T-2716

NUMERICAL SIMULATION OF PHASE SEGREGATION IN THE PRIMARY POROSITY (MATRIX BLOCKS) IN TWO-POROSITY POROUS MEDIA

by

J. R. Gilman

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

T-2716

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Chemical and Petroleum-Refining Engineering).

Golden, Colorado
Date 4/11/83

Signed: James R. Gilman
James R. Gilman

Approved: M. C. Jones
Dr. M. C. Jones

Golden, Colorado
Date 4/11/83

P. F. Dickson
Dr. P. F. Dickson, Head
Chemical and Petroleum
Refining Department

Abstract

An efficient finite different method is presented for simulating unsteady state multiphase flow in the matrix blocks (primary porosity) of naturally fractured systems or other reservoirs which have two different porosities. The scheme obtains pressure and saturation distributions in the matrix blocks (rather than average values) by dividing the matrix into sub-domains. The method is an extension of the double-porosity concept in which fractures are the continuum for fluid flow and the matrix rock is the primary storage medium which acts as a source term to the fractures.

The fracture medium is the major flow path to the well-bore with the matrix sub-domains as source terms. The matrix sub-domains can be connected to each other or to the fractures in any way desired to conform to the physics of the reservoir. This approach lends itself to an efficient scheme for reducing the system of flow equations to the same structure as conventional single-porosity simulators. The method is presented for a fully-implicit, two-phase (oil-water) simulator, but it can readily be extended to multi-phase, multidimensional systems as well as to problems involving heat flow or chemical transport.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank the management of Marathon Oil for permission to publish this paper and for the use of Marathon facilities in completing this work. I would also like to thank Kathy England for typing my thesis.

A special thanks goes to my wife, Anita, for bearing with me while completing my degree and for the help in preparing figures for this thesis.

Introduction

Numerical simulation of petroleum reservoirs has proved to be a very useful tool for understanding the flow of fluids in the reservoir and for predicting the effect of altering the physical system through various production/injection schemes. Although simplified models can be used to match the field behavior under a given operating scheme, more realistic models are needed to predict behavior when the physical system is changed.

In naturally fractured media, described as a permeable host rock containing a network of highly interconnected fissures, the application of a conventional single-porosity model of porous media has proved inadequate. The concept of a double-porosity system has been developed for use in simulating these systems. Fractures are assumed to be interconnected and provide the main flow path in the reservoirs. The host rock is assumed to be small disconnected blocks (matrix blocks) surrounded by the fractures. The matrix block size (or fracture spacing) is usually assumed to be described by some statistical function which is a property of the system. This concept has been applied by many authors and appears to be a realistic and practical method for modeling many fractured systems.

The double-porosity concept as normally applied in numerical simulators is used to obtain average fluid properties for the host rock (matrix blocks) in a given region because all the matrix blocks are lumped into one source term connected to the fracture in each mathematical grid block. The model becomes inadequate for simulating heat conduction, gravity segregation or other transient phenomena in individual matrix blocks. When matrix blocks are large and wells are shut in or producing at low rates, fluid or heat movement in individual matrix blocks are important phenomenon. Boundary conditions between matrix and fractures may also be more realistically modeled by the use of smaller matrix sub-domains.

The finite-difference scheme presented in this paper is an extension of an existing double-porosity concept (1). A fully-implicit, two-phase numerical simulator is presented in which matrix-blocks are represented by several sub-domains in a given grid block. The model is used to show gravity segregation effects in the matrix rock. An efficient scheme is presented for solving the equations so that the solution time does not become prohibitive as the number of matrix sub-domains increases. This method could also be used to simulate heat conduction or chemical transport in

two porosity systems and in multilayer systems where low permeability layers produce largely by vertical flow into high permeability layers.

Theory of Flow in Porous Media

The theory of flow in porous media originated in the 1850's with the work of Henry Darcy (2). He presented an empirical relation which related fluid flow in a sandpack to the pressure drop across the system through a constant of proportionality called the permeability of the system. King Hubbert (3) discusses Darcy's Law in relation to the Navier Stokes equation and for system geometries other than those studied by Darcy.

Darcy's Law is expressed in the following form (2):

$$v = - \frac{k}{\mu} \rho \nabla \Phi, \qquad [1]$$

where

$$\Phi = \int \frac{dp}{\rho} - gD. \qquad [2]$$

When density (ρ) is a function of pressure only, Equations 1 and 2 can be combined to the following familiar form of Darcy's Law,

$$v = - k/\mu \, [\nabla p - \rho g \nabla D], \qquad [3]$$

with terms defined as follows:

v - superficial velocity, L/T

k - permeability, L^2

μ - viscosity, $M/(L \cdot T)$

ρ - fluid density, M/L^3

g - acceleration of gravity, L/T^2

D - depth (positive down), L

∇ - gradient, L^{-1}

For single phase fluid flow, the continuity equation(conservation of mass) is,

$$-\nabla \cdot (\rho v) + \rho q = \frac{\partial (\rho \phi)}{\partial t} \qquad [4]$$

where ρq is a point source or sink mass rate per unit bulk volume, $M^3/(L^3 \cdot T)$, and ϕ is void fraction of the porous media. Combining Equations 3 and 4 gives the partial differential equation which describes the flow of a single phase fluid in a porous medium.

$$\nabla \cdot [\rho k/\mu (\nabla p - \rho g \nabla D)] + \rho q = \frac{\partial (\rho \phi)}{\partial t} \qquad [5]$$

Two Phase Flow Equations

For two phase immiscible flow, each phase is assumed to follow Darcy's Law. The fraction of the void space occupied by a given phase, α, is called the saturation of that phase, $S\alpha$. Because the permeability, k, in Equation 5 is defined

for single phase fluid flow, the permeability to a given phase is expressed as $k \cdot k_{r\alpha}$ for multiphase flow where $k_{r\alpha}$ is the relative permeability and has a value in the range 0 to 1. The relative permeability for a given phase is normally assumed to be a function of the phase saturation and increases with increasing saturation. Figure 1 is an example of relative permeability measured in the laboratory for an oil-water system (4).

