 1,nelmax)
c
c... find the half band width of the global matrix
c          from the element connectivities ...
c
        ihalfb = 0
        do 40 nel = 1,nelmax
          do 50 i = 1,nnmp
            ijki = ijk(nel,i)
            do 50 j = 1,nnmp
              ijkj = ijk(nel,j)
              ijkij = ijkj-ijki
              if(ijkij.gt.ihalfb) ihalfb = ijkij
50          continue
40      continue
        iband = ihalfb*2 + 1
        write(3,*)
        write(3,*) 'ihalfb,iband = ',ihalfb,iband
c
c... pressure head of each phase ...
c
        heafac = 0.d0

```

```

        read(1,1000) htitl
        write(3,1100) htitl
        read(1,*) indexc,fpw,fpo
        if(indexc.gt.0) then
            do 60 ip = 1,npmax
                hi(ip,1) = fpw
60         hi(ip,2) = fpo
            else
                read(1,*) (ii,hi(ip,1),hi(ip,2),ip = 1,npmax)
                do 70 ip = 1,npmax
                    hi(ip,1) = fpw*hi(ip,1)
c         write(6,*) hi(ip,1)
70         hi(ip,2) = fpo*(hi(ip,2) + heafac)
            endif
        c
        write(3,*) 'initial water pressure head'
        call outp(hi,1,3)
        write(3,*) 'initial oil pressure head'
        call outp(hi,2,3)
        c
c... porosity of geologic domain ...
        c
        read(1,1000) potitl
        c write(3,1100) potitl
        read(1,*) indexn,facn
        if(indexn.gt.0) then
            do 72 ip = 1,npmax
                rn(ip) = facn
72         continue
            else
                read(1,*) (ii,rn(ip),ip = 1,npmax)
                do 74 ip = 1,npmax
                    rn(ip) = facn*rn(ip)
74         continue
            endif
        c
        write(3,*) 'porosity of domain'
        do 76 ip = 1,npmax
            rn1(ip,1) = rn(ip)
76         continue
        call outp(rn1,1,3)
        c
c... boundary conditions ...
        c
        read(1,1000) btitle
        write(3,1100) btitle
        read(1,*) indb1,indb2,indb3,btime

```

```

        write(3,2000) indb1,indb2,indb3,btime
c
c... 1st type boundary conditions ...
c
    if(indb1.eq.1) then
    do 80 iwo = 1,2
    read(1,1000) b1titl
    write(3,1100) b1titl
    read(1,*) nbe1(iwo),penalt
    write(3,*) 'nbe1,penalt = ',nbe1(iwo),penalt
    nb = nbe1(iwo)
    read(1,*) (noeb1(ip,iwo),hbe1(ip,iwo),ip = 1,nb)
80    write(3,1650) (noeb1(ip,iwo),hbe1(ip,iwo),ip = 1,nb)
        continue
    endif
c
c... 2nd type boundary conditions ...
c
    if(indb2.eq.1) then
    do 90 iwo = 1,2
    read(1,1000) b2titl
    write(3,1100) b2titl
    read(1,*) nbe2(iwo)
    nb = nbe2(iwo)
    write(3,*) 'nbe2 = ',nb
    read(1,*) ((noeb2(ip,iwo,ib),
S        hbe2(ip,iwo,ib),ib = 1,nnmp/2),ip = 1,nb)
    write(3,1800) ((noeb2(ip,iwo,ib),
S        hbe2(ip,iwo,ib),ib = 1,nnmp/2),ip = 1,nb)
90    continue
        endif
c
c... 3rd type boundary conditions ...
c
    if(indb3.eq.1) then
    do 100 k = 1,2
    read(1,1000) b3titl
    write(3,1100) b3titl
    read(1,*) nbe3(k)
    nb = nbe3(k)
    write(3,*) 'nbe3 = ',nb
    read(1,*) (noeb3(ip,k),xkbe3(ip,k),hbe3(ip,k),ip = 1,nb)
    write(3,1600) (noeb3(ip,iwo),xkbe3(ip,iwo),hbe3(ip,iwo),ip = 1,nb)
100    continue
        endif
c
1000 format(a80)

```

```

1100 format(/a80/)
1200 format(/t2,'ndmp = ',i2,t10,'nsmp = ',i2,t18,'namp = ',i2,
    &      t26,'nnmp = ',i2,t34,'nelmax = ',i5,t50,'npmax = ',i5)
1300 format(i5,3f15.5)
1400 format(5i5)
1600 format(/3(i5,2f7.4,3x))
1650 format(/5(i5,f7.4,2x))
1800 format(/6(i5,f7.4))
1900 format(/t2,'tmax = ',f10.4,t20,'time = ',f10.4,t40,'dt = ',f10.4,
    +      /t2,'cp = ',f10.4,t20,'ptime = ',f10.4,t40,'dptime = ',f10.4
    +      /t2,'pltime = ',f10.4,t20,'dpltim = ',f10.4
    +      /t2,'axi = ',f10.4,t20,'aeta = ',f10.4
    +      /t2,'ipx = ',i5,t15,'ipy = ',i5,t30,'npx = ',i5,t40,'npy = ',i5,
    &      t50,'npz = ',i5)
2000 format(/t2,'indb1 = ',i5,t15,'indb2 = ',i5,t30,'indb3 = ',i5,
    &      t45,'btime = ',f10.4/)
2100 format(/t2,4f10.6)
c
c
    return
end

```



```

c
c
c
  subroutine binput(btime)
c
c
c... this subroutine input changed boundary conditions ...
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension
    &nbe1(nam),nbe2(nam),nbe3(nam),noeb1(npm,nam),noeb3(npm,nam),
    &noeb2(npm,nam,nnm),hbe1(npm,nam),hbe3(npm,nam),
    &hbe2(npm,nam,nnm),xkbe3(npm,nam)
c
  character*80 btitle,b1titl,b2titl,b3titl
c
  common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
  S      ptime,dpltim,erc,wold,wnew
c
  common /ib/ indb1,indb2,indb3
  common /ib1/ nbe1,noeb1
  common /b1/ penalt,hbe1
  common /ib2/ nbe2,noeb2
  common /b2/ hbe2
  common /ib3/ nbe3,noeb3
  common /b3/ hbe3,xkbe3
c
  read(1,100) btitle
  read(1,*) indb1,indb2,indb3,btime,dt
  write(3,150) btitle
  write(3,1000) indb1,indb2,indb3
c
  if(indb1.ne.1) goto 250
  do 240 iwo = 1,2
    read(1,100) b1titl
    read(1,*) nbe1(iwo),penalt
    nb = nbe1(iwo)
    read(1,*) (noeb1(ip,iwo),hbe1(ip,iwo),ip = 1,nb)
    write(3,100) b1titl
    write(3,*) 'nbe1,penalt = ',nbe1(iwo),penalt
    write(3,650) (noeb1(ip,iwo),hbe1(ip,iwo),ip = 1,nb)
240  continue
c
250  if(indb2.ne.1) goto 270
    do 260 iwo = 1,2

```

```

    read(1,100) b2titl
    read(1,*) nbe2(iwo)
    nb = nbe2(iwo)
    read(1,*) ((noeb2(ip,iwo,ib),ib = 1,nnm/2),
& (hbc2(ip,iwo,ib),ib = 1,nnm/2),ip = 1,nb)
    write(3,100) b2titl
    write(3,*) 'nbe2 = ',nb
    write(3,800) ((noeb2(ip,iwo,ib),ib = 1,nnm/2),
& (hbc2(ip,iwo,ib),ib = 1,nnm/2),ip = 1,nb)
260  continue
c
270  if(indb3.nc.1) goto 290
    do 280 k = 1,2
        read(1,100) b3titl
        read(1,*) nbe3(k)
        nb = nbe3(k)
        read(1,*) (noeb3(ip,k),xkbe3(ip,k),hbc3(ip,k),ip = 1,nb)
        write(3,100) b3titl
        write(3,*) 'nbe3 = ',nb
        write(3,600) (noeb3(ip,iwo),xkbe3(ip,iwo),hbc3(ip,iwo),ip = 1,nb)
280  continue
c
100  format(a80)
150  format(/a80/)
600  format(/5(i5,2e10.3))
650  format(/5(i5,e15.5))
800  format(/4(2i5,2e10.3))
1000 format(/t10,'indb1 = ',i5,t30,'indb2 = ',i5,t50,'indb3 = ',i5/)
c
290  return
    end

```

```

c
c
c
  subroutine partit(sat)
c
c... this subroutine computes the instant equilibrium relations between
c every species of the four phases ...
c
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension wm(npm,nsm,4),oldwm(npm,nsm,4),ijk(npm,nnm),rhor(nam)
  dimension sat(npm,nam),c(npm),oldc(npm),rn(npm),rho(npm,4)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S      ipx,ipy,npx,ppy,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,ep,axi,acta,rgr,ptime,dptime,
S      pltime,dpltim,erc,wold,wnew
  common /part/ wm,oldwm
  common /part2/ c,oldc
  common /poro/ rn
  common /dens/ rho,rhor
c
  sro = 0.30
  hwoo = 1.1012e-3
c
c
  nam1 = namp + 1
c
  do 10 ip = 1,npmax
    wm(ip,1,1) = 1.d0
    wm(ip,2,2) = 1.d0
    oldwm(ip,1,1) = 1.d0
    oldwm(ip,2,2) = 1.d0
    if(sat(ip,2).gt.sro) then
      wm(ip,2,1) = wm(ip,2,2)*hwoo
      wm(ip,1,1) = 1.d0-wm(ip,2,1)
      oldwm(ip,2,1) = wm(ip,2,1)
      oldwm(ip,1,1) = wm(ip,1,1)
      c(ip) = wm(ip,2,1)
c1   c(ip) = wm(ip,2,1)*rn(ip)*rho(ip,1)*sat(ip,1)
    endif
c
c   wm(ip,2,3) = wm(ip,2,1)*hgwo
c   wm(ip,2,3) = 0.d0
c   rkd = foc*dexp(-0.56*dlog(wm(ip,2,1)) + 3.64)

```

```

c   wm(ip,2,4) = wm(ip,2,1)*rkd
cw  write(3,100) wm(ip,1,1),wm(ip,2,1),wm(ip,1,2),wm(ip,2,2),
cw  S   wm(ip,3,2),wm(ip,4,2)
10  continue
c
c
c.. debug ..
c
   if(time.le.dt) then
   write(3,*)
   write(3,*) 'mass fraction'
   do 30 ip = 1,3
   write(3,*) 'ip = ',ip
   write(3,1000) ((wm(ip,is,ia),ia = 1,nam1),is = 1,namp)
1000 format(8f8.5)
30  continue
   endif
c
   return
   end

```

```

c
c
c
c
    subroutine disper
c
c... this subroutine computes the hydrodynamic dispersion coefficient
c of every species in three phases. ...
c
c
    implicit real*8 (a-h,o-z)
    parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
    dimension ijk(npm,nnm),rd(npm,nsm,nam,ndm,ndm)
c
    common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S      ipx,ipy,npx,nty,ndmp,nsmp,namp,nnmp
    common /set1/ tmax,time,dt,ep,axi,acta,rgr,ptime,dptime,
S      pltime,dpltim,erc,wold,wnew
c
    common /disp/ rd
c
    rdfac = 1.00d0
c
    do 10 is = 1,nsmp
      do 10 ia = 1,namp
        do 10 id = 1,ndmp
          do 10 jd = 1,ndmp
            do 10 ip = 1,npmax
              rd(ip,is,ia,id,jd) = 0.d0
10      continue
c
          do 20 id = 1,ndmp
            do 20 is = 1,nsmp
              do 20 ip = 1,npmax
                rd(ip,is,1,id,id) = 0.5574*rdfac
                rd(ip,is,2,id,id) = 0.2787*rdfac
                rd(ip,is,3,id,id) = 1.0034*rdfac
20      continue
c
c... debug ...
c
    write(3,*)
    write(3,*) 'dispersion coefficients of water, oil, gas phase'
    write(3,*) 'rdfac = ',rdfac
    write(3,1000) rd(1,1,1,1,1),rd(1,2,2,1,1),rd(1,2,3,1,1)
c

```

```
1000 format(/3e15.5)
```

```
c
```

```
  return  
end
```

```

c
c
c
  subroutine densit
c
c... this subroutine computes the densities of four phases from
c the thermodynamic relations.
c density is depending on pressure, compressibility and
c temperature....
c
c
c implicit real*8 (a-h,o-z)
c parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c dimension ijk(npm,nnm),rho(npm,4),rhor(nam),h(npm,nam)
c dimension drh(npm,4,nam)
c
c common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S ipx,ipy,npx,ppy,ndmp,nsmp,namp,nnmp
c common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
S pltime,dpltim,erc,wold,wnew
c
c common /dens/ rho,rhor
c1 common /dens/ rho,rhor,drh
c
c do 10 ia = 1,namp
c   rhor(ia) = 0.d0
c   do 10 ip = 1,npmax
c     rho(ip,ia) = 0.d0
c     do 10 ja = 1,namo
c       drh(ip,ia,ja) = 0.d0
10 continue
c
c   rhor(1) = 1.d0
c   rhor(2) = 1.2d0
c
c31   rhor(2) = 1.2d0
c4   rhor(2) = 1.d0
c
c do 20 ip = 1,npmax
c   rho(ip,1) = 1000.d0
c   rho(ip,2) = 1200.d0
c   rho(ip,3) = 1.205d0
c   rho(ip,4) = 1200.d0
20 continue
c
c
c

```

```
c... debug ...
c
  if(time.le.dt) then
    write(3,*)
    write(3,*) 'density of water phase'
    call outp(rho,1,3)
    write(3,*) 'density of oil phase'
    call outp(rho,2,3)
    write(3,*) 'density of air phase'
    call outp(rho,3,3)
  endif
c
c
  return
end
```



```

c
c
c
  subroutine porsit
c
c... this subroutine computes the porosity variation
c by pressure head ...
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension rn(npm),rn1(npm,nam),ijk(npm,nnm)
  dimension dnh(npm,nam)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S      ipx,ipy,npx,nty,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
S      ptime,dpltim,erc,wold,wnew
  common /poro/ rn
c
  common /poro/ rn,dnh
c
  do 5 ia = 1,namp
    do 5 ip = 1,npmax
      dnh(ip,ia) = 0.d0
5    continue
c
c... debug ...
c
  if(time.eq.-1) then
  write(3,*) 'porosity of domain'
  do 76 ip = 1,npmax
    rn1(ip,1) = rn(ip)
76  continue
  call outp(rn1,1,3)
  endif
c
  return
  end

```

```

c
c
c
  subroutine dsat(h,oldh,sat,oldsat,sh,rk)
c
c.. this subroutine compute the saturations and saturation derivatives
c of 3 phases from the capillary head
c relative permeability is computed from the saturation values...
c
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension
    &h(npm,nam),oldh(npm,nam),
    &how(npm),hao(npm),haw(npm),hocr(npm),satt(npm),
    &x(npm),y(npm),z(npm),ijk(npm,nnm),rkr(npm,nam)
c
  dimension
    &sat(npm,nam),oldsat(npm,nam),sh(npm,nam,nam),
    &rk(npm,nam,ndm,ndm)
  dimension rn(npm),dnh(npm,nam)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
  S      ipx,ipy,npx,npy,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
  S      pltime,dpltim,erc,wold,wnew
  common /space/ x,y,z
c
  common /poro/ rn
c
c... initialize saturation derivatives wrt capillary head ...
c
  do 5 ia = 1,namp
    do 5 ja = 1,namp
      do 5 ip = 1,npm
5        sh(ip,ia,ja) = 0.d0
c
c... initialize fluid conductivity ...
c
  do 6 ia = 1,namp
    do 6 id = 1,ndmp
      do 6 jd = 1,ndmp
        do 6 ip = 1,npm
6          rk(ip,ia,id,jd) = 0.d0
c
c

```

```

c... required coefficients ...
c
c  write(3,*)
c  write(3,*) 'water pressure'
c  call outp(h,1,3)
c  write(3,*)
c  write(3,*) 'oil pressure'
c  call outp(h,2,3)
c
dthfac = 1e-5
atm = 0.d0
rksw = 0.5
rmuo = 0.5
alaw = 5.2d0
alao = 11.d0
alow = 9.9d0
xn = 1.8*1.1
bcow = alow/alaw
beao = alao/alaw
xm = 1.d0-1.d0/xn
ssw = 0.80
srw = 0.20
sso = 0.80
sro = 0.20
ssa = 0.60
sra = 0.00
sst = 1.00
srt = 0.40
c
c
c  ***** saturation and fluid conductivity *****
c
do 15 i = 1,npmax
  how(i) = h(i,2)-h(i,1)
  hao(i) = atm-h(i,2)
  haw(i) = atm-h(i,1)
  hocr(i) = bcow/(bcow + beao)*h(i,1)
c
c... in the case of two phase saturation ...
c
  if(h(i,2).le.hocr(i)) goto 20
  goto 30
c
20  if(haw(i).le.0.d0) then
    sat(i,1) = ssw
  else
    sat(i,1) = (ssw-srw)*(1. + (alaw*haw(i))**xn)**(-xm) + srw

```

```

endif
sat(i,2) = sro
satt(i) = sat(i,1) + sat(i,2)
se = (sat(i,1)-srw)/(1-srw)
sh(i,1,1) = (xn-1.)*se**(1/xm)*(1.-(se)**(1/xm))**xm*
S      alaw*(ssw-srw)*rn(i)
sh(i,1,2) = -dthfac
sh(i,2,1) = dthfac
sh(i,2,2) = -dthfac
oxm = 1./xm
dse = ((1-se**oxm))**xm
rkr(i,1) = dsqrt(se)*((1-dse))**2
rkr(i,2) = dthfac
croy   rkr(i,1) = (sat(i,1))**2
croÿ   rkr(i,2) = 1-rkr(i,1)
      goto 40
c
c three phase saturation
c
30   if(hao(i).le.0.d0) then
      satt(i) = sst
    else
      satt(i) = (sst-srt)*(1. + (alao*hao(i))**xn)**(-xm) + srt
    endif
c   write(3,*) hao(i),how(i)
   if(how(i).le.0.d0) then
     sat(i,1) = ssw
   else
     sat(i,1) = (ssw-srw)*(1. + (alow*how(i))**xn)**(-xm) + srw
   endif
   sat(i,2) = satt(i)-sat(i,1)
   if(sat(i,2).lt.sro) sat(i,2) = sro
   se = (sat(i,1)-srw)/(1.d0-srw)
   sh(i,1,1) = (xn-1.)*se**(1/xm)*(1.-(se)**(1/xm))**xm*
S     alow*(ssw-srw)*rn(i)
sh(i,1,2) = -sh(i,1,1)
oxm = 1./xm
dse = ((1-se**oxm))**xm
rkr(i,1) = dsqrt(se)*((1-dse))**2
croy   rkr(i,1) = (sat(i,1))**2
set = (satt(i)-srt)/(1.d0-srt)
sh(i,2,2) = (xn-1.)*set**(1/xm)*(1.-(set)**(1/xm))**xm*
S     alao*(sst-srt)*rn(i) + sh(i,1,1)
sh(i,2,1) = sh(i,1,2)
dset = dabs(set-se)
dsett = ((1-set**oxm))**xm

```

```

      rkr(i,2) = dsqrt(dset)*((dsc-dsett)**2
croy  rkr(i,2) = 1-rkr(i,1)
c
40   sat(i,3) = 1.d0-sat(i,1)-sat(i,2)
c
c fluid conductivity at saturation and viscosity of oil/water
c
c4   rk(i,1,1,1) = rkr(i,1)*rksw
c4   rk(i,2,1,1) = rkr(i,2)*rksw
c
c31  rk(i,1,1,1) = rkr(i,1)*rksw*2.00
c31  rk(i,2,1,1) = rkr(i,2)*rksw/rmuo*2.00
c
c32  rk(i,1,2,2) = rkr(i,1)*rksw*2.00
c32  rk(i,2,2,2) = rkr(i,2)*rksw/rmuo*2.00
c
c2   rk(i,1,1,1) = rkr(i,1)*rksw*0.10
c    rk(i,1,2,2) = rkr(i,1)*rksw*2.0
c    rk(i,2,1,1) = rkr(i,2)*rksw/rmuo*0.10
c2   rk(i,2,2,2) = rkr(i,2)*rksw/rmuo*2.0
c
    if(rkr(i,1).gt.1.d0) rkr(i,1) = 1.d0
    if(rkr(i,2).gt.1.d0) rkr(i,2) = 1.d0
c
    rk(i,1,1,1) = rkr(i,1)*rksw
    rk(i,1,2,2) = rkr(i,1)*rksw
    rk(i,2,1,1) = rkr(i,2)*rksw/rmuo
    rk(i,2,2,2) = rkr(i,2)*rksw/rmuo
c
c
15  continue
c  endif
c
c... debug ...
c
c  if(time.le.dt) then
c  do 200 i = 1,10
c  write(3,*) 'sat,ip =',i
c  write(3,1000) (sat(i,ia),ia = 1,namp)
c  write(3,*) 'sh,ip =',i
c  write(3,1000) ((sh(i,ia,ja),ja = 1,namp),ia = 1,namp)
c 200 continue
c  endif
c1000 format(3e15.5)
c
c

```

c4100 continue

c

return
end

```

c
c
c
  subroutine veloc(h,rk,sat,vel)
c
c... this subroutine computes local and global velocity of each phase ...
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension
&vel(npm,nam,ndm,nnm),velp(npm,nam),weifp(npm,nam),
&ijk(npm,nnm),h(npm,nam),
&rk(npm,nam,ndm,ndm,nnm)
c
  dimension
&bn(nnm,nnm),bnd(ndm,nnm,nnm),bndr(ndm,nnm,nnm),
&bnr(ndm,nnm,nnm),
&dnr(npm,ndm,nnm,nnm),wdn(ndm,nnm,nnm),dwdn(ndm,nnm,nnm),
&dj(nam,nam),dji(npm,nam,nam,nnm),det(npm,nnm),
&w(npm,nam,nnm,nnm),dwr(npm,nam,ndm,nnm,nnm),
&weif(npm,nam,ndm,nnm)
c
  dimension
&sat(npm,nam),oldsat(npm,nam),dsh(npm,nam,nam),
&rk(npm,nam,ndm,ndm)
  dimension rn(npm),dnh(npm,nam)
  dimension rho(npm,4),rhor(nam),drh(npm,4,nam)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S      ipx,ipy,npx,npy,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
S      pltime,dpltim,erc,wold,wnew
c
  common /bas/ bn,bnd,bndr,bnr,dnr,dji,det,wdn,dwdn
  common /wei/ w,dwr,weif
c
  common /poro/ rn
  common /dens/ rho,rhor
c
c
c
c... initialize global and local velocity ...
c
  do 10 ia = 1,namp
    do 10 id = 1,ndmp
      do 10 i = 1,nnmp

```

```

        do 10 ie = 1,nelmax
            vel(ie,ia,id,i) = 0.d0
            weif(ie,ia,id,i) = 0.d0
10    continue
c
c... element wise evaluation of fluid conductivity ...
c
        do 30 ia = 1,namp
            do 30 id = 1,ndmp
                do 30 jd = 1,ndmp
                    do 30 i = 1,nnmp
                        do 30 ie = 1,nelmax
30    rkg(ie,ia,id,jd,i) = 0.d0
c
        do 40 ia = 1,namp
            do 40 id = 1,ndmp
                do 40 jd = 1,ndmp
                    do 40 i = 1,nnmp
                        do 40 ig = 1,nnmp
                            do 40 ie = 1,nelmax
                                ijkig = ijk(ie,ig)
                                th = rn(ijkig)*sat(ijkig,ia)
                                rkg(ie,ia,id,jd,i) = rk(ijkig,ia,id,jd)/th*bn(i,ig)
                                &
                                    + rkg(ie,ia,id,jd,i)
c    write(3,*) 'kx = ',rkg(ie,ia,id,jd,i),rk(ijkig,ia,id,jd),ie
40    continue
c
c... global velocity ...
c
        do 50 ia = 1,namp
            do 50 in = 1,nnmp
                do 60 id = 1,ndmp
                    do 60 jd = 1,ndmp
                        do 60 ig = 1,nnmp
c    write(3,*) 'ia,in,id,jd,ig = ',ia,in,id,jd,ig
                            do 60 ie = 1,nelmax
                                ijkig = ijk(ie,ig)
                                vel(ie,ia,id,in) = vel(ie,ia,id,in)
                                &
                                    -rkg(ie,ia,id,jd,in)*dnr(ie,jd,ig,in)*h(ijkig,ia)
c    write(3,*) 'vel = ',ie,vel(ie,ia,id,in)
60    continue
50    continue
c
        do 65 ia = 1,namp
            do 65 in = 1,nnmp
c    write(3,*) 'ia,in = ',ia,in
            do 65 ie = 1,nelmax

```



```

        vel(ic,ia,2,in) = vel(ic,ia,2,in) - rkg(ic,ia,2,2,in) * rhor(ia)
c31  . vel(ic,ia,1,in) = vel(ic,ia,1,in) - rkg(ic,ia,1,1,in) * rhor(ia)
c32  . vel(ic,ia,2,in) = vel(ic,ia,2,in) - rkg(ic,ia,2,2,in) * rhor(ia)
c    write(3,*) 'vel = ', vel(ic,ia,1,in)
65   continue
c
c
c
c    ***** weighting factor *****
c
do 70 ia = 1, namp
do 70 ic = 1, nelmax
c
    dnom = dabs(vel(ic,ia,1,1)) + dabs(vel(ic,ia,1,2))
    if(dnom.eq.0.d0) then
        weif(ic,ia,1,1) = 0.d0
    else
        weif(ic,ia,1,1) = axi*(vel(ic,ia,1,1) + vel(ic,ia,1,2))/dnom
    endif
    weif(ic,ia,1,2) = weif(ic,ia,1,1)
c
    dnom = dabs(vel(ic,ia,1,3)) + dabs(vel(ic,ia,1,4))
    if(dnom.eq.0.d0) then
        weif(ic,ia,1,3) = 0.d0
    else
        weif(ic,ia,1,3) = axi*(vel(ic,ia,1,3) + vel(ic,ia,1,4))/dnom
    endif
    weif(ic,ia,1,4) = weif(ic,ia,1,3)
c
    dnom = dabs(vel(ic,ia,2,1)) + dabs(vel(ic,ia,2,4))
    if(dnom.eq.0.d0) then
        weif(ic,ia,2,1) = 0.d0
    else
        weif(ic,ia,2,1) = axi*(vel(ic,ia,2,1) + vel(ic,ia,2,4))/dnom
    endif
    weif(ic,ia,2,4) = weif(ic,ia,2,1)
c
    dnom = dabs(vel(ic,ia,2,2)) + dabs(vel(ic,ia,2,3))
    if(dnom.eq.0.d0) then
        weif(ic,ia,2,2) = 0.d0
    else
        weif(ic,ia,2,2) = axi*(vel(ic,ia,2,2) + vel(ic,ia,2,3))/dnom
    endif
    weif(ic,ia,2,3) = weif(ic,ia,2,2)
c
70   continue
c

```

```

c
c.., debug ...
c
  if(time.lc.-1) then
  write(3,*)
  do 220 ia = 1,namp
  do 220 id = 1,ndmp
  do 222 in = 1,nnmp
  write(3,*) 'ia,id,in = ',ia,id,in
  do 224 ie = 1,nelmax
  ip = ijk(ie,in)
  velp(ip,ia) = vel(ie,ia,id,in)
  weifp(ip,ia) = weif(ie,ia,id,in)
224  continue
  write(3,*) 'velocity'
  call outp(velp,ia,3)
  write(3,*) 'weighting factor'
  call outp(weifp,ia,3)
222  continue
220  continue
  endif
c
c
  return
  end

```

```

c
c
c
c   subroutine coeff(sat,vel)
c
c... this subroutine compute the coefficients for multispecies mass
c   balance equation ...
c
c
c   implicit real*8 (a-h,o-z)
c   parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c
c   dimension ck1(npm),ck2(npm,ndm),ck3(npm,ndm,ndm),cg(npm)
c
c   dimension vel(npm,nam,ndm,nnm),ijk(npm,nnm)
c
c   dimension
c   &sat(npm,nam),rk(npm,nam,ndm,ndm),rd(npm,nsm,nam,ndm,ndm)
c   dimension rn(npm)
c   dimension rho(npm,4),rhor(nam)
c   dimension source(npm,nsm,nam)
c
c   common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
c   &          ipx,ipy,npz,ndmp,nsmp,namp,nnmp
c   common /set1/ tmax,time,dt,cp,axi,aeta,rgr,ptime,dptime,
c   &          pltime,dpltim,erc,wold,wnew
c
c   common /dens/ rho,rhor
c   common /poro/ rn
c   common /sour/ source
c   common /disp/ rd
c   common /coef/ ck1,ck2,ck3,cg
c
c
c   hawo = 0.01
c   hswo = 0.01
c
c... initialize ...
c
c   do 2 ip = 1,npmax
c     ck1(ip) = 0.d0
2   continue
c
c   do 4 id = 1,ndmp
c   do 4 jd = 1,ndmp
c   do 4 ip = 1,npmax

```