Two immiscible phases will also have a pressure difference across any interface because of capillary and interfacial forces in the pores and the fluids.

Capillary pressure is most easily visualized by assuming that the porous medium consists of capillary tubes of varying sizes. If the porous medium is contacted to a free water surface, then water will rise in the capillaries to different heights depending on tube radii and surface tensions. The free water surface (where there is no capillary force) is called the water-oil contact. The variation in tube radii leads to a decreasing average water saturation with height above the water-oil contact. This height versus average saturation relation can be converted to an average capillary pressure versus saturation by assuming that oil and water pressures are equal at the water-oil contact and

that the two phases are in gravity equilibrium above this contact (dashed lines in Figure 2). If the average capillary pressure is defined as the difference between oil and water pressures, then the average capillary pressure at a given height, h, above the water-oil contact is:

$$Pc = g(\rho_w - \rho_o) \cdot h \qquad [6]$$

For a water-wet system, we get a capillary pressure versus water saturation curve as shown by the solid lines in Figure 2. The capillary pressure curves depend on the saturation direction. When oil displaces water such as during formation of the reservoir, then the "drainage" curve is followed. The water saturation will decrease to some minimum value at a large capillary pressure. This irreducible water saturation is where the water phase becomes discontinuous and the relative permeability becomes zero. When water displaces oil during production of a reservoir, the "imbibition" curve is followed and capillary pressure is normally assumed to go to zero at the residual oil saturation where oil becomes discontinuous and relative permeability is zero. For simplicity, the imbibition curve is always used in this model because that is generally the direction of saturation change. Discussion of initialization of the model is given in the section on Program Organization.

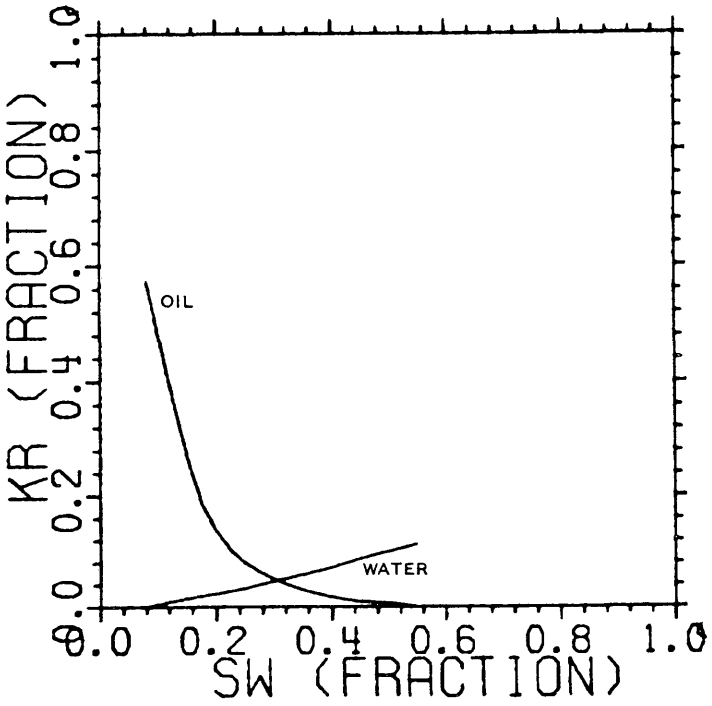

Figure 1. Two Phase Relative Permeability

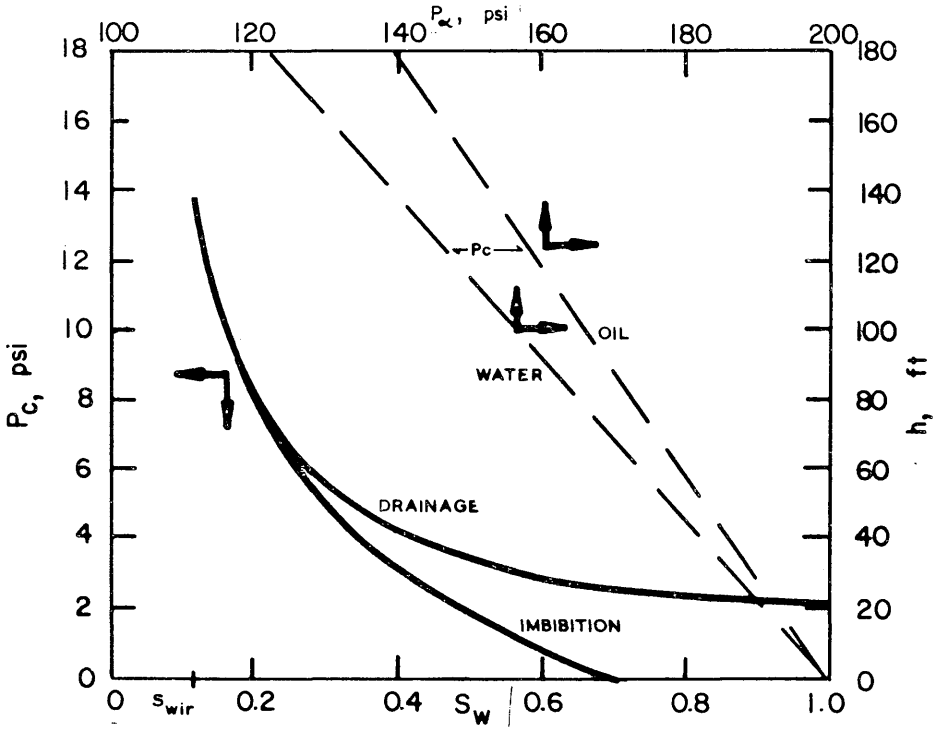

Figure 2. Two Phase Capillary Pressure

For two phase systems with low compressibility, it is often desirable to write Equation 5 in volumetric form because the petroleum industry most often deals with volume quantities. A dimensionless quantity called the formation volume factor is introduced and can be defined as

$$B\alpha = \rho_{s\alpha}/\rho_{r\alpha} \qquad [7]$$

where s and r represent surface and reservoir.

Combining Equations 5 and 7 with relative permeability and saturation gives the most often used form of the reservoir flow equations for oil-water systems:

$$\nabla\cdot\left[k\left(\frac{k_r}{\mu B}\right)_\alpha (\nabla p_\alpha - \rho_\alpha g \nabla D)\right] + q_{s\alpha} = \frac{\partial\left(\frac{\phi S}{B}\right)_\alpha}{\partial t} \qquad [8]$$

q_s is a volumetric flow rate per unit volume at surface conditions.

For two phase (oil-water) systems, Equation 8 is written for each phase. All properties are a function of either pressure or saturation. The four variables, pressure and saturation for each phase, are reduced to two by the following auxiliary relations:

$$S_o = 1 - S_w \qquad [9]$$

$$p_o = p_w + P_c \qquad [10]$$

where the capillary pressure, P_c, is given as a function of water saturation as discussed earlier.

Boundary Conditions

Equation 8 can be used to describe flow in a porous medium with the use of appropriate boundary conditions. Some of the most common boundary conditions are:

1) no flow -

$$(\nabla p_\alpha - \rho_\alpha g \nabla D)_b = 0 \qquad [11]$$

2) constant rate -

$$(\nabla p_\alpha - \rho_\alpha g \nabla D)_b = \text{constant} \qquad [12]$$

3) pot aquifer -

$$(\nabla p_\alpha - \rho_\alpha g \nabla D)_b = \frac{\partial p_\alpha}{\partial t} \qquad [13]$$

4) constant pressure -

$$(p_\alpha)_b = \text{constant} \qquad [14]$$

For balanced systems in which fluid injection equals fluid production, as in many waterflood systems, the no flow boundary condition is often used.

Theory of Flow in Fractured Reservoirs

The theory of flow in fractured porous media developed in the 1960's when Barenblatt (5) described a naturally fractured reservoir as a double porosity medium in which the two pore systems were coupled by a fluid transfer term. The

secondary porosity (fractures) is very permeable and is the main flow path while the primary porosity (matrix) has a much lower permeability, but contains the bulk of the fluids. Warren and Root (1) and later Odeh (6) introduced Barenblatt's work into the petroleum literature and gave analytical solutions for the pressure distribution in such a two-porosity system. They described the fractures as a highly interconnected system with the matrix separated into numerous small "matrix blocks" which must feed into the fractures (Fig. 3). The block size (or fracture spacing) is assumed to be a statistical representation of the system geometry.

Most of the literature on fractured reservoirs and models of double porosity reservoirs have developed along the lines of a statistical description of the reservoir because the number of fractures, size of the reservoir, and difficulty in directly measuring fracture distribution prohibits simulation of the exact geometry. However, Gilman and Kazemi (4) showed that a two-porosity model can represent a single fracture in addition to randomly fractured media if the fracture is the primary flow path (i.e., a single high permeability "fracture" connected to a low permeability "matrix" of known geometry.)