```

        ck3(ip,id,jd) = 0.d0
4      continue
c
      do 10 ip = 1,npmax
        ck1(ip) = 1
cl     ck1(ip) = (1.d0 + sat(ip,3)*rho(ip,3)/sat(ip,1)/rho(ip,1)*hawo
cl     & + (1-rn(ip))*rho(ip,4)/rn(ip)/sat(ip,1)/rho(ip,1)*hsw0)
c     & *rn(ip)*sat(ip,1)*rho(ip,1)
10    continue
c
      do 20 id = 1,ndmp
        do 20 ip = 1,npmax
          ck3(ip,id,id) = rd(ip,2,1,id,id)
cl     ck3(ip,id,id) = rd(ip,2,1,id,id)*(1.d0
cl     & + rd(ip,2,3,id,id)*sat(ip,3)*rho(ip,3)*hawo
cl     & /rd(ip,2,1,id,id)/sat(ip,1)/rho(ip,1))
c     & *rn(ip)*sat(ip,1)*rho(ip,1)
20    continue
c
c     write(3,*) 'ck1(11),ck3(11,1,1) = ',ck1(11),ck3(11,1,1)
c
      return
      end

```

```

c
c
c
c      subroutine sourit
c
c... this subroutine computes the point sources
c      such as pumping, recharging, chemical, biological reaction ...
c
c
c
c      implicit real*8 (a-h,o-z)
c      parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c      dimension source(npm,nsm,nam),ijk(npm,nnm),sourl(npm,nam)
c
c      common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S          ipx,ipy,npz,ndmp,nsmpr,namp,nnmp
c      common /setl/ tmax,time,dt,ep,axi,acta,rgr,ptime,dptime,
S          pltime,dpltim,erc,wold,wnew
c      common /sour/ source
c
c      do 5 is = 1,nsmpr
c      do 5 ia = 1,namp
c          do 5 ip = 1,npmax
c              source(ip,is,ia) = 0.d0
5          continue
c
c      source(1,1,1) = 13.d0
c
c... debug ...
c
c      if(time.le.-1) then
c          write(3,*)
c          write(3,*) 'point source'
c          do 50 is = 1,nsmpr
c              write(3,*) 'is = ',is
c          do 50 ia = 1,namp
c              write(3,*) 'ia = ',ia
c          do 60 ip = 1,npmax
c              sourl(ip,ia) = source(ip,is,ia)
60          continue
c          call outp(sourl,ia,3)
50          continue
c      endif
c
c      return
c      end

```

```

c
c
c
  subroutine femh(h,oldh,sat,oldsat)
c
c... this subroutine is for finite element scheme about
c  pressure head ...
c
c
c  implicit real*8 (a-h,o-z)
c  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c  dimension
c  &agh(npm,45,nsm,nam),fh(npm,nsm,nam),fn(npm,nsm),
c  &h(npm,nam),oldh(npm,nam),dh(npm,nam),hiter(npm,nam),
c  &x(npm),y(npm),z(npm),ijk(npm,nnm),sat(npm,nam),oldsat(npm,nam),
c  &ramda(2),ramdao(2),ramdas(2),dpmax(2),dpmaxo(2),s(2),adh(2),
c  &ab(npm,45),lb(npm)
c
c  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
c  S          ipx,ipy,npx,ppy,ndmp,nsmp,namp,nnmp
c  common /set1/ tmax,time,dt,ep,axi,acta,rgr,ptime,dptime,
c  S          pltime,dpltim,erc,wold,wnew
c  common /space/ x,y,z
c  common /part2/ c,oldc
c
c
c... iteration criteria ...
c
c4      ercc = 0.005d0
c31     ercc = 0.010d0
c
c      ercc = 0.10d0
c      adh(1) = 10.d0
c      adh(2) = 10.d0
c      ercre = ercc*0.01
c      ercab = ercc
c      imin = 4
c      imax = 8
c      imaxx = 50
c      timefa = 0.10
c
c      if(time.eq.dt) write(2,1000) imax,axi,ep,wold
1000  format(t2,'imax,weight,ep,wold = ',i3,3f7.4)
c
c      do 10 ia = 1,namp
c      do 10 ip = 1,npmax

```

```

        hiter(ip,ia) = oldh(ip,ia)
10    continue
    c
    c
    c
    c    ***** picard iteration *****
    c
        iter = 0
20    iter = iter + 1
    c
    c    increase or decrease time step
    c
        if(iter.lt.imin) dt = dt*(1 + timefa)
        if(iter.gt.imax) dt = dt/(1 + timefa)
    c
        if(iter.gt.imaxx) goto 999
    c
    c... compute global matrix and load vector
    c
        call asemh(h,oldh,sat,oldsat,agh,fh)
    c
    c... initialize load vector ...
    c
        do 25 is = 1,nsmp
        do 25 ip = 1,npmax
25    fn(ip,is) = 0.d0
    c
        do 30 is = 1,nsmp
    c
        if(is.eq.1) iaa = 2
        if(is.eq.2) iaa = 1
    c
    c... load vector ...
    c
        do 40 ip = 1,npmax
        fn(ip,is) = fh(ip,is,1) + fh(ip,is,2)
        do 40 j = 1,iband
            jj = ip + j - (ihalfb + 1)
            if(jj.lt.1) goto 40
            if(jj.gt.npmax) goto 40
            fn(ip,is) = -agh(ip,j,is,iaa)*h(jj,iaa) + fn(ip,is)
40    continue
    c
        do 45 ip = 1,npmax
45    fb(ip) = fn(ip,is)
    c

```

```

c... banded ag matrix for water species in water phase ...
c
  do 50 j = 1,iband
    do 50 ip = 1,npmax
50      ab(ip,j) = agh(ip,j,is,is)
c
c... debug ...
c
      if(time.le.dt) then
        write(3,*)
        write(3,*) 'is = ',is
        write(3,*) 'global banded matrix'
        write(3,1500) ((ab(ip,j),j = 1,3),ip = 1,3)
1500    format(3e9.2)
c32      write(3,1500) ((ab(ip,j),j = 1,9),ip = 1,6)
c32 1500    format(9e9.2)
c31      write(3,1500) ((ab(ip,j),j = 1,3),ip = 1,3)
c31 1500    format(3e9.2)
        endif
c
c... solve for water pressure head ...
c
      call solve(1,ab,fb,npmax,ihalfb)
c
      call solve(2,ab,fb,npmax,ihalfb)
c
c... store the solution in pressure head h(ip,is) ...
c
      do 60 ip = 1,npmax
        h(ip,is) = fb(ip)
60      continue
c
c... debug ...
c
      if(time.le.dt) then
        write(3,*)
        write(3,*) 'is,iteration = ',is,iter
        call outp(h,is,3)
      endif
c
30      continue
c
c... cooly's iteration scheme ...
c
      do 65 is = 1,nsmp
        dpmax(is) = -10e6

```



```

do 70 ip = 1,npmax
c   h(ip,is) = (wnew*h(ip,is) + wold*oldh(ip,is))/(wnew + wold)
      dh(ip,is) = h(ip,is)-hiter(ip,is)
      abdh = dabs(dh(ip,is))
      if(abdh.gt.dpmax(is)) dpmax(is) = abdh
70  continue
      if(iter.eq.1) then
        s(is) = 1.d0
      else
        s(is) = dpmax(is)/ramdao(is)/dpmaxo(is)
      endif
      if(s(is).ge.-1.d0) then
        ramdas(is) = (3 + s(is))/(3 + dabs(s(is)))
      else
        ramdas(is) = 0.5/(dabs(s(is)))
      endif
      adpp = adh(is)/dpmax(is)
      if(ramdas(is).le.adpp) then
        ramda(is) = ramdas(is)
      else
        ramda(is) = adpp
      endif
c
do 80 ip = 1,npmax
      h(ip,is) = ramda(is)*dh(ip,is) + hiter(ip,is)
80  continue
c
c... debug ...
c
c   if(time.le.dt) then
c     write(3,*)
c     write(3,*) 'is,iteration = ',is,iter
c     call outp(h,is,3)
c   endif
c
65  continue
c30 continue
c
c... reset the variables ...
c
do 90 is = 1,nsmp
      ramdao(is) = ramdas(is)
      dpmaxo(is) = dpmax(is)
do 90 ip = 1,npmax
      hiter(ip,is) = h(ip,is)
90  continue
c

```

```

c
c
c... check the error criteria ...
c
    if(time.le.dt) then
        write(3,2000) dt,time,iter,dpmax(1),dpmax(2)
    endif
    do 100 ia = 1,namp
    do 100 ip = 1,npmax
        erc = ercre*dabs(oldh(ip,ia)) + ercab
c4     erc = ercab
        if(dpmax(ia).gt.erc) goto 20
100    continue
c
c time step, time, maximum iteration, maximum error of water & oil head
c
999 continue
c   if(time.ge.ptime) then
        write(3,2000) dt,time,iter,dpmax(1),dpmax(2)
c   endif
c4   if(iter.lt.imin) dt = dt*(1 + timefa)
c4   if(iter.gt.imax) dt = dt/(1 + timefa)
2000 format('dt,time,iter,dp =',2f10.3,i5,2f10.6)
c
    return
    end

```

```

c
c
c
  subroutine femc(h,oldh,sat,oldsat)
c
c... this subroutine is for finite element scheme
c about mass fraction ...
c
c
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension
    &h(npm,nam),oldh(npm,nam),sat(npm,nam),oldsat(npm,nam),
    &age(npm,45),fc(npm),wm(npm,nsm,4),c(npm),oldc(npm),
    &ab(npm,45),fb(npm),wmp(npm,nam),ijk(npm,nnm),oldwm(npm,nsm,4),
    &rn(npm),rho(npm,4),rhor(nam)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S      ipx,ipy,npz,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
S      pltime,dpltim,erc,wold,wnew
  common /part/ wm,oldwm
  common /part2/ c,oldc
  common /poro/ rn
  common /dens/ rho,rhor
c
c... compute global matrix and load vector
c
  call asemc(h,oldh,sat,oldsat,age,fc)
c
c
  do 45 ip = 1,npmax
45   fb(ip) = fc(ip)
c
c... banded ag matrix for oil species in water phase ...
c
  do 50 j = 1,iband
    do 50 ip = 1,npmax
50   ab(ip,j) = age(ip,j)
c
c... debug ...
c
  if(time.le.dt) then
    write(3,*)
    write(3,*) 'is = ',is
    write(3,*) 'global banded matrix'

```

```

        write(3,1500) ((ab(ip,j),j = 1,3),ip = 1,3)
1500  format(3e9.2)
c32  write(3,1500) ((ab(ip,j),j = 1,9),ip = 1,6)
c32 1500  format(9e9.2)
c31  write(3,1500) ((ab(ip,j),j = 1,3),ip = 1,3)
c31 1500  format(3e9.2)
        endif
c
c... solve for mass fraction ...
c
        call solve(1,ab,fb,npmax,ihalfb)
c
        call solve(2,ab,fb,npmax,ihalfb)
c
c... store the solution in mass fraction w(ip,2,1) ...
c
        do 60 ip = 1,npmax
            c(ip) = fb(ip)
            wm(ip,2,1) = c(ip)
c1    wm(ip,2,1) = c(ip)/rn(ip)/rho(ip,1)/sat(ip,1)
            wmp(ip,2) = fb(ip)
60    continue
c
c... debug ...
c
        if(time.le.dt) then
            write(3,*)
            call outp(wmp,2,3)
        endif
c
30  continue
c
c
        return
        end

```

```

c
c
c
  subroutine asemh(h,oldh,sat,oldsat,agh,fh)
c
c... this subroutine assembles the element matrices
c  about pressure head ...
c
c
c  implicit real*8 (a-h,o-z)
c  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c  dimension
c  &agh(npm,45,nsm,nam),fh(npm,nsm,nam),
c  &h(npm,nam),oldh(npm,nam),dh(npm,nam),
c  &c(npm),oldc(npm),
c  &x(npm),y(npm),z(npm),ijk(npm,nnm),fc(npm,nsm,nam,nnm)
c
c  dimension
c  &eth(npm,nsm,nam,nnm,nnm),ekh(npm,nsm,nam,ndm,ndm,nnm,nnm),
c  &etm(npm,nam,nnm,nnm),ekm(npm,nam,ndm,nnm,nnm),
c  &etc(npm,nnm,nnm),ekc(npm,ndm,nnm,nnm),,
c  &edc(npm,ndm,ndm,nnm,nnm),egh(npm,nsm,nam,ndm,ndm,nnm,nnm),
c  &edm(npm,nsm,nam,ndm,ndm,nnm,nnm), &cs(npm,nnm,nnm)
c
c  dimension velp(npm,nam),weifp(npm,nam),
c  &weil(npm,nam,ndm,nnm),vel(npm,nam,ndm,nnm)
c
c  dimension
c  &w(npm,nam,nnm,nnm),dwr(npm,nam,ndm,nnm,nnm),
c  &bn(nnm,nnm),bnd(ndm,nnm,nnm),bndr(ndm,nnm,nnm),
c  &bnr(ndm,nnm,nnm),
c  &dnr(npm,ndm,nnm,nnm),dji(npm,nam,nam,nnm),det(npm,nnm),
c  &wdn(ndm,nnm,nnm),dwdn(ndm,nnm,nnm),sh(npm,nam,nam)
c
c  dimension
c  &sat(npm,nam),oldsat(npm,nam),dsh(npm,nam,nam),
c  &rk(npm,nam,ndm,ndm)
c  dimension rn(npm),dnh(npm,nam)
c  dimension wm(npm,nsm,4),oldwm(npm,nsm,4),wmp(npm,nam)
c  dimension rho(npm,4),rhor(nam),drh(npm,4,nam)
c  dimension sourcec(npm,nsm,nam)
c
c  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
c  &      ipx,ipy,npx,npy,ndmp,nsmp,namp,nnmp
c  common /set1/ tmax,time,dt,ep,axi,acta,rgr,ptime,dptime,
c  &      pltime,dpltim,erc,wold,wnew

```