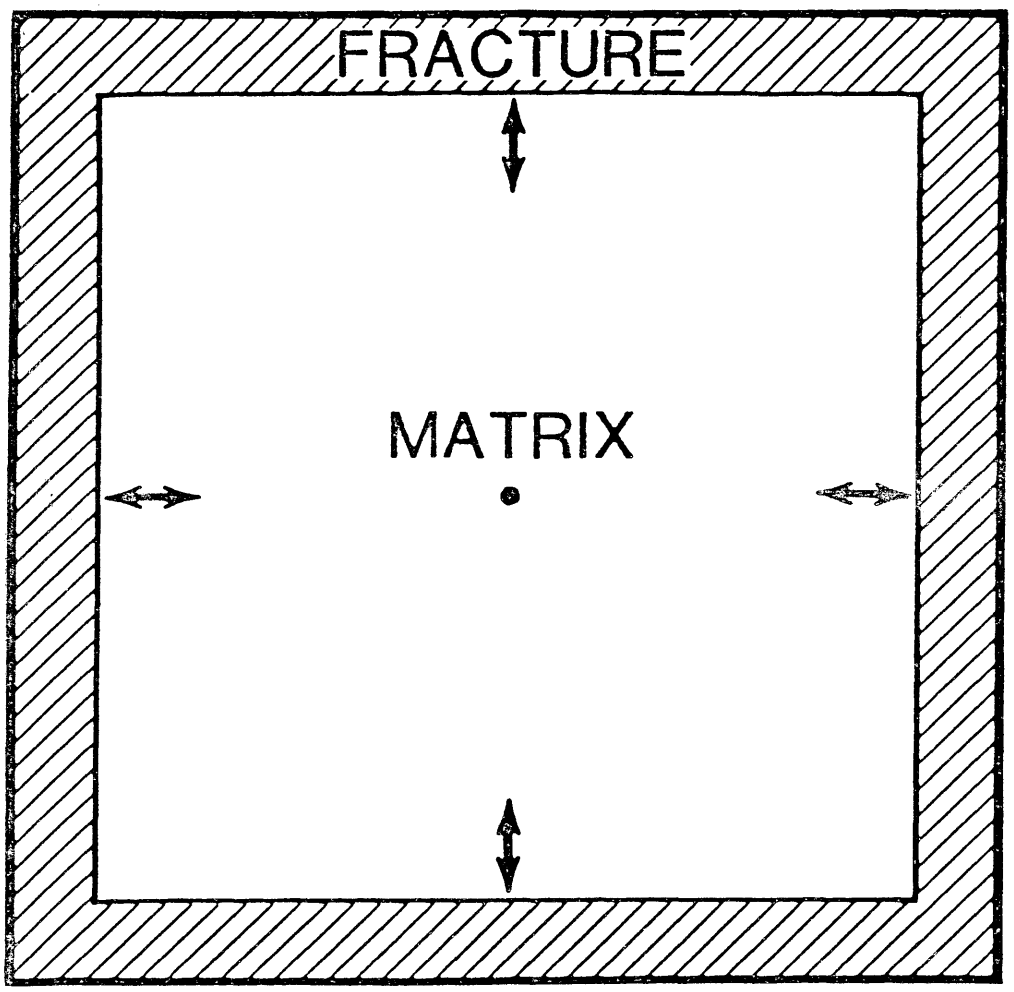

Figure 3. Double Porosity Model

Kazemi (7) and Duguid and Lee (8) were first to incorporate the two-porosity model into a numerical model and they showed the usefulness of the models for simulating flow in large scale systems.

Flow in the fractures is generally assumed to follow Darcy's Law in the two-porosity models. Several authors such as Witherspoon (9) have shown that Darcy's Law can be used to describe flow in the fractures, the permeability being related to fracture width.

Very little discussion of the data required for a two-porosity model is given in this thesis. It is assumed that the following data is available:

- Fracture and matrix relative permeability
- Fracture and matrix capillary pressure
- Fracture and matrix porosity
- Fracture and matrix permeability
- Matrix block size
- Fluid and formation compressibility
- Fluid viscosities
- Fluid densities
- Reservoir size

Some functional relationships for the data used in this simulator are given in a later section.

The equation to describe flow in the fractures is Equation 8 with an additional source term to describe flow between matrix and fractures.

$$\nabla\cdot[k_f(\frac{k_r}{\mu B})_{\alpha f}(\nabla p_{\alpha f} - \rho_\alpha g \nabla D_f)] + \tau_{maf\alpha} + q_{s\alpha} = \frac{\partial(\frac{\phi S}{B})_{\alpha f}}{\partial t} \quad [15]$$

The subscript f refers to the fracture and ma to the matrix. τ_{maf} is the volumetric rate of fluid transfer between matrix and fracture per unit bulk volume. Several authors have used empirical relations to describe τ_{maf} (10), but if Darcy's Law applies for flow between matrix and fracture, it can be shown to be

$$-\tau_{maf\alpha} = [k(\frac{k_r}{\mu B})_\alpha \sigma]_{ma}[\nabla p_{\alpha ma} - \rho_\alpha g \nabla Dma] \quad [16]$$

where σ depends only on the size and shape of the matrix blocks, not on the fluid, and is therefore termed the shape factor. For a rectangular block of lengths Lx, Ly, and Lz, the shape factor for a finite difference formulation is:

$$\sigma = 4\ (\frac{1}{Lx^2} + \frac{1}{Ly^2} + \frac{1}{Lz^2}) \quad [17]$$

Use of Equations 15 and 16 to describe flow in fractured systems accounts for viscous, capillary, and gravity forces between matrix and fracture and therefore should be

more reliable for predicting fracture-matrix flow than empirical relations.

The above two-porosity formulation does not require the choosing of a "transfer function" as many have misinterpreted. One only needs to find a σ which is a statistical (or could be an exact) representation of a geometry. In theory σ can be measured, however, in practice it is determined by history matching reservoir behavior. Most other properties in Equation 16 such as permeability, porosity, relative permeability, and capillary pressure could be measured on laboratory core samples. The spacing parameter, σ, is assumed to be a property of the system independent of grid size, as are permeability and porosity, but it can vary from one area of the reservoir to the next. Equation 17 is normally only used to determine apparent matrix block sizes (L).

The amount of transer to (from) the fracture must equal the loss (gain) in the matrix as described by Equation 18.

$$\tau_{maf\ \alpha} = \frac{\partial \left(\frac{\phi S}{B}\right)_{\alpha,ma}}{\partial t} \qquad [18]$$

This description of fracture-matrix flow appears to agree with the imbibition phenomena measured in the laboratory (11). During imbibition, water is drawn into the matrix rock by capillary force and oil is subsequently dis-

placed. The matrix capillary pressure curve is thus important in recovery efficiency in such systems.

A conventional simulator could be used to exactly reproduce this double-porosity model by using separate nodes for fracture and matrix, but the size and therefore the computing time would be prohibitive (4). This was discussed by Gilman and Kazemi (4).

Finite Difference Equations for Double Porosity Systems

Analytical solutions of the reservoir flow equations can only be obtained for very simplified systems, therefore numerical techniques are often used to solve the equations. Finite differences are most often used to solve the partial differential equations as they have proven to be the most reliable and easiest methods to implement. Other methods such as moving points, method of characteristics, finite element, Galerkin, and collocation have been attempted, but they are not proven as practical as finite differences and are computationally more expensive (12). Finite differences may not always be as accurate as some of the above-mentioned methods, however. Time and spatial truncation errors tend to smear sharp fronts (numerical dispersion). In heat flow or chemical transport, numerical dispersion may be very large and special techniques may be needed to avoid the problem.

Solutions can also depend on which way the grid is oriented with respect to production and injection wells.