```

common /space/ x,y,z
c
c common /ib/ indb1,indb2,indb3
c
common /bas/ bn,bnd,bndr,bnr,dnr,dji,det,wdn,dwdn
common /wei/ w,dwr,weif
c
common /dens/ rho,rhor
common /part/ wm,oldwm
common /part2/ c,olde
common /poro/ rn
common /sour/ source
c
c... weighting factor for generalized fdm ...
c
eps = ep-1.d0
c
c... initialize the matrix ag and vector f ...
c
do 5 is = 1,nsmp
do 5 ia = 1,namp
do 5 j = 1,iband
do 5 ip = 1,npmax
agh(ip,j,is,ia) = 0.d0
fh(ip,is,ia) = 0.d0
5 continue
c
c... saturation wrt capillary head & relative permeability ...
c
call dsat(h,oldh,sat,oldsat,sh,rk)
call partit(sat)
c
c... velocity of each phase
c and weighting factor for upstream weighting ...
c
call veloc(h,rk,sat,vel)
c write(3,*) 'jceral = ',vel(1,1,1,1)
c
c... debug ...
c
c if(time.le.dt) then
c write(3,*)
c do 20 ia = 1,namp
c do 22 in = 1,nnmp
c write(3,*) 'ia,in = ',ia,in
c do 24 ic = 1,nelmax
c ip = ijk(ic,in)

```

```

c      velp(ip,ia) = vel(ic,ia,1,in)
c      weifp(ip,ia) = weif(ic,ia,1,in)
c24   continue
c      write(3,*) 'velocity'
c      call outp(velp,ia,3)
c      write(3,*) 'weighting factor'
c      call outp(weifp,ia,3)
c22   continue
c20   continue
c     endif
c
c... debug ...
c
      do 7 ip = 1,npmax
        wmp(ip,2) = wm(ip,2,1)
7     continue
c
      if(time.le.dt) then
        write(3,*) 'water saturation'
        call outp(sat,1,3)
        write(3,*) 'oil saturation'
        call outp(sat,2,3)
        write(3,*) 'water pressure'
        call outp(h,2,3)
        write(3,*) 'oil pressure'
        call outp(h,2,3)
        write(3,*) 'mass fraction'
        call outp(wmp,2,3)
      endif
c
c
c... evaluate the weighting functions and its derivatives ...
c
      if(ndmp.eq.1) call weigh1
      if(ndmp.eq.2) call weigh2
      if(ndmp.eq.3) call weigh3
c
c
c ***** assembling procedure *****
c
c
c... compute element matrices ...
c
      call elemen(eth,etm,etc,ekh,ekm,ekc,egh,edm,edc,es,sh,rk,sat,vel)
c
c1   elemen(eth,ekh,egh,edm,es,sc,nel)

```

```

cb
c
c... debug ...
c
c   if(time.le.dt) then
c     do 30 ie = 1,nelmax
c       write(3,*) 'element =',ie
c       write(3,*) 'et'
c       write(3,2000) ((eth(ic,1,1,i,j),j = 1,nnmp),i = 1,nnmp)
c       write(3,2000) ((eth(ic,1,2,i,j),j = 1,nnmp),i = 1,nnmp)
c       write(3,2000) ((eth(ic,2,1,i,j),j = 1,nnmp),i = 1,nnmp)
c       write(3,2000) ((eth(ic,2,2,i,j),j = 1,nnmp),i = 1,nnmp)
c       write(3,*) 'ek'
c       write(3,2000) ((ekh(ic,1,1,1,1,i,j),j = 1,nnmp),i = 1,nnmp)
c       write(3,2000) ((ekh(ic,1,2,1,1,i,j),j = 1,nnmp),i = 1,nnmp)
c       write(3,2000) ((ekh(ic,2,1,1,1,i,j),j = 1,nnmp),i = 1,nnmp)
c       write(3,2000) ((ekh(ic,2,2,1,1,i,j),j = 1,nnmp),i = 1,nnmp)
c       write(3,*) 'eg'
c       write(3,2000) (egh(ic,1,1,2,2,j,j),j = 1,nnmp)
c       write(3,2000) (egh(ic,2,2,2,2,j,j),j = 1,nnmp)
c 2000   format(4f10.3)
c 30     continue
c       endif
c
c
c... assemble element matrices to form banded global form ag ...
c
c   do 50 is = 1,nsmp
c     do 50 ia = 1,namp
c       do 50 i = 1,nnmp
c         do 50 j = 1,nnmp
c           do 50 ie = 1,nelmax
c             ijki = ijk(ic,i)
c             ijkj = ijk(ic,j)
c             ji = ijkj + (ihalfb + 1) - ijki
c             if(ji.le.0) goto 50
c             if(ji.gt.iband) goto 50
c             agh(ijki,ji,is,ia) = agh(ijki,ji,is,ia) + eth(ic,is,ia,i,j) +
c             S       ep*(ekh(ic,is,ia,1,1,i,j) + ekh(ic,is,ia,2,2,i,j))
c31 S       ep*(ekh(ic,is,ia,1,1,i,j))
c32 S       ep*(ekh(ic,is,ia,1,1,i,j) + ekh(ic,is,ia,2,2,i,j))
c       50   continue
c
c... assemble element matrices to form r.h.s. vector f ...
c
c   do 70 i = 1,nnmp
c     do 75 ie = 1,nelmax

```



```

        fe(ic,1,1,i) = 0.d0
        fe(ic,1,2,i) = 0.d0
        fe(ic,2,1,i) = 0.d0
        fe(ic,2,2,i) = 0.d0
75    continue
c
c31    & -(egh(ic,is,ia,1,1,i,j)) + (eth(ic,is,ia,i,j)
c32    & -(egh(ic,is,ia,1,2,i,j) + egh(ic,is,ia,2,2,i,j))
c      & + (eth(ic,is,ia,i,j)
c2    & -(egh(ic,is,ia,1,2,i,j) + egh(ic,is,ia,2,2,i,j))
c      & + (eth(ic,is,ia,i,j)
c
        do 80 j = 1,nnmp
        do 80 is = 1,nsmp
        do 80 ia = 1,namp
        do 80 ic = 1,nelmax
            ijkj = ijk(ic,j)
            fe(ic,is,ia,i) = fe(ic,is,ia,i)
            & -(egh(ic,is,ia,1,2,i,j) + egh(ic,is,ia,2,2,i,j)) + (eth(ic,is,ia,i,j)
            & + eps*(ekh(ic,is,ia,1,1,i,j) + ekh(ic,is,ia,2,2,i,j))*oldh(ijkj,ia)
c31    & + eps*(ekh(ic,is,ia,1,1,i,j))*oldh(ijkj,ia)
c32    & + eps*(ekh(ic,is,ia,1,1,i,j) + ekh(ic,is,ia,2,2,i,j))*oldh(ijkj,ia)
c      & -etm(ic,ia,i,j)*(wm(ijkj,is,ia)-oldwm(ijkj,is,ia))
c      & -(ekm(ic,ia,1,i,j))*wm(ijkj,is,ia)
c32    & -(ekm(ic,ia,1,i,j) + ekm(ic,ia,2,i,j))*wm(ijkj,is,ia)
c32    & -(edm(ic,is,ia,1,1,i,j) + edm(ic,is,ia,2,2,i,j))
c      & -(edm(ic,is,ia,1,1,i,j))
c      & *wm(ijkj,is,ia)
            & + es(ic,i,j)*source(ijkj,is,ia)
c      & -es(ic,i,j)*(1-rn(ijkj))*rho(ijkj,4)
c      & *(wm(ijkj,is,4)-oldwm(ijkj,is,4))/dt
80    continue
        do 90 is = 1,nsmp
        do 90 ia = 1,namp
        do 90 ic = 1,nelmax
            ijki = ijk(ic,i)
            fh(ijki,is,ia) = fh(ijki,is,ia) + fe(ic,is,ia,i)
90    continue
70    continue
c
c
c... print the banded global matrix & load vector of each species ...
c
c    if(time.le.dt) then
c    write(3,*) 'assembled global banded matrix'
c    do 100 is = 1,nsmp
c    do 100 ia = 1,namp

```

```

c      write(3,*) 'species, phase = ',is,ia
c      write(3,1000) ((agh(ip,j,is,ia),j = 1,10),ip = 1,6)
c      write(3,*) 'load vector'
c      write(3,1000) (fh(ip,is,ia),ip = 1,6)
c100 continue
c      endif
c
c
c      ***** force boundary conditions *****
c
c
c      do 105 is = 1,nsmp
c      if(indb1.eq.1) call bound1(agh,fh,is,is)
c      if(indb2.eq.1.and.ndmp.eq.1) call bound2(fh,is,is)
c      if(indb2.eq.1.and.ndmp.eq.2) call bound22(fh,is,is)
c      if(indb2.eq.1.and.ndmp.eq.3) call bound32(fh,is,is)
105 continue
c
c... print the banded global matrix & load vector of each species ...
c
c      if(time.le.dt) then
c      write(3,*) 'assembled global banded matrix after b.c.'
c      do 110 is = 1,nsmp
c      do 110 ia = 1,namp
c      write(3,*) 'species, phase = ',is,ia
c      write(3,1000) ((agh(ip,j,is,ia),j = 1,10),ip = 1,6)
c      write(3,*) 'load vector'
c      write(3,1000) (fh(ip,is,ia),ip = 1,6)
c110 continue
c
c1000 format(10e9.2)
c      endif
c
c
c      return
c      end

```

```

c
c
c
  subroutine asemc(h,oldh,sat,oldsat,agc,fc)
c
c... this subroutine assembles the element matrices
c about mass fraction ...
c
c
c   implicit real*8 (a-h,o-z)
c   parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c   dimension
c   &age(npm,45),fc(npm),fec(npm,nnm),
c   &h(npm,nam),oldh(npm,nam),dh(npm,nam),
c   &c(npm),oldc(npm),
c   &x(npm),y(npm),z(npm),ijk(npm,nnm),fc(npm,nsm,nam,nnm)
c
c   dimension
c   &eth(npm,nsm,nam,nnm,nnm),ekh(npm,nsm,nam,ndm,ndm,nnm,nnm),
c   &etm(npm,nam,nnm,nnm),ekm(npm,nam,ndm,nnm,nnm),
c   &etc(npm,nnm,nnm),ekc(npm,ndm,nnm,nnm),
c   &edc(npm,ndm,ndm,nnm,nnm),egh(npm,nsm,nam,ndm,ndm,nnm,nnm),
c   &edm(npm,nsm,nam,ndm,ndm,nnm,nnm),cs(npm,nnm,nnm)
c
c   dimension velp(npm,nam),weifp(npm,nam),
c   &weif(npm,nam,ndm,nnm),vel(npm,nam,ndm,nnm)
c
c   dimension
c   &w(npm,nam,nnm,nnm),dwr(npm,nam,ndm,nnm,nnm),
c   &bn(nnm,nnm),bnd(ndm,nnm,nnm),bndr(ndm,nnm,nnm),
c   &bnr(ndm,nnm,nnm),
c   &dnr(npm,ndm,nnm,nnm),dji(npm,nam,nam,nnm),det(npm,nnm),
c   &wdn(ndm,nnm,nnm),dwdn(ndm,nnm,nnm),sh(npm,nam,nam)
c
c   dimension
c   &sat(npm,nam),oldsat(npm,nam),dsh(npm,nam,nam),
c   &rk(npm,nam,ndm,ndm)
c   dimension rn(npm),dnh(npm,nam)
c   dimension wm(npm,nsm,4),oldwm(npm,nsm,4),wmp(npm,nam)
c   dimension rho(npm,4),rhor(nam),drh(npm,4,nam)
c   dimension source(npm,nsm,nam)
c
c   common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
c   &          ipx,ipy,npx,ipy,ndmp,nsmp,namp,nnmp
c   common /setl/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
c   &          pltime,dpltim,erc,wold,wnew

```

```

common /space/ x,y,z
c
common /ib/ indb1,indb2,indb3
c
common /bas/ bn,bnd,bndr,bnr,dnr,dji,det,wdn,dwdn
common /wei/ w,dwr,weif
c
common /dens/ rho,rhor
common /part/ wm,oldwm
common /part2/ c,olde
common /poro/ rn
common /sour/ source
c
c... weighting factor for generalized fdm ...
c
    eps = ep-1.d0
c
c... initialize the matrix agc and vector fc ...
c
    do 5 j = 1,iband
        do 5 ip = 1,npmax
            agc(ip,j) = 0.d0
            fc(ip) = 0.d0
5    continue
c
c... saturation wrt capillary head & relative permeability ...
c
    call dsat(h,oldh,sat,oldsat,sh,rk)
    call partit(sat)
c
c... velocity of each phase
c    and weighting factor for upstream weighting ...
c
    call veloc(h,rk,sat,vel)
c
c... debug ...
c
c    if(time.le.dt) then
c    write(3,*)
c    do 20 ia = 1,namp
c    do 22 in = 1,nnmp
c        write(3,*) 'ia,in = ',ia,in
c        do 24 ie = 1,nelmax
c            ip = ijk(ie,in)
c            velp(ip,ia) = vel(ie,ia,1,in)
c            weifp(ip,ia) = weif(ie,ia,1,in)
c24    continue

```

```

c      write(3,*) 'velocity'
c      call outp(velp,ia,3)
c      write(3,*) 'weighting factor'
c      call outp(weifp,ia,3)
c22   continue
c20   continue
c     endif
c
c... debug ...
c
c     do 7 ip = 1,npmax
c       wmp(ip,2) = wm(ip,2,1)
7     continue
c
c     if(time.le.dt) then
c       write(3,*) 'water saturation'
c       call outp(sat,1,3)
c       write(3,*) 'oil saturation'
c       call outp(sat,2,3)
c       write(3,*) 'water pressure'
c       call outp(h,2,3)
c       write(3,*) 'oil pressure'
c       call outp(h,2,3)
c       write(3,*) 'mass fraction'
c       call outp(wmp,2,3)
c     endif
c
c
c... evaluate the weighting functions and its derivatives ...
c
c     if(ndmp.eq.1) call weigh1
c     if(ndmp.eq.2) call weigh2
c     if(ndmp.eq.3) call weigh3
c
c
c     ***** assembling procedure *****
c
c
c... compute element matrices ...
c
c     call elemen(eth,ctm,etc,ekh,ekm,ekc,egh,edm,edc,es,sh,rk,sat,vel)
c
c
c... debug ...
c
c     if(time.le.dt) then

```

```

c      do 30 ie = 1,nelmax
c      write(3,*) 'element = ',ie
c      'write(3,*) 'etc'
c      write(3,2000) ((etc(ie,i,j),j = 1,nnmp),i = 1,nnmp)
c      write(3,*) 'ekc'
c      write(3,2000) ((ekc(ie,1,i,j),j = 1,nnmp),i = 1,nnmp)
c      write(3,2000) ((ekc(ie,2,i,j),j = 1,nnmp),i = 1,nnmp)
c      write(3,*) 'edc'
c      write(3,2000) ((edc(ie,1,1,i,j),j = 1,nnmp),i = 1,nnmp)
c      write(3,2000) ((edc(ie,2,2,i,j),j = 1,nnmp),i = 1,nnmp)
c 2000  format(4f10.3)
c 30    continue
c      endif
c
c
c... assemble element matrices to form banded global form ag ...
c
      do 57 i = 1,nnmp
      do 57 j = 1,nnmp
      do 57 ie = 1,nelmax
      ijki = ijk(ie,i)
      ijkj = ijk(ie,j)
      ji = ijkj + (ihalfb + 1)-ijki
      if(ji.le.0) goto 57
      if(ji.gt.iband) goto 57
      agc(ijkj,ji) = agc(ijki,ji) + etc(ie,i,j)
      S  + ep*(ekc(ie,1,i,j) + ekc(ie,2,i,j)
      S  + edc(ie,1,1,i,j) + edc(ie,2,2,i,j))
c31 S  + ep*(ekc(ie,1,i,j)
c31 S  + edc(ie,1,1,i,j))
c32 S  + ep*(ekc(ie,1,i,j) + ekc(ie,2,i,j)
c32 S  + edc(ie,1,1,i,j) + edc(ie,2,2,i,j))
      57  continue
c
c... assemble element matrices to form r.h.s. vector fc ...
c
      do 292 i = 1,nnmp
      do 295 ie = 1,nelmax
      fec(ie,i) = 0.d0
295  continue
c
      do 297 j = 1,nnmp
      do 297 ie = 1,nelmax
      ijkj = ijk(ie,j)
      fec(ie,i) = fec(ie,i)
      & + (etc(ie,i,j)
      & + eps*(ekc(ie,1,i,j) + ekc(ie,2,i,j)

```

```

    & +edc(ie,1,1,i,j) + edc(ie,2,2,i,j))) *oldc(ijkj)
c31 & +eps*(ekc(ie,1,i,j)
c31' & +edc(ie,1,1,i,j))) *oldc(ijkj)
c32 & +eps*(ekc(ie,1,i,j) + ekc(ie,2,i,j)
c32 & +edc(ie,1,1,i,j) + edc(ie,2,2,i,j))) *oldc(ijkj)
    & +es(ie,i,j)*(source(ijkj,2,1) + source(ijkj,2,2) + source(ijkj,2,3))
297     continue
    do 298 ic = 1,nelmax
        ijki = ijk(ic,i)
        fc(ijki) = fc(ijki) + fec(ie,i)
298     continue
292 continue
c
c
c... print the banded global matrix & load vector of each species ...
c
c   if(time.le.dt) then
c   write(3,*) 'assembled global banded matrix'
c     write(3,*) 'species,phase =',is,ia
c     write(3,1000) ((agc(ip,j),j = 1,10),ip = 1,6)
c     write(3,*) 'load vector'
c     write(3,1000) (fc(ip),ip = 1,6)
c   endif
c
c
c   ***** force boundary conditions *****
c
c
c   if(indb1.cq.1) call bouncl(agc,fc,sat)
c
c... print the banded global matrix & load vector of each species ...
c
c   if(time.le.dt) then
c   write(3,*) 'assembled global banded matrix after b.c.'
c     write(3,*) 'species, phase =',is,ia
c     write(3,1000) ((agc(ip,j),j = 1,10),ip = 1,6)
c     write(3,*) 'load vector'
c     write(3,1000) (fc(ip),ip = 1,6)
c
c1000 format(10e9.2)
c   endif
c
c
c   return
c   end

```

```

c
c
c
  subroutine elcmen
  & (eth,etm,etc,ekh,ekm,ekc,egh,edm,edc,es,sh,rk,sat,vel)
c
c... this subroutine construct element matrices
c by gaussian integration ...
c
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension
  &x(npm),y(npm),z(npm),ijk(npm,nnm),vel(npm,nam,ndm,nnm),
  &eth(npm,nsm,nam,nnm,nnm),ekh(npm,nsm,nam,ndm,ndm,nnm,nnm),
  &etm(npm,nam,nnm,nnm),ekm(npm,nam,ndm,nnm,nnm),
  &etc(npm,nnm,nnm),ekc(npm,ndm,nnm,nnm),
  &es(npm,nnm,nnm),et(nnm,nnm),ek(nnm,nnm),ed(nnm,nnm),
  &egh(npm,nsm,nam,ndm,ndm,nnm,nnm),edc(npm,ndm,ndm,nnm,nnm),
  &edm(npm,nsm,nam,ndm,ndm,nnm,nnm)
c
  dimension
  &bn(nnm,nnm),bnd(ndm,nnm,nnm),bndr(ndm,nnm,nnm),
  &bnr(ndm,nnm,nnm),
  &dnr(npm,ndm,nnm,nnm),
  &dj(nam,nam),dji(npm,nam,nam,nnm),det(npm,nnm),
  &wdn(ndm,nnm,nnm),dwdn(ndm,nnm,nnm)
c
  dimension
  &w(npm,nam,nnm,nnm),dwr(npm,nam,ndm,nnm,nnm),
  &weif(npm,nam,ndm,nnm)
c
  dimension rkg(npm,nsm,nam,ndm,ndm,nnm)
  dimension rkmg(npm,nam,ndm,nnm)
  dimension sc(npm,nsm,nam,nam),scg(npm,nsm,nam,nam,nnm)
  dimension shmg(npm,nam,nnm)
  dimension sh(npm,nam,nam),shg(npm,nsm,nam,nam,nnm)
  dimension
  &rd(npm,nsm,nam,ndm,ndm),rdg(npm,nsm,nam,ndm,ndm,nnm)
c
  dimension
  &sat(npm,nam),oldsat(npm,nam),dsh(npm,nam,nam),
  &rk(npm,nam,ndm,ndm)
  dimension wm(npm,nsm,4),oldwm(npm,nsm,4)
  dimension rho(npm,4),rhor(nam),drh(npm,4,nam)
  dimension rn(npm),dnh(npm,nam)

```



```

dimension ck1(npm),ck2(npm,ndm),ck3(npm,ndm,ndm),cg(npm)
dimension
&ck1g(npm,nnm),ck2g(npm,ndm,nnm),ck3g(npm,ndm,ndm,nnm)
c
common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S      ipx,ipy,npx,ppy,ndmp,nsmp,namp,nnmp
common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
S      pltime,dpltim,erc,wold,wnew
common /space/ x,y,z
common /bas/  bn,bnd,bndr,bnr,dnr,dji,det,wdn,dwdn
common /wei/  w,dwr,weif
c
common /dens/ rho,rhor
common /part/ wm,oldwm
common /poro/ rn
common /disp/ rd
common /coef/ ck1,ck2,ck3,cg
c
c... parameter for multispecies eq'n ...
c
call coeff(sat,vel)
c
c... initialize element matrices ...
c
do 10 is = 1,nsmp
do 10 ia = 1,namp
do 10 i = 1,nnmp
do 10 j = 1,nnmp
do 10 id = 1,ndmp
do 10 jd = 1,ndmp
do 10 ie = 1,nelmax
et(i,j) = 0.d0
ek(i,j) = 0.d0
ed(i,j) = 0.d0
es(ic,i,j) = 0.d0
eth(ic,is,ia,i,j) = 0.d0
etm(ic,ia,i,j) = 0.d0
etc(ie,i,j) = 0.d0
ekh(ie,is,ia,id,jd,i,j) = 0.d0
ckm(ic,ia,id,i,j) = 0.d0
ckc(ie,id,i,j) = 0.d0
egh(ie,is,ia,id,jd,i,j) = 0.d0
edm(ie,is,ia,id,jd,i,j) = 0.d0
cdc(ie,id,jd,i,j) = 0.d0
10  continuc
c
c

```

```

c ***** element wise evaluation of parameters *****
c
c
do 15 is = 1,nsmp
do 15 ia = 1,namp
do 15 ja = 1,nam
do 15 i = 1,nnmp
do 15 id = 1,ndmp
do 15 jd = 1,ndmp
do 15 ie = 1,nelmax
shg(ic,is,ia,ja,i) = 0.d0
shmg(ic,ia,i) = 0.d0
rdg(ic,is,ia,id,jd,i) = 0.d0
rkmg(ic,ia,id,i) = 0.d0
rkg(ic,is,ia,id,jd,i) = 0.d0
ck1g(ic,i) = 0.d0
ck2g(ic,id,i) = 0.d0
ck3g(ic,id,jd,i) = 0.d0
15 continue
c
c if(time.le.-1) then
c write(3,*)
c write(3,*) 'velocity'
c do 17 id = 1,ndmp
c do 17 ia = 1,namp
c do 17 ie = 1,nelmax
c write(3,1100) id,ia,ie,(vel(ic,ia,id,in),in = 1,nnmp)
c17 continue
c1100 format('id,ia,ie = ',3i4,8e14.5)
c write(3,*)
c write(3,*) 'k1'
c write(3,1200) (ip,ck1(ip),ip = 1,npmax)
c1200 format(5(' ip = ',i3,e14.5))
c write(3,*)
c write(3,*) 'k3'
c do 18 id = 1,ndmp
c do 18 jd = 1,ndmp
c write(3,*) 'id,jd = ',id,jd
c write(3,1200) (ip,ck3(ip,id,jd),ip = 1,npmax)
c18 continue
c endif
c
c
do 20 i = 1,nnmp
do 30 ig = 1,nnmp
c
do 32 ie = 1,nelmax

```

```

    ijkig = ijk(ie,ig)
    ck1g(ie,i) = ck1g(ie,i) + ck1(ijkig)*bn(i,ig)
32  continue
c
c
    do 34 id = 1,ndmp
    do 34 ie = 1,nelmax
        ijki = ijk(ie,i)
        ck2g(ie,id,i) = ck2g(ie,id,i) + vel(ie,1,id,i)*bn(i,ig)
c      ck2g(ie,id,i) = ck2g(ie,id,i) + vel(ie,1,id,i)*bn(i,ig)
c      & *rn(ijki)*sat(ijki,1)*rho(ijki,1)
34  continue
c
c      if(time.le.-1) then
c      write(3,*)
c      write(3,*) 'k2'
c      do 35 id = 1,ndmp
c      do 35 ie = 1,nelmax
c      write(3,1100) id,1,ie,(ck2g(ie,id,in),in = 1,nnmp)
c 35  continue
c      endif
c
c
    do 36 id = 1,ndmp
    do 36 jd = 1,ndmp
    do 36 ie = 1,nelmax
        ijkig = ijk(ie,ig)
        ck3g(ie,id,jd,i) = ck3g(ie,id,jd,i) + ck3(ijkig,id,jd)*bn(i,ig)
36  continue
c
c
c... derivative of saturation wrt capillary head ...
c
    do 40 is = 1,nsmp
    do 40 ia = 1,namp
c      write(3,*) ijkig,is,ia,wm(ijkig,is,ia)
        do 40 ja = 1,namp
        do 40 ie = 1,nelmax
            ijkig = ijk(ie,ig)
            shg(ie,is,ia,ja,i) = shg(ie,is,ia,ja,i)
c      & + rho(ijkig,ia)*wm(ijkig,is,ia)*sh(ijkig,ia,ja)*bn(i,ig)
40  continue
c
c
    do 45 ia = 1,namp
c      write(3,*) ijkig,ia,sat(ijkig,ia)
        do 45 ie = 1,nelmax
            ijkig = ijk(ie,ig)

```

```

          shmg(ie,ia,i) = shmg(ie,ia,i)
    &      + rn(ijkig)*rho(ijkig,ia)*sat(ijkig,ia)*bn(i,ig)
45 .    continue
    c
c... fluid conductivity ...
    c
      do 50 is = 1,nsmp
      do 50 ia = 1,namp
      do 50 id = 1,ndmp
      do 50 jd = 1,ndmp
      do 50 ie = 1,nelmax
      ijkig = ijk(ie,ig)
      rkg(ie,is,ia,id,jd,i) = rkg(ie,is,ia,id,jd,i)
    &      + rho(ijkig,ia)*wm(ijkig,is,ia)*rk(ijkig,ia,id,jd)*bn(i,ig)
50 .    continue
    c
      do 55 ia = 1,namp
      do 55 jd = 1,ndmp
      do 55 ie = 1,nelmax
      ijkig = ijk(ie,ig)
      rkmg(ie,is,jd,i) = rkmg(ie,ia,jd,i)
    &      + rho(ijkig,ia)*vel(ie,ia,jd,i)*bn(i,ig)
55 .    continue
    c
c... dispersive coefficients ...
    c
      do 60 is = 1,nsmp
      do 60 ia = 1,namp
      do 60 id = 1,ndmp
      do 60 jd = 1,ndmp
      do 60 ie = 1,nelmax
      ijkig = ijk(ie,ig)
      rdg(ie,is,ia,id,jd,i) = rdg(ie,is,ia,id,jd,i)
    &      + rd(ijkig,is,ia,id,jd)*bn(i,ig)
    &      *rn(ijkig)*sat(ijkig,ia)*rho(ijkig,ia)
60 .    continue
    c
30 .    continue
    c
c... debug ...
    c
    c    if(time.le.dt) then
    c      do 70 ie = 1,nelmax
    c      do 70 ii = 1,nnmp
    c      ijkig = ijk(ie,ii)
    c      do 80 is = 1,nsmp
    c      write(3,*) 'nel,is,ijkig = ',ie,is,ijkig

```

```

c      write(3,*) 'sh'
c      write(3,1000) ((sh(ijkig,ia,ja),ja = 1,namp),ia = 1,namp)
c      write(3,*) 'shg'
c      write(3,1000) ((shg(ie,is,ia,ja,ii),ja = 1,namp),ia = 1,namp)
c1000   format(3(/3c9.2))
c
c      do 90 ia = 1,namp
c        write(3,*) 'rk,ia = ',ia
c        write(3,2000) ((rk(ijkig,ia,id,jd),jd = 1,ndmp),id = 1,ndmp)
c        write(3,*) 'rkg,ia = ',ia
c        write(3,2000) ((rkg(ie,is,ia,id,jd,ii),jd = 1,ndmp),id = 1,ndmp)
c2000   format(3(/3c9.2))
c90     continue
c
c80     continue
c70     continue
c      endif
c
c20     continue
c
c
c      ***** computation of element matrices *****
c
c
c... es matrix ...
c
c      do 95 ig = 1,nnmp
c        do 95 i = 1,nnmp
c          do 95 j = 1,nnmp
c            do 95 ic = 1,nelmax
c              es(ie,i,j) = es(ie,i,j) + bn(i,ig)*bn(j,ig)*det(ie,ig)
95      continue
c
c      do 100 ig = 1,nnmp
c        do 100 i = 1,nnmp
c
c... etc matrix ...
c
c      do 103 j = 1,namp
c        do 103 ic = 1,nelmax
c          etc(ie,i,j) = etc(ie,i,j)
c          &      + (ck1g(ie,j))
c          &      *det(ie,ig)*bn(j,ig)*bn(i,ig)/dt
c          et(i,j) = etc(ie,i,j)
c          &      /((ck1g(ie,j))/det(ie,ig))
c          &      *dt
103     continue

```