Equations 15 and 18 can be written in a compact finite difference form as given by Equations 19 and 20.

$$\Delta[T_{\alpha f}(\Delta p_{\alpha f}-\gamma_{\alpha f}\Delta D_f)] + T_{\alpha maf}[(p_{\alpha ma}-p_{\alpha f})-\gamma_{\alpha maf}(D_{ma}-D_f)] + q_{\alpha s} = \frac{V}{\Delta t}\Delta_t\left(\frac{\phi S}{B}\right)_{\alpha f} \quad [19]$$

$$T_{\alpha maf}[(p_{\alpha f}-p_{\alpha ma})-\gamma_{\alpha maf}(D_f-D_{ma})] = \frac{V}{\Delta t}\Delta_t\left(\frac{\phi S}{B}\right)_{\alpha ma} \quad [20]$$

where the coefficients, $T\alpha f$ and $T\alpha maf$ are called transmissibilities and are defined below. The no flow boundary conditions most often used are accounted for by setting the $\Delta p_{\alpha f}$ and ΔD_f terms to zero at the boundary. Note that the matrix pressure and depth gradients are defined as matrix values minus fracture values. This is the approximation normally used for the double porosity formulation. In oil water systems, Equations 19 and 20 are written once for each phase. The auxilliary relations which reduce the number of unknowns to four are:

$$S_{oma} = 1 - S_{wma} \quad [21]$$

$$S_{of} = 1 - S_{wf} \quad [22]$$

$$P_{oma} = P_{wma} + Pc_{ma} \quad [23]$$

$$P_{of} = P_{wf} + Pc_f \quad [24]$$

By substituting the above relations into Equations 19 and 20 written for the oil phase, there will be four equations and four unknowns (S_{wma}, S_{wf}, P_{ma}, P_{wf}); all other terms are a function of these four unknowns.

Terms in Equations 19 and 20 are defined by Equations 25-28.

$$\Delta p = p_{i+1} - p_i \text{ or } p_i - p_{i-1} \quad [25]$$

$$T_{\alpha f} = \left(\frac{\lambda \cdot \text{Area}}{\Delta L}\right) (k\phi)_f \left(\frac{k_r}{\mu B}\right)_{\alpha f} \quad [26]$$

$$T_{\alpha ma} = (\lambda \cdot V)\ (k\sigma)_{ma} \left(\frac{k_r}{\mu B}\right)_{\alpha ma} \quad [27]$$

$$\Delta_t \left(\frac{\phi S}{B}\right) = \left(\frac{\phi S}{B}\right)^{n+1} - \left(\frac{\phi S}{B}\right)^{n} \quad [28]$$

where λ is 0.001127 for oil field units given in the nomenclature and V, Area, and ΔL are the volume of the grid block, flow area between grid blocks and length between grid block centers respectively. The fracture spacing parameter (shape factor), σ, as previously discussed, has units of inverse square length and is assumed to be a property of the system.It will not normally be a function of grid size. More details on the evaluation of the transmissibilities as given by Equations 26 and 27 will be given later. They are evaluated using standard techniques given in the petroleum literature.

Wellbore Source Terms

The source (sink) term resulting from a well in a grid block is normally derived assuming steady state radial flow between the grid block boundary and a point source at the center of the grid block. Using Darcy's Law and assuming that the node pressure, $p_{\alpha f}$, is the volume average pressure of fluid in the grid block, the flow rate is:

$$q_\alpha = \frac{\lambda \cdot 2 \cdot \pi \cdot \Delta z (k\phi)_f}{\frac{r_1^2}{r_1^2 - r_w^2} \ln (r_1/r_w) - 1/2 + s} \cdot \left(\frac{kr}{\mu B}\right)_{\alpha f} \cdot (p_{wL} - p_{\alpha f}) \qquad [29]$$

where Δz is the grid block thickeness and and r_1 is the outer radius of the grid block. For rectangular blocks, r_1 is usually assumed to be a function of block dimensions:

$$r_1 = \frac{\Delta x \Delta y}{\pi} \qquad [30]$$

More exact relations have been proposed (13).

Equation 29 results in an additional unknown for every well -- the wellbore pressure, p_{wL}. Each well must have some constraint in order to totally specify the system. Normally a total fluid production (or injection) rate is known.

$$Q_T = \sum_{\ell} \sum_{\alpha} q_{\alpha\ell} \qquad [31]$$

where Q_T is the total fluid rate and ℓ and α represent the layers and phases respectively.

The equations presented in this section represent a well defined system. However, the coefficients are functions of the variables and therefore the equations are non-linear. The next section will discuss one method for linearizing the equations. Numerous other methods have been used (12, 14).

Finite Difference Equations for Extended Double Porosity Model

The purpose of this thesis and the simulator is to extend the double porosity concept by dividing the matrix in a given grid block into several sub-domains which may be connected to one another and/or to the fracture. Figure 3 showed one matrix sub-domain in a given grid block. Figure 4 shows two possible divisions of the matrix block into 3 sub-domains. This is still the same system as shown in Figure 3, but the sub-domains are chosen to more closely represent the physics of the problem. Equation 19 will contain several matrix/fracture terms and Equation 20 will need to include terms resulting from finite difference of the matrix. As shown in Figure 4, not all matrix sub-domains may be connected to the fractures, and the flow areas between matrix and fracture may be different. Therefore, the shape factor, σ, may be different for each sub-domain. No direct flow is allowed between matrix blocks from one grid block to

another because it is assumed that the matrix is discontinuous and the fractures are the flow path from one grid block to the next.

This program is developed for one-dimensional flow in both fracture and matrix and therefore the finite difference equations will be presented as such. This method could readily be extended to multidimensions for either the fracture or the matrix sub-domain. The solution method in this paper is actually independent of the intramatrix connections, therefore the connections could be as complex as desired to match the physics of the problem.

Darcy's law is applied to flow in the fractures, in the matrix blocks, and between matrix and fractures; therefore, the equations presented earlier still apply. Equation 20 needs to have an additional term representing transport between matrix sub-domains, and Equation 19 will have a fracture-matrix transfer term for each sub-domain connected to the fracture in a given grid block. The number of variables increases from four per grid block to 2*(NMB+1), where NMB is the number of matrix sub-domains.

For each phase (oil and water), a fracture equation is required for each grid block, and a matrix equation is required for each sub-domain in each grid block.