```

c
c... ekc matrix ...
c
  do 105 j = 1,nnmp
  do 105 id = 1,ndmp
  do 105 ic = 1,nelmax
    ekc(ic,id,i,j) = ekc(ic,id,i,j)
c & + det(ic,ig)*bn(i,ig)
  & + ck2g(ic,id,j)*det(ic,ig)*w(ic,1,i,ig)
  & *dnr(ic,id,j,ig)
c & *dwr(ic,ia,id,j,ig)
c   ck(i,j) = ekc(ic,id,i,j)
c & /det(ic,ig)
croy write(3,*) 'ek = ',ek(i,j)
105 continue
c
c... edc matrix ...
c
  do 107 j = 1,nnmp
  do 107 id = 1,ndmp
  do 107 jd = 1,ndmp
  do 107 ic = 1,nelmax
    edc(ic,id,jd,i,j) = edc(ic,id,jd,i,j)
  & + ck3g(ic,id,jd,j)*det(ic,ig)*dnr(ic,jd,j,ig)
c & *dnr(ic,id,i,ig)
  & *dwr(ic,1,id,i,ig)
c   ed(i,j) = edc(ic,id,jd,i,j)
c & /ck3g(ic,id,jd,j)/det(ic,ig)
croy write(3,*) 'ed = ',ed(i,j)
107 continue
c
c
  do 100 ia = 1,namp
c
c... etm matrix ...
c
  do 102 j = 1,nnmp
  do 102 ie = 1,nelmax
    etm(ic,ia,i,j) = etm(ic,ia,i,j)
  & + (shmg(ic,ia,j))
  & *det(ic,ig)*bn(i,ig)*bn(j,ig)/dt
102 continue
c
c... ekm matrix ...
c
  do 104 j = 1,nnmp
  do 104 ie = 1,nelmax

```

```

      ekm(ic,ia,id,i,j) = ekm(ic,ia,id,i,j)
      &   + rkmg(ic,ia,id,j)*det(ic,ig)*bn(i,ig)
      &   *dnr(ic,id,j,ig)
c   &   *dwr(ic,ia,id,i,ig)
104  continue
c
      do 100 is = 1,nsmp
c
c   et matrix
c
      do 110 ie = 1,nelmax
      eth(ic,is,ia,i,i) = eth(ic,is,ia,i,i)
      &   + (shg(ic,is,1,ia,i) + shg(ic,is,2,ia,i) + shg(ic,is,3,ia,i))
      &   *det(ic,ig)*bn(i,ig)*bn(i,ig)/dt
110  continue
c
c
      do 100 j = 1,nnmp
      do 100 id = 1,ndmp
c
c... eg matrix ...
c
c4   eg(ic,is,ia,1,1,i,j) = eg(ic,is,ia,1,1,i,j) + rkg(ic,is,ia,1,1,j)*0
c31  eg(ic,is,ia,1,1,i,j) = eg(ic,is,ia,1,1,i,j) + rkg(ic,is,ia,1,1,j)
c32  eg(ic,is,ia,id,2,i,j) = eg(ic,is,ia,id,2,i,j) + rkg(ic,is,ia,id,2,j)
c2   eg(ic,is,ia,id,2,i,j) = eg(ic,is,ia,id,2,i,j) + rkg(ic,is,ia,id,2,j)
c
      do 120 ie = 1,nelmax
      egh(ic,is,ia,id,2,i,j) = egh(ic,is,ia,id,2,i,j) + rkg(ic,is,ia,id,2,j)
      &   *dnr(ic,id,i,ig)*rhor(ia)*bn(j,ig)*det(ic,ig)
c   &   *dnr(ic,id,i,ig)*rhor(ia)*bn(j,ig)*det(ic,ig)
c   &   *det(ic,ig)*dnr(ic,2,i,ig)*rhor(ia)
c   &   *det(ic,ig)*dwr(ic,ia,2,i,ig)*rhor(ia)
120  continue
c
      do 100 jd = 1,ndmp
c
c... ek matrix ...
c
      do 130 ie = 1,nelmax
      ekh(ic,is,ia,id,jd,i,j) = ekh(ic,is,ia,id,jd,i,j)
      &   + rkg(ic,is,ia,id,jd,j)*det(ic,ig)*dnr(ic,jd,j,ig)
      &   *dnr(ic,id,i,ig)
c   &   *dwr(ic,ia,id,i,ig)
c
c... ed matrix ...
c

```

```
    edm(ic,is,ia,id,jd,i,j) = edm(ic,is,ia,id,jd,i,j)
    &   +rdg(ic,is,ia,id,jd,j)*det(ic,ig)*dnr(ic,jd,j,ig)
    &   *dnr(ic,id,i,ig)
c   &   *dwr(ic,ia,id,i,ig)
130  continue
c
100  continue
c
c   write(3,*) 'et,ek,ed = ',et(1,1),ek(1,1),ed(1,1)
c
    return
    end
```



```

c
c
c
  subroutine basis1
c
c... this subroutine evaluates one dimensional basis functions
c and the jacobian matrix, its determinant and inverse ...
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2, nsm = 2, nam = 3, nnm = 4, npm = 112)
c
  dimension
&x(npm),y(npm),z(npm),ijk(npm,nnm),xc(nnm),yc(nnm),zc(nnm),
&cn(ndm,nnm),gp(ndm,nnm)
c
  dimension
&bn(nnm,nnm),bnd(ndm,nnm,nnm),bndr(ndm,nnm,nnm),
&bnr(ndm,nnm,nnm),dnr(npm,ndm,nnm,nnm),
&dj(nam,nam),dji(npm,nam,nam,nnm),det(npm,nnm),
&wdn(ndm,nnm,nnm),dwdn(ndm,nnm,nnm)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S      ipx,ipy,npx,ppy,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
S      pltime,dpltim,erc,wold,wnew
  common /space/ x,y,z
  common /bas/ bn,bnd,bndr,bnr,dnr,dji,det,wdn,dwdn
  common /points/ gp,cn
c
  do 10 ig = 1,2
    do 15 i = 1,2
c
c... directional component of basis function ...
c
      bnd(1,i,ig) = 0.5*(1.d0 + cn(1,i)*gp(1,ig))
c
c... basis function bn(i,ig) ...
c
      bn(i,ig) = bnd(1,i,ig)
.croy   write(3,*) 'bn = ',bn(i,ig)
c
c... derivative of directional component bndr = dnxidxi ...
c
      bndr(1,i,ig) = 0.5*cn(1,i)
.croy   write(3,*) 'bndr = ',bndr(1,i,ig)
c

```

```

c... derivative of basis function bnr = dndxi ...
c
c   * bnr(1,i,ig) = bndr(1,i,ig)
c
c... additional component for upstream weighting ...
c
c   wdn(1,i,ig) = 0.75*cn(1,i)*(1-gp(1,ig)**2)
c   dwdn(1,i,ig) = 1.5*cn(1,i)*(-gp(1,ig))
c
15  continue
c
c... element coordinates ...
c
c   do 20 nel = 1,nelmax
c
c   do 30 i = 1,2
c     ijki = ijk(nel,i)
c     xc(i) = x(ijki)
30  continue
c
c
c... compute the elements of jacobian matrix ...
c
c   do 40 i = 1,1
c     do 40 j = 1,1
40  dj(i,j) = 0.d0
c
c   do 50 i = 1,2
c     dj(1,1) = dj(1,1) + bnr(1,i,ig)*xc(i)
50  continue
c
c... compute the determinant of jacobian matrix ...
c
c   det(nel,ig) = dj(1,1)
croy write(3,*) 'det =',nel,ig,det(nel,ig)
c
c   if(det(nel,ig).eq.0) then
c     write(6,*) 'determinant 0, so stop'
c     stop
c   endif
c
c   rdet = 1.d0/det(nel,ig)
c
c... evaluate the inverse of the jacobian matrix ...
c
c   dji(nel,1,1,ig) = rdet
c

```

c... evaluate the derivative of global basis functions ...

```
c
  do*60 i = 1,2
    dnr(nel,1,i,ig) = dji(nel,1,1,ig)*bnr(1,i,ig)
  croy write(3,*) 'dnr = ',dnr(nel,1,i,ig)
60  continue
c
20  continue
c
10  continue
c
999 return
end
```

```

c
c
c
c
  subroutine basis2
c
c... this subroutine evaluates two dimensional basis functions and
c the jacobian matrix, its determinant and inverse ...
c
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npn = 112)
c
  dimension
&x(npn),y(npn),z(npn),ijk(npn,nnm),xe(nnm),ye(nnm),ze(nnm)
c
  dimension cn(ndm,nnm),gp(ndm,nnm)
c
  dimension
&bn(nnm,nnm),bnd(ndm,nnm,nnm),bndr(ndm,nnm,nnm),
&bnr(ndm,nnm,nnm),
&dnr(npn,ndm,nnm,nnm),wdn(ndm,nnm,nnm),dwdn(ndm,nnm,nnm),
&dj(nam,nam),dji(npn,nam,nam,nnm),det(npn,nnm)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S      ipx,ipy,ipx,ipy,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,ep,axi,acta,rgr,ptime,dptime,
S      pltime,dpltim,erc,wold,wnew
  common /space/ x,y,z
  common /bas/ bn,bnd,bndr,bnr,dnr,dji,det,wdn,dwdn
  common /points/ gp,cn
c
  do 10 ig = 1,4
    do 15 i = 1,4
c
c... directional component of basis function ...
c
      bnd(1,i,ig) = 0.5*(1.d0 + cn(1,i)*gp(1,ig))
      bnd(2,i,ig) = 0.5*(1.d0 + cn(2,i)*gp(2,ig))
c
c... basis function bn(i,ig) ...
c
      bn(i,ig) = bnd(1,i,ig)*bnd(2,i,ig)
c
c... derivative of directional component bndr = dnaxidxi ...
c
      bndr(1,i,ig) = 0.5*cn(1,i)

```

```

        bndr(2,i,ig) = 0.5*cn(2,i)
c
c... derivative of basis function bnr = dndxi ...
        bnr(1,i,ig) = bndr(1,i,ig)*bnd(2,i,ig)
        bnr(2,i,ig) = bndr(2,i,ig)*bnd(1,i,ig)
c
c... additional component for upstream weighting ...
c
        wdn(1,i,ig) = 0.75*cn(1,i)*(1-gp(1,ig)**2)
        wdn(2,i,ig) = 0.75*cn(2,i)*(1-gp(2,ig)**2)
        dwdn(1,i,ig) = 1.5*cn(1,i)*(-gp(1,ig))
        dwdn(2,i,ig) = 1.5*cn(2,i)*(-gp(2,ig))
c
15  continue
c
c... element coordinates ...
c
        do 20 nel = 1,nelmax
c
        do 30 i = 1,4
            ijki = ijk(nel,i)
            xc(i) = x(ijki)
            yc(i) = y(ijki)
30  continue
c
c
c... compute the elements of jacobian matrix ...
c
        do 40 i = 1,2
            do 40 j = 1,2
40         dj(i,j) = 0.d0
c
c
        do 50 i = 1,4
            dj(1,1) = dj(1,1) + bnr(1,i,ig)*xc(i)
            dj(1,2) = dj(1,2) + bnr(1,i,ig)*yc(i)
            dj(2,1) = dj(2,1) + bnr(2,i,ig)*xc(i)
            dj(2,2) = dj(2,2) + bnr(2,i,ig)*yc(i)
50  continue
c
c... compute the determinant of jacobian matrix ...
c
        det(nel,ig) = dj(1,1)*dj(2,2)-dj(1,2)*dj(2,1)
c        write(3,*) 'nel,ig,det = ',nel,ig,det(nel,ig)
c
c... debug ...
c

```

```

c  if(time.eq.-1) then
c  write(3,*) 'time,nel,ig,det,dj,xc,yc'
c  write(3,1000) time,nel,ig,det(nel,ig),
c  &          dj(1,1),dj(1,2),dj(2,1),dj(2,2),
c  &          xc(1),yc(1),xc(2),yc(2),xc(3),yc(3),xc(4),yc(4)
c000 format(/f10.4,2i5,f10.4/4f10.4/8f10.4)
c  endif
c
c
c  if(det(nel,ig).eq.0) then
c    write(3,*) 'determinant 0, so stop'
c    stop
c  endif
c
c  rdet = 1.d0/det(nel,ig)
c
c... evaluate the inverse of the jacobian matrix ...
c
c  dji(nel,1,1,ig) = rdet*dj(2,2)
c  dji(nel,1,2,ig) = -rdet*dj(1,2)
c  dji(nel,2,1,ig) = -rdet*dj(2,1)
c  dji(nel,2,2,ig) = rdet*dj(1,1)
c
c... evaluate the derivative of global basis functions ...
c
c  do 60 i = 1,4
c    dnr(nel,1,i,ig) = dji(nel,1,1,ig)*bnr(1,i,ig)
c    &          + dji(nel,1,2,ig)*bnr(2,i,ig)
c    dnr(nel,2,i,ig) = dji(nel,2,1,ig)*bnr(1,i,ig)
c    &          + dji(nel,2,2,ig)*bnr(2,i,ig)
60  continue
c
20  continue
c
10  continue
c
c  999  return
c      end

```

```

c
c
c
c
  subroutine basis3
c
c... this subroutine evaluates three dimensional basis functions
c   and the jacobian matrix, its determinant and inverse ...
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension
&x(npm),y(npm),z(npm),ijk(npm,nnm),xe(nnm),ye(nnm),ze(nnm)
c
  dimension cn(ndm,nnm),gp(ndm,nnm)
c
  dimension
&bn(nnm,nnm),bnd(ndm,nnm,nnm),bndr(ndm,nnm,nnm),
&bnr(ndm,nnm,nnm),
&dnr(npm,ndm,nnm,nnm),wdn(ndm,nnm,nnm),dwdn(ndm,nnm,nnm),
&dj(nam,nam),dji(npm,nam,nam,nnm),det(npm,nnm)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S          ipx,ipy,npx,npy,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,cp,axi,acta,rgr,ptime,dptime,
S          pltime,dpltim,erc,wold,wnew
  common /space/ x,y,z
  common /bas/ bn,bnd,bndr,bnr,dnr,dji,det,wdn,dwdn
  common /points/ gp,cn
c
  do 10 ig = 1,8
    do 15 i = 1,8
c
c... directional component of basis function ...
c
      bnd(1,i,ig) = 0.5*(1.d0 + cn(1,i)*gp(1,ig))
      bnd(2,i,ig) = 0.5*(1.d0 + cn(2,i)*gp(2,ig))
      bnd(3,i,ig) = 0.5*(1.d0 + cn(3,i)*gp(3,ig))
c
c... basis function bn(i,ig) ...
c
      bn(i,ig) = bnd(1,i,ig)*bnd(2,i,ig)*bnd(3,i,ig)
c
c... derivative of directional component bndr = dnxidxi ...
c
      bndr(1,i,ig) = 0.5*cn(1,i)

```