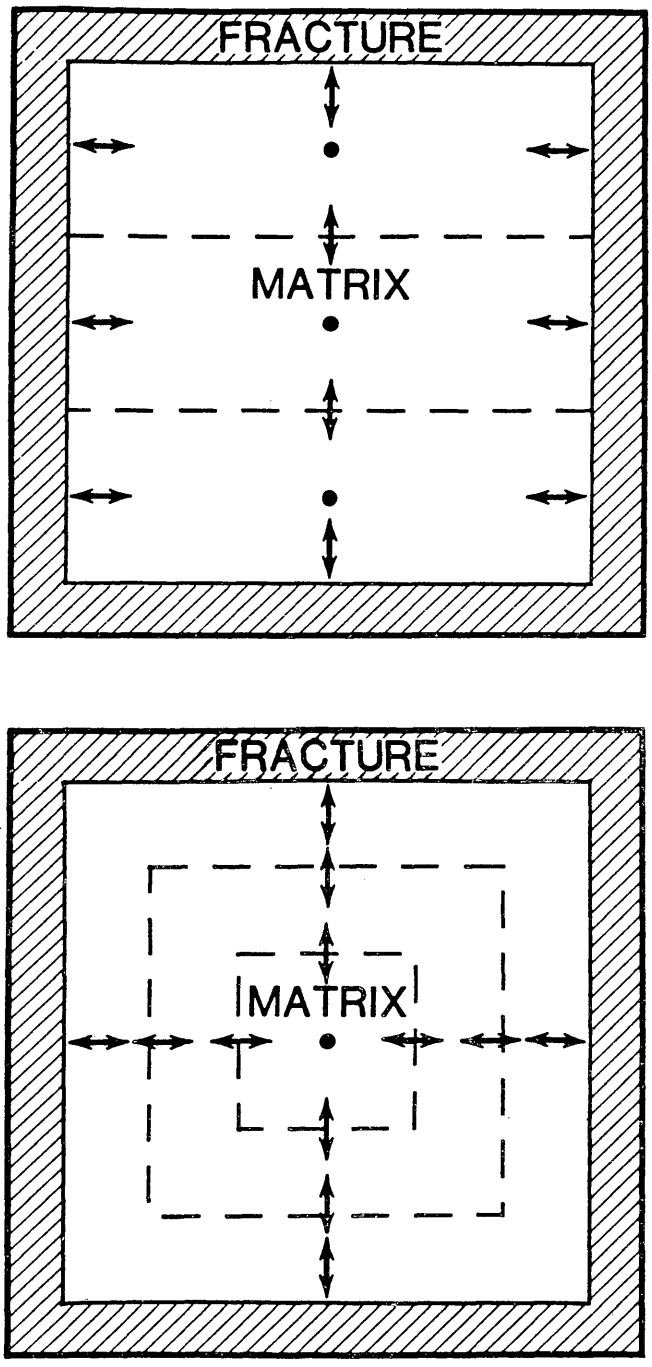

Figure 4. Extended Double Porosity Model.

The concept presented in this paper is similar to ideas presented by Pruess and Narisimhan (15), Eric and Ershagi (16) and Joseph Papay (17).

If Equations 19 and 20 are written in expanded form for one-dimensional flow in fracture and in the matrix, the water equations for grid block 'i' and matrix sub-domain "j" are:

$$T_{wf,i+1/2}[(p_{wf,i+1}-p_{wf,i})-\gamma_{wf,i+1/2}(D_{f,i+1}-D_{f,i})]$$

$$-T_{wf,i-1/2}[(p_{wf,i}-p_{wf,i-1})-\gamma_{wf,i-1/2}(D_{f,i}-D_{f,i-1})]$$

$$+\sum_{j=1}^{NMB} T_{w,maf,ij}[p_{wma,i,j}-p_{wf,i}-\gamma_{maf,i,j}(D_{ma,i,j}-D_{f,i})]$$

$$+q_{w,i}=\frac{V}{\Delta t}\left[\left(\frac{\phi S}{B}\right)_{wf,i}-\left(\frac{\phi S}{B}\right)^{n}_{wf,i}\right] \qquad [32]$$

$$T_{wma,i,j+1/2}[p_{wma,i,j+1}-p_{wma,i,j}-\gamma_{ma,i,j+1/2}(D_{ma,i,j+1}-D_{ma,i,j})]$$

$$-T_{wma,i,j-1/2}[p_{wma,i,j}-p_{wma,i,j-1}-\gamma_{ma,i,j-1/2}(D_{ma,i,j}-D_{ma,i,j-1})]$$

$$-T_{wmaf,i,j}[p_{wma,i,j}-p_{wf,i}-\gamma_{maf,i,j}(D_{ma,i,j}-D_{f,i})]$$

$$\frac{V}{\Delta t}\left[\left(\frac{\phi S}{B}\right)_{wma,i}-\left(\frac{\phi S}{B}\right)^{n}_{wma,i}\right]_j \qquad [33]$$

Two additional equations are written for the oil phases. Oil pressure and saturation are then eliminated with Equations 21 through 24.

NMB is the number of matrix sub-domains in a given grid block and the n superscript represents values at the last time level. All variables without superscripts are assumed to be evaluated at the current time level. If a given grid block is not connected to the fracture, then the appropriate T_{maf} term could be set to zero.

To simulate the no flow conditions for both fracture and matrix, the i+1 (j+1) or i-1 (j-1) terms are set equal to the corresponding i(j) terms at the boundary nodes.

The matrix equation has no well source or sink term since it is assumed that the fractures are the primary flow path. Either Equation 32 or 33 could easily be extended to multiple dimensions independent of the number of dimensions of the other porosity.