```

        bndr(2,i,ig) = 0.5*cn(2,i)
        bndr(3,i,ig) = 0.5*cn(3,i)
c
c... derivative of basis function bnr = dndxi ...
c
        bnr(1,i,ig) = bndr(1,i,ig)*bnd(2,i,ig)*bnd(3,i,ig)
        bnr(2,i,ig) = bnd(1,i,ig)*bndr(2,i,ig)*bnd(3,i,ig)
        bnr(3,i,ig) = bnd(1,i,ig)*bnd(2,i,ig)*bndr(3,i,ig)
c
c... additional component for upstream weighting ...
c
        wdn(1,i,ig) = 0.75*cn(1,i)*(1-gp(1,ig)**2)
        wdn(2,i,ig) = 0.75*cn(2,i)*(1-gp(2,ig)**2)
        wdn(3,i,ig) = 0.75*cn(3,i)*(1-gp(3,ig)**2)
        dwdn(1,i,ig) = 1.5*cn(1,i)*(-gp(1,ig))
        dwdn(2,i,ig) = 1.5*cn(2,i)*(-gp(2,ig))
        dwdn(3,i,ig) = 1.5*cn(3,i)*(-gp(3,ig))
c
15  continue
c
c
c... element coordinates ...
c
        do 20 nel = 1,nelmax
c
        do 30 i = 1,8
            ijki = ijk(nel,i)
            xc(i) = x(ijki)
            ye(i) = y(ijki)
            ze(i) = z(ijki)
30  continue
c
c
c... compute the elements of jacobian matrix ...
c
        do 40 i = 1,3
            do 40 j = 1,3
40          dj(i,j) = 0.d0
c
c
        do 50 i = 1,8
            dj(1,1) = dj(1,1) + bnr(1,i,ig)*xc(i)
            dj(1,2) = dj(1,2) + bnr(1,i,ig)*ye(i)
            dj(1,3) = dj(1,3) + bnr(1,i,ig)*ze(i)
            dj(2,1) = dj(2,1) + bnr(2,i,ig)*xc(i)
            dj(2,2) = dj(2,2) + bnr(2,i,ig)*ye(i)
            dj(2,3) = dj(2,3) + bnr(2,i,ig)*ze(i)

```



```

    dj(3,1) = dj(3,1) + bnr(3,i,ig)*xe(i)
    dj(3,2) = dj(3,2) + bnr(3,i,ig)*ye(i)
    dj(3,3) = dj(3,3) + bnr(3,i,ig)*ze(i)
50  continue
c
c... compute the determinant of jacobian matrix ...
c
    det(nel,ig) = dj(1,1)*(dj(2,2)*dj(3,3)-dj(2,3)*dj(3,2))
&              -dj(1,2)*(dj(2,1)*dj(3,3)-dj(2,3)*dj(3,1))
&              +dj(1,3)*(dj(2,1)*dj(3,2)-dj(2,2)*dj(3,1))
c
    if(det(nel,ig).eq.0) then
        write(6,*) 'determinant 0, so stop'
        stop
    endif
c
    rdet = 1.d0/det(nel,ig)
c
c... evaluate the inverse of the jacobian matrix ...
c
    dji(nel,1,1,ig) = rdet*(dj(2,2)*dj(3,3)-dj(2,3)*dj(3,2))
    dji(nel,1,2,ig) = -rdet*(dj(1,2)*dj(3,3)-dj(1,3)*dj(3,2))
    dji(nel,1,3,ig) = rdet*(dj(1,2)*dj(2,3)-dj(1,3)*dj(2,2))
    dji(nel,2,1,ig) = -rdet*(dj(2,1)*dj(3,3)-dj(2,3)*dj(3,1))
    dji(nel,2,2,ig) = rdet*(dj(1,1)*dj(3,3)-dj(1,3)*dj(3,1))
    dji(nel,2,3,ig) = -rdet*(dj(1,1)*dj(2,3)-dj(1,3)*dj(2,1))
    dji(nel,3,1,ig) = rdet*(dj(2,1)*dj(3,2)-dj(2,2)*dj(3,1))
    dji(nel,3,2,ig) = -rdet*(dj(1,1)*dj(3,2)-dj(1,2)*dj(3,1))
    dji(nel,3,3,ig) = rdet*(dj(1,1)*dj(2,2)-dj(1,2)*dj(2,1))
c
c... evaluate the derivative of global basis functions ...
c
    do 60 i = 1,nnmp
        dnr(nel,1,i,ig) = dji(nel,1,1,ig)*bnr(1,i,ig)
&                  + dji(nel,1,2,ig)*bnr(2,i,ig)
&                  + dji(nel,1,3,ig)*bnr(3,i,ig)
        dnr(nel,2,i,ig) = dji(nel,2,1,ig)*bnr(1,i,ig)
&                  + dji(nel,2,2,ig)*bnr(2,i,ig)
&                  + dji(nel,2,3,ig)*bnr(3,i,ig)
        dnr(nel,3,i,ig) = dji(nel,3,1,ig)*bnr(1,i,ig)
&                  + dji(nel,3,2,ig)*bnr(2,i,ig)
&                  + dji(nel,3,3,ig)*bnr(3,i,ig)
60  continue
c
20  continue
c
10  continue

```

c
999 return
end

```

c
c
c
  subroutine weigh1
c
c... this subroutine evaluates one dimensional weighting function
c and its derivative ...
c
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension
  &ijk(npm,nnm),weif(npm,nam,ndm,nnm),
  &w(npm,nam,nnm,nnm),wd(nam,ndm,nnm,nnm),
  &wdr(nam,ndm,nnm,nnm),dwr(npm,nam,ndm,nnm,nnm)
c
  dimension
  &bn(nnm,nnm),bnd(ndm,nnm,nnm),bndr(ndm,nnm,nnm),
  &bnr(ndm,nnm,nnm),
  &dnr(npm,ndm,nnm,nnm),dj(nam,nam),dji(npm,nam,nam,nnm),
  &wdn(ndm,nnm,nnm),dwdn(ndm,nnm,nnm),det(npm,nnm)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
  S      ipx,ipy,npx,npy,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,ep,axi,acta,rgr,ptime,dptime,
  S      pltime,dpltim,erc,wold,wnew
  common /bas/ bn,bnd,bndr,bnr,dnr,dji,det,wdn,dwdn
  common /wei/ w,dwr,weif
c
c
c... evaluate the local weighting functions ...
c
c  write(3,*) 'wxi(i),w(i)'
c
  do 10 ia = 1,namp
  do 10 i = 1,2
  do 10 ig = 1,2
  do 10 nel = 1,nelmax
c    wd(ia,1,i,ig) = bnd(1,i,ig)-0.6*wdn(1,i,ig)
    wd(ia,1,i,ig) = bnd(1,i,ig) + weif(nel,ia,1,i)*wdn(1,i,ig)
    w(nel,ia,i,ig) = wd(ia,1,i,ig)
c
c    write(3,100) wdn(1,i,ig),dwdn(1,i,ig),weif(nel,ia,1,i)
c100  format('wdn,dwdn,weif = ',3e10.3)
c
c... evaluate the derivatives of the local weighting functions ...

```

```

c
c   write(3,*) 'dwdxi(i)'
c
c   wdr(ia,l,i,ig) = bnr(l,i,ig)-0.6*dwdn(l,i,ig)
c   wdr(ia,l,i,ig) = bnr(l,i,ig) + wcif(nel,ia,l,i)*dwdn(l,i,ig)
c
c... evaluate the derivatives of the global weighting functions ...
c
c   dwr(nel,ia,l,i,ig) = dji(nel,l,l,ig)*wdr(ia,l,i,ig)
c
10  continue
c
c
c   return
c   end

```

```

c
c
c
  subroutine weigh2
c
c... this subroutine evaluates two dimensional weighting functions
c   and their derivative ...
c
c
c   implicit real*8 (a-h,o-z)
c   parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c   dimension
c   &ijk(npm,nnm),weif(npm,nam,ndm,nnm),
c   &w(npm,nam,nnm,nnm),wd(nam,ndm,nnm,nnm),
c   &wdr(nam,ndm,nnm,nnm),dwr(npm,nam,ndm,nnm,nnm)
c
c   dimension
c   &bn(nnm,nnm),bnd(ndm,nnm,nnm),bndr(ndm,nnm,nnm),
c   &bnr(ndm,nnm,nnm),
c   &dnr(npm,ndm,nnm,nnm),dj(nam,nam),dji(npm,nam,nam,nnm),
c   &wdn(ndm,nnm,nnm),dwdn(ndm,nnm,nnm),det(npm,nnm)
c
c   common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
c   S      ipx,ipy,npx,npy,ndmp,nsmp,namp,nnmp
c   common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
c   S      pltime,dpltim,erc,wold,wnew
c   common /bas/ bn,bnd,bndr,bnr,dnr,dji,det,wdn,dwdn
c   common /wei/ w,dwr,weif
c
c
c... evaluate the local weighting functions ...
c
c   write(3,*) 'wxi(i),weta(i),w(i)'
c
c   do 10 ia = 1,namp
c   do 10 i = 1,4
c   do 10 ig = 1,4
c   do 10 nel = 1,nelmax
c     wd(ia,1,i,ig) = bnd(1,i,ig) + weif(nel,ia,1,i)*wdn(1,i,ig)
c     wd(ia,2,i,ig) = bnd(2,i,ig) + weif(nel,ia,2,i)*wdn(2,i,ig)
c     w(nel,ia,i,ig) = wd(ia,1,i,ig)*wd(ia,2,i,ig)
c
c   write(3,100) wd(ia,1,i,ig),wd(ia,2,i,ig),w(nel,ia,i,ig)
c100  format(3e10.3)
c
c... evaluate the derivatives of the local weighting functions ...

```

```

c
c   write(3,*) 'dwdx(i),dwdeta(i)'
c
   wdr(ia,1,i,ig) = bnd(2,i,ig)*(bnr(1,i,ig) +
&      weif(nel,ia,1,i)*dwdn(1,i,ig))
   wdr(ia,2,i,ig) = bnd(1,i,ig)*(bnr(2,i,ig) +
&      weif(nel,ia,2,i)*dwdn(2,i,ig))
c
c... evaluate the derivatives of the global weighting functions ...
c
   dwr(nel,ia,1,i,ig) = dji(nel,1,1,ig)*wdr(ia,1,i,ig)
&      + dji(nel,1,2,ig)*wdr(ia,2,i,ig)
   dwr(nel,ia,2,i,ig) = dji(nel,2,1,ig)*wdr(ia,1,i,ig)
&      + dji(nel,2,2,ig)*wdr(ia,2,i,ig)
c
10  continue
c
c
   return
   end

```

```

c
c
c
  subroutine weigh3
c
c... this subroutine evaluates three dimensional weighting function
c and their derivatives ...
c
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension
    &ijk(npm,nnm),weif(npm,nam,ndm,nnm),
    &w(npm,nam,nnm,nnm),wd(nam,ndm,nnm,nnm),
    &wdr(nam,ndm,nnm,nnm),dwr(npm,nam,ndm,nnm,nnm)
c
  dimension
    &bn(nnm,nnm),bnd(ndm,nnm,nnm),bndr(ndm,nnm,nnm),
    &bnr(ndm,nnm,nnm),
    &dnr(npm,ndm,nnm,nnm),dji(npm,nam,nam,nnm),
    &wdn(ndm,nnm,nnm),dwdn(ndm,nnm,nnm),det(npm,nnm)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
  S          ipx,ipy,npx,npy,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,ep,axi,acta,rgr,ptime,dptime,
  S          pltime,dpltim,erc,wold,wnew
  common /bas/ bn,bnd,bndr,bnr,dnr,dji,det,wdn,dwdn
  common /wei/ w,dwr,weif
c
c
c... evaluate the local weighting functions ...
c
c  write(3,*) 'wxi(i),weta(i),w(i)'
c
  do 10 ia = 1,namp
  do 10 i = 1,8
  do 10 ig = 1,8
  do 10 nel = 1,nelmax
    wd(ia,1,i,ig) = bnd(1,i,ig) + weif(nel,ia,1,i)*wdn(1,i,ig)
    wd(ia,2,i,ig) = bnd(2,i,ig) + weif(nel,ia,2,i)*wdn(2,i,ig)
    wd(ia,3,i,ig) = bnd(3,i,ig) + weif(nel,ia,3,i)*wdn(3,i,ig)
    w(nel,ia,i,ig) = wd(ia,1,i,ig)*wd(ia,2,i,ig)*wd(ia,3,i,ig)
c
c  write(3,100) wd(ia,1,i,ig),wd(ia,2,i,ig),w(nel,ia,i,ig)
c100  format(3c10.3)
c

```

```

c... evaluate the derivatives of the local weighting functions ...
c
c   write(3,*) 'dwdx(i),dwdeta(i)'
c
   wdr(ia,1,i,ig) = bnd(2,i,ig)*bnd(3,i,ig)*(bnr(1,i,ig) +
&               weif(nel,ia,1,i)*dwdn(1,i,ig))
   wdr(ia,2,i,ig) = bnd(1,i,ig)*bnd(3,i,ig)*(bnr(2,i,ig) +
&               weif(nel,ia,2,i)*dwdn(2,i,ig))
   wdr(ia,3,i,ig) = bnd(1,i,ig)*bnd(2,i,ig)*(bnr(3,i,ig) +
&               weif(nel,ia,3,i)*dwdn(3,i,ig))
c
c... evaluate the derivatives of the global weighting functions ...
c
   dwr(nel,ia,1,i,ig) = dji(nel,1,1,ig)*wdr(ia,1,i,ig)
&               + dji(nel,1,2,ig)*wdr(ia,2,i,ig)
&               + dji(nel,1,3,ig)*wdr(ia,3,i,ig)
   dwr(nel,ia,2,i,ig) = dji(nel,2,1,ig)*wdr(ia,1,i,ig)
&               + dji(nel,2,2,ig)*wdr(ia,2,i,ig)
&               + dji(nel,2,3,ig)*wdr(ia,3,i,ig)
   dwr(nel,ia,3,i,ig) = dji(nel,3,1,ig)*wdr(ia,1,i,ig)
&               + dji(nel,3,2,ig)*wdr(ia,2,i,ig)
&               + dji(nel,3,3,ig)*wdr(ia,3,i,ig)
c
10  continue
c
c   return
   end

```