Evaluation of the transmissibility terms ($T_{i+1/2}$ or $T_{i-1/2}$) between grid blocks has been of concern for many years. There are a number of ways to evaluate them (18). Equations 34-39 have been found to generally give the best results. These are the standard techniques found in the petroleum literature.

$$T_{\alpha f,i+1/2} = \lambda\left\{\frac{\text{Area}}{\Delta x}(k\phi)_f\right\}_{i+1/2}\left\{\omega_{i+1}\left(\frac{k_r}{\mu B}\right)_{\alpha f,i+1}+(1-\omega)_{i+1}\left(\frac{k_r}{\mu B}\right)_{\alpha f,i}\right\} \quad [34]$$

where ω provides for upstream weighting. (The direction flow is coming from is upstream).

$$\omega_{i+1} = 1 \text{ if } p_{i+1}-p_i-\gamma_{i+1/2}(D_{i+1}-D_i) > 0 \quad [35]$$

$$\omega_{i+1} = 0 \text{ if } p_{i+1}-p_i-\gamma_{i+1/2}(D_{i+1}-D_i) < 0 \quad [36]$$

Upstream weighting has been found to be stable, but introduces numerical dispersion. It is generally first order correct. Terms of Equation 34 are further defined as:

$$\text{Area}_{i+1/2} = \frac{\text{Area}_{i+1} + \text{Area}_i}{2} \quad [37]$$

$$\Delta x_{i+1/2} = \frac{(\Delta x_{i+1} + \Delta x_i)}{2} \quad [38]$$

$$(k\phi)_{f,i+1/2} = \frac{2\Delta x_{i+1/2}(k\phi)_{f,i+1}(k\phi)_{fi}}{\Delta x_{i+i}(k\phi)_{f,i} + \Delta x_i(k\phi)_{f,i+1}} \quad [39]$$

Equation 39 is a harmonic average permeability between two nodes and ensures that the lower permeability dominates the value of $k_{i+1/2}$. The effective fracture permeability, $(k\phi)_f$, is used in the previous equation as required for double-porosity systems (4). For flow between the matrix sub-domains, Equations 34 through 39 could be used with

matrix data replacing the fracture data. There would be no ϕ term, and the Area and Δx terms would be the values for the matrix sub-domains.

For flow between the matrix and fractures, the transmissibility is:

$$T_{\alpha maf,i,j} = \{V\sigma k_{ma}\}_{i,j}\{\omega_{maf,i,j}\left(\frac{k_r}{\mu B}\right)_{f,i} + (1-\omega_{maf})_{i,j}\left(\frac{k_r}{\mu B}\right)_{ma,i,j}\} \quad [40]$$

The V,σ, and k_{ma} terms are for the given matrix sub-domain and therefore do not need to be averaged. A harmonic average of k_{ma} and $k_{f\phi f}$ could be used, but the smaller k_{ma} value would dominate. Again, ω_{maf} is a weighting factor depending on whether the fracture or matrix is upstream. If flow is from the fracture to matrix as determined by equations similar to 35 and 36, then ω_{maf} is 1.0 because the fracture is upstream and thereforey, $(kr/\mu B)_f$, is used in Equation 40. Other methods have been presented for evaluating mobility between the fracture and matrix (19) as the true relative permeability between matrix and fracture is not obvious. The gravity gradients, γ in Equations 32 and 33 are normally assumed to be average values between two nodes.

The wellbore water relative permeability in Equation 29 also is assumed to depend on the upstream direction (whether the well is an injector or producer).

$$q_{w,i} = PI_i \left(\frac{krw + \omega kro}{\mu B_w}\right)_f (p_{wL} - p_{\alpha f}) \qquad [41]$$

where PI is the constant terms in Equation 29 and ω is defined as:

$$\omega = 1, \; q_w > 0 \text{ (injection)}$$

$$\omega = 0, \; q_w < 0 \text{ (production)} \qquad [42]$$

The oil rate is then:

$$q_{oi} = PI_i \left(\frac{(1-\omega)kr}{\mu B}\right)_{of} (\qquad) \qquad [43]$$

and therefore will be zero for injection. Evaluation of terms as given by Equations 34 through 43 will result in the two porosity model giving the same answers as a conventional porous medium simulator with separate grid blocks used for the fracture and matrix as shown in Reference 4.

Radial Transformation

For some systems such as a single well, single phase transient simulation, radial systems are more appropriate. Equations 39-43 can be used with the following exact transformations in which grid radii are distributed exponentially. These transformations have been included in the model.

Let

$$\Delta u = \frac{1}{IMAX} \ln r_e / r_w \qquad [44]$$

where IMAX is the number of nodes, r_e is an external boundary radius and r_w is the wellbore radius.

$$\Delta x_{i+1/2} = r_w \left(e^{(i+1/2)\Delta u} - e^{(i-1/2)\Delta u}\right) \qquad [45]$$

$$Area_{i+1/2} = 2\pi r_w e^{i\Delta u} \Delta z \qquad [46]$$

$$V_i = \pi r_w^2 \left(e^{2i\Delta u} - e^{2(i-1)\Delta u}\right) \Delta z \qquad [47]$$

$$k_{i+1/2} = \frac{\Delta x_{i+1/2}\, k_{i+1}\, k_i}{\left(e^{(i+1/2)\Delta u} - e^{i\Delta u}\right) k_i + \left(e^{i\Delta u} - e^{(i-1/2)\Delta u}\right) k_{i+1}} \qquad [48]$$

For short time transient data it is also desirable to take time steps exponentially as given by equations 49 and 50

$$\Delta u_t = \frac{1}{NUMTS-1} \ln(TMAX/\Delta t_1) \qquad [49]$$

$$\Delta t_{i+1} = \sum_i \Delta t_i \left(e^{\Delta u_t} - 1\right) \qquad [50]$$

where NUMTS is the number of time-steps and TMAX is the final time.

Linearization of the Finite Difference Equations

As previously mentioned, the porous media equations can be highly non-linear. With the availability of high speed

computers with large memory, the Newton-Raphson approach has been widely used to solve for all unknowns simultaneously. This fully implicit formulation results in very stable systems, even when time-steps are large. Time truncation error can be significant in this approach, therefore, time-step size must be chosen carefully.