```

c
c
c
  subroutine bound1(agh,fh,is,ia)
c
c... this subroutine evaluates first-type boundary conditions
c . about pressure head ...
c
c   < h = hbel(i) >
c
c
c   nbel      : total number of nodes on 1st boundary
c   noeb1(i)  : node number of 1st boundary
c   hbel(i)   : value of head on boundary point noeb1(i)
c   penalt    : penalty value
c
c
c   implicit real*8 (a-h,o-z)
c   parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c   dimension
c   &agh(npm,45,nsm,nam),fh(npm,nsm,nam),ijk(npm,nnm),
c   &noeb1(npm,nam),hbel(npm,nam),nbel(nam),h(npm,nam)
c
c   common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
c   S          ipx,ipy,npx,upy,ndmp,nsmp,namp,nnmp
c
c   common /ib1/ nbel,noeb1
c   common /b1/ penalt,hbel
c
c
c   do 10 ip = 1,nbel(ia)
c     j = noeb1(ip,is)
c     agh(j,ihalfb + 1,is,ia) = agh(j,ihalfb + 1,is,ia) + penalt
c     fh(j,is,ia) = fh(j,is,ia) + penalt*hbel(ip,ia)
c     write(3,*) 'hbc,i = ',hbel(ip,ia),ip
c 10  continue
c
c   write(3,*) 'hbel(1,1),agh(2,2,1,18) = ',hbel(1,1),agh(2,2,1,18)
c
c   return
c   end

```

```

c
c
c
  subroutine bouncl(agg,fc,sat)
c
c... this subroutine evaluates first-type boundary conditions
c   for mass fraction ...
c
c
c   < wm = wmbel(i) >
c
c
c   penalt      : penalty value
c
c
c   implicit real*8 (a-h,o-z)
c   parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c   dimension
c   &agg(npm,45),fc(npm),ijk(npm,nnm),c(npm),oldc(npm),
c   &wm(npm,nsm,4),oldwm(npm,nsm,4),sat(npm,nam),
c   &rn(npm),rho(npm,4),rhor(nam)
c
c   common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
c   S           ipx,ipy,npx,npy,ndmp,nsmp,namp,nnmp
c
c   common /part/ wm,oldwm
c   common /part2/ c,oldc
c   common /poro/ rn
c   common /dens/ rho,rhor
c
c   hwoo = 1.1012e-3
c   sro = 0.30
c   penalt = 100000000.
c
c   do 10 ip = 1,npmax
c     if(sat(ip,2).gt.sro) then
c       wm(ip,2,1) = wm(ip,2,2)*hwoo
c       c(ip) = wm(ip,2,1)
c1    c(ip) = wm(ip,2,1)*rn(ip)*sat(ip,1)*rho(ip,1)
c       agg(ip,ihalfb + 1) = agg(ip,ihalfb + 1) + penalt
c       fc(ip) = fc(ip) + penalt*c(ip)
c     write(3,*) 'fc,c,i = ',fc(j),c(ip),ip
c     endif
c   10 continue
c
c   return

```

end

```

c
c
c
  subroutine boun12(fh,is,ia)
c
c... this subroutine evaluates one dimensional second-type
c boundary conditions ...
c
c
c   < dh/dx = hbc2(i) >
c
c
c   nbe2      : total number of nodes on 2nd boundary
c   noeb2(i)  : node number of 2nd boundary
c   hbc2(i)   : value of flux on noeb2(i)
c
c
c   implicit real*8 (a-h,o-z)
c   parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c   dimension
c   &fh(npm,nsm,nam),ijk(npm,nnm),x(npm),y(npm),z(npm),
c   &noeb2(npm,nam,nnm),hbc2(npm,nam,nnm),nbe2(nam)
c
c   common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
c   S          ipx,ipy,npx,npy,ndmp,nsmp,namp,nnmp
c   common /set1/ tmax,time,dt,ep,axi,acta,rgr,ptime,dptime,
c   S          pltime,dpltim,erc,wold,wnew
c   common /space/ x,y,z
c
c   common /ib2/ nbe2,noeb2
c   common /b2/  hbc2
c
c
c   write(3,*) fh(noeb2(ip,ia,1),is,ia)
c   do 10 ip = 1,nbe2(ia)
c     fh(noeb2(ip,ia,1),is,ia) = fh(noeb2(ip,ia,1),is,ia) + hbc2(ip,ia,1)
10  continue
c   write(3,*) fh(noeb2(ip,ia,1),is,ia)
c
c   return
c   end

```

```

c
c
c
  subroutine boun22(fh,is,ia)
c
c... this subroutine evaluates two dimensional second-type
c boundary conditions ...
c
c
c   < dh/dx = hbe2(i) >
c
c
c   nbe2      : total number of nodes on 2nd boundary
c   noeb2(i)  : node number of 2nd boundary
c   hbe2(i)   : value of flux on noeb2(i)
c
c
c   implicit real*8 (a-h,o-z)
c   parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npn = 112)
c
c   dimension
c   &fh(npn,nsm,nam),ijk(npn,nnm),x(npn),y(npn),z(npn),
c   &noeb2(npn,nam,nnm),hbe2(npn,nam,nnm),nbe2(nam)
c
c   common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
c   S           ipx,ipy,npn,npj,ndmp,nsmp,namp,nnmp
c   common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
c   S           pltime,dpltim,erc,wold,wnew
c   common /space/ x,y,z
c
c   common /ib2/ nbe2,noeb2
c   common /b2/  hbe2
c
c
c   do 10 ip = 1,nbe2(ia)
c     db = dsqrt((x(noeb2(ip,ia,1))-x(noeb2(ip,ia,2)))**2
c   &  + (y(noeb2(ip,ia,1))-y(noeb2(ip,ia,2)))**2)
c     dh = 0.5d0*(hbe2(ip,ia,1) + hbe2(ip,ia,2))
c     fh(noeb2(ip,ia,1),is,ia) = fh(noeb2(ip,ia,1),is,ia) + 0.5d0*db*dh
c     fh(noeb2(ip,ia,2),is,ia) = fh(noeb2(ip,ia,2),is,ia) + 0.5d0*db*dh
10  continue
c
c   return
c   end

```

```

c
c
c
  subroutine boun32(fh,is,ia)
c
c... this subroutine evaluates three dimensional second-type
c boundary conditions ...
c
c
c
c   < dh/dx = hbe2(i) >
c
c   nbe2      : total number of nodes on 2nd boundary
c   noeb2(i)  : node number of 2nd boundary
c   hbe2(i)   : value of flux on noeb2(i)
c
c
c   implicit real*8 (a-h,o-z)
c   parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c   dimension
c   &fh(npm,nsm,nam),ijk(npm,nnm),x(npm),y(npm),z(npm),
c   &noeb2(npm,nam,nnm),hbe2(npm,nam,nnm),nbe2(nam)
c
c   common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
c   S          ipx,ipy,npx,ppy,ndmp,nsmp,namp,nnmp
c   common /set1/ tmax,time,dt,ep,axi,acta,rgr,ptime,dptime,
c   S          pltime,dpltim,erc,wold,wnew
c   common /space/ x,y,z
c
c   common /ib2/ nbe2,noeb2
c   common /b2/  hbe2
c
c
c   do 10 ip = 1,nbe2(ia)
c     db = dabs((x(noeb2(ip,ia,1))-x(noeb2(ip,ia,2)))
c   & *(y(noeb2(ip,ia,1))-y(noeb2(ip,ia,4))))
c     dh = 0.25d0*(hbe2(ip,ia,1) + hbe2(ip,ia,2)
c   &      + hbe2(ip,ia,3) + hbe2(ip,ia,4))
c     fh(noeb2(ip,ia,1),is,ia) = fh(noeb2(ip,ia,1),is,ia) + 0.25d0*db*dh
c     fh(noeb2(ip,ia,2),is,ia) = fh(noeb2(ip,ia,2),is,ia) + 0.25d0*db*dh
c     fh(noeb2(ip,ia,3),is,ia) = fh(noeb2(ip,ia,3),is,ia) + 0.25d0*db*dh
c     fh(noeb2(ip,ia,4),is,ia) = fh(noeb2(ip,ia,4),is,ia) + 0.25d0*db*dh
10  continue
c
c   return

```

end

```

c
c
c
c
c      subroutine solve(kkk,b,r,neq,ihalfb)
c
c... this subroutine solves asymmetric band matrix
c      using Doolittle method ...
c
c
c
c      kkk = 1 triangularizes the band matrix b
c      kkk = 2 solves for right side r, solution returns in r
c
c      implicit real*8 (a-h,o-z)
c      parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
c      dimension b(npm,45),r(npm)
c
c      nrs = neq-1
c      ihbp = ihalfb + 1
c      if (kkk.eq.2) go to 30
c
c... triangularize matrix using doolittle method ...
c
c      do 10 k = 1,nrs
c      pivot = b(k,ihbp)
c      kk = k + 1
c      kc = ihbp
c      do 20 i = kk,neq
c      kc = kc-1
c      if(kc.le.0) go to 10
c      c = -b(i,kc)/pivot
c      b(i,kc) = c
c      ki = kc + 1
c      lim = kc + ihalfb
c      do 20 j = ki,lim
c      jc = ihbp + j-kc
c      write(3,*) 'b(',i,',',j,') = ',b(i,j),',c = ',c,',b(k,jc)',b(j,kc)
c      20  b(i,j) = b(i,j) + c*b(k,jc)
c      10  continue
c      go to 100
c
c... modify load vector r ...
c
c      30  nn = neq + 1
c      iband = 2*ihalfb + 1

```



```

do 40 i = 2, neq
  jc = ihbp - i + 1
  ji = 1
  if (jc.le.0) go to 50
  go to 60
50  jc = 1
   ji = i - ihbp + 1
60  sum = 0.0
   do 70 j = jc, ihalfb
     sum = sum + b(i, j) * r(ji)
70  ji = ji + 1
40  r(i) = r(i) + sum
c
c... back solution ...
c
  r(neq) = r(neq) / b(neq, ihbp)
  do 80 iback = 2, neq
    i = nn - iback
    jp = i
    kr = ihbp + 1
    mr = min0(iband, ihalfb + iback)
    sum = 0.0
    do 90 j = kr, mr
      jp = jp + 1
90  sum = sum + b(i, j) * r(jp)
80  r(i) = (r(i) - sum) / b(i, ihbp)
c
100 return
end

```

```

c
c
c
  subroutine output(h,sat)
c
c... this subroutine prints out results of simulation ...
c
c
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension h(npm,nam),sat(npm,nam),ijk(npm,nnm)
  dimension wm(npm,nsm,4),oldwm(npm,nsm,4),wmp(npm,nam)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S      ipx,ipy,npz,ndmp,nsmp,namp,nnmp
  common /set1/ tmax,time,dt,ep,axi,aeta,rgr,ptime,dptime,
S      pltime,dpltim,erc,wold,wnew
  common /part/ wm,oldwm
c
  write(2,1000) time
  write(9,1000) time
  write(10,1000) time
c
c
  do 10 is = 1,nsmp
c
  write(6 + is,1000) time
c
c... pressure head ...
c
  write(2,*) 'pressure'
  write(2,*) 'is = ',is
  call outp(h,is,2)
c
c... saturation ...
c
  write(2,*) 'saturation'
  call outp(sat,is,2)
c
c... plot oil saturation in file *.plt ...
c
  call outp(sat,is,6 + is)
c
10  continue
c

```

```

c... plot mass fraction of tce species in water phase ...
c
  do 20 ip = 1,npmax
    wmp(ip,2) = wm(ip,2,1)*1000
  20 continue
    call outp(wmp,2,9)
c
c... plot concentration of tce species in water phase
c over elementary volume ...
c
  do 30 ip = 1,npmax
    wmp(ip,2) = sat(ip,1)*wm(ip,2,1)*1000
  30 continue
    call outp(wmp,2,10)
c
1000 format(/t10,'time = ',f10.4/)
c
  return
end

```

```

c
c
c
  subroutine outp(h,ia,iw)
c
c... this subroutine plot results ...
c
c
c
  implicit real*8 (a-h,o-z)
  parameter (ndm = 2,nsm = 2,nam = 3,nnm = 4,npm = 112)
c
  dimension h(npm,nam),px(41,41),py(41,41),ph(41,41),ijk(npm,nnm)
c
  common /iset/ nelmax,npmax,ijk,ihalfb,iband,imax,
S      ipx,ipy,npx,npj,ndmp,nsmp,namp,nnmp
  common /axis/ px,py
c
  kk = 0
  do 10 i = 1,npx
    do 10 j = 1,npj
      kk = kk + 1
10    ph(i,j) = h(kk,ia)
c
  write(iw,*) 'iz = 1'
  write(iw,1000) (px(i,1),i = 1,npx,ipx)
  do 20 j = npy,1,ipy
20  write(iw,2000) py(1,j),(ph(i,j),i = 1,npx,ipx)
c
  if(ndmp.eq.3) then
  do 30 i = 1,npx
    do 30 j = 1,npj
      kk = kk + 1
30    ph(i,j) = h(kk,ia)
c
  write(iw,*) 'iz = 2'
  write(iw,1000) (px(i,1),i = 1,npx,ipx)
  do 40 j = npy,1,ipy
40  write(iw,2000) py(1,j),(ph(i,j),i = 1,npx,ipx)
  endif
c
1000 format(t4,'y/x',t8,22f7.2)
2000 format(22f7.2)
c
  return
  end
//go.ft01f001 dd dsn = igpckim.cmgm.dat,disp = old

```

```
//go.ft02f001 dd dsn = igpckim.cmgm.out,disp = old  
//go.ft03f001 dd dsn = igpckim.cmgm.deb,disp = old  
//go.ft07f001 dd dsn = igpckim.cmgm.pl1,disp = old  
//go.ft08f001 dd dsn = igpckim.cmgm.pl2,disp = old  
//go.ft09f001 dd dsn = igpckim.cmgm.pl3,disp = old  
//go.ft10f001 dd dsn = igpckim.cmgm.pl4,disp = old
```