Equations 32 and 33 can be written in matrix form for any number of nodes as:

$$Ax = b \qquad [51]$$

where, for each node i:

$$x_i = \begin{bmatrix} S_{wf} \\ P_{wf} \\ S_{wm,1} \\ P_{wm,2} \\ \vdots \\ S_{wm,NMB} \\ P_{wm,NMB} \end{bmatrix}_i \qquad [52]$$

In equation 51 the coefficients A and the right-hand side, b are functions the unknowns, x and therefore Equation 51 is non-linear.

One common method of solving the equation is by Newton-Raphson iteration. Equation 51 is written in residual form:

$$R = Ax - b, \qquad [53]$$

then by Taylor Series expansion, neglecting second order and higher terms

$$R^{n+1} = R^n + \left(\frac{\partial R}{\partial x}\right)^n (x^{n+1} - x^n) \quad [54]$$

If R^{n+1} is to go to zero, then

$$-\left(\frac{\partial R}{\partial x}\right)^{\ell} \delta x^{\ell+\ell} = R^{\ell} \quad [55]$$

where ℓ represents the iteration level and

$$\delta x^{\ell+1} = x^{\ell+1} - x^{\ell} \quad [56]$$

Equation 55 is again of the form of Equation 51. Iteration is performed until δx is less than some arbitrary convergence limit.

The $\frac{\partial R}{\partial x}$ term is the Jacobian matrix and it represents the partial derivatives of all residuals with respect to all unknowns. In the finite difference formulation many terms are zero and a sparse matrix will be formed. Numerous authors have presented details on efficient methods for solv ing the reservoir flow equations by the Newton-Raphson metho (20).

For a given node, i, the fracture equation will depend only on fracture variables of immediately surrounding nodes, the matrix variables of node i and the wellbore pressure if the node contains a well. The residual equations for the fracture can be written in the following form.

$$R_{fi} = A_i \delta x_{f,i-1} + Bl_i \ \delta x_{f,i} + B2_i \ \delta x_{m,i} + C_i \delta x_{f,i+1} + D_i \delta p_{w,i} \quad [57]$$

$$A_i = \frac{\partial R_{f,i}}{\partial x_{f,i-1}} \quad [58]$$

$$Bl_{,i} = \frac{\partial R_{f,i}}{\partial x_{f,i}} \quad [59]$$

$$B2_{,i} = \frac{\partial R_{f,i}}{\partial x_{m,i}} \quad [60]$$

$$C_i = \frac{\partial R_{f,i}}{\partial X_{f,i+1}} \quad [61]$$

$$D_i = \frac{\partial R_{f,i}}{\partial p_{w,i}} \quad [62]$$

$$R_{f,i} = \begin{bmatrix} R_{fw,i} \\ R_{fo,i} \end{bmatrix} \quad [63]$$

$$\delta x_{f,i} = \begin{bmatrix} \delta S_{wf,i} \\ \delta p_{wf,i} \end{bmatrix} \quad [64]$$

$$\delta x_{m,i} = \begin{bmatrix} \delta S_{wm,1} \\ \delta p_{wm,1} \\ \vdots \\ \delta s_{wm,NMB} \\ \delta p_{wm,NMB} \end{bmatrix}_i \quad [65]$$

For the matrix residuals:

$$R_{m,i} = B3_i \ \delta x_{f,i} + B4_i \ \delta x_{m,i} \quad [66]$$

where

$$Rm,i = \begin{bmatrix} R_{mw,1} \\ R_{mo,1} \\ \vdots \\ R_{mw,NMB} \\ R_{mo,NMB} \end{bmatrix} \quad [67]$$

$$B3_i = \frac{\partial R_{m,i}}{\partial x_{f,i}} \quad [68]$$

$$B4_i = \frac{\partial R_{mi}}{\partial x_{m,i}} \quad [69]$$

The wellbore residual for a given well, wL is:

$$R_{wL} = E_{wL}\ \delta x_{f,i} + B5_{wL}\delta p_{wL} \quad [70]$$

where

$$E_{wL} = \frac{\partial R_{wL}}{\partial x_{f,i}} \quad [71]$$

$$B5_{wL} = \frac{\partial R_{wL}}{\partial p_{wL}} \quad [72]$$

Figure 5a shows the most compact equation form for a one-dimensional, three-node system with a well in the first node. The form is identical to that for any two-porosity simulator, but the partial derivative sub-matrices are of a larger size depending on the number of sub-domains in a given grid block. The A, B1, and D coefficients will be 2x2, the B2 matrix will be 2x2NMB, B3 -- (2NMBx2), B4 -- (2NMBx2NMB), E -- (1x2), and B5 -- (1x1), where NMB is the number of matrix sub-domains.

Derivatives can be taken either analytically or numerically. Analytical derivatives will generally require less work and storage if they can be found. Analytical derivatives were used in this model.

Solution of the Linear Equations

Solution of the equation matrix will often be a major portion of the total computer time in the reservoir simulator. The structure of the double-porosity form can be used to advantage to reduce the matrix to a simpler form, and thus greatly reduce the time required to solve the equations. Implicit wells terms will also have to be rearranged to get the matrix into an efficient band structure.

Implicit Wells

Solution of implicit wells has been discussed by several authors (21). The D coefficient in Figure 5a can be eliminated by the B5 coefficient below it (Figure. 5b) and then the upper banded portion of the matrix can be solved first. The change in wellbore pressure is then given by:

$$\delta p_{wL} = B5_{wL}^{-1}(R_{wL} - E_{wL}\delta x_{fi}) \qquad [73]$$

Reduction of the Double Porosity Coefficients

The bandwidth of the matrix is still much larger than a conventional simulator. Solution time is normally on the order of bandwidth squared, therefore it is important to put

the matrix in the most compact form. The B4 coefficients can be used to eliminate the B2 terms above them as shown in Figure 5b. The matrix can then be rearranged as in Figure 5c and the upper left hand band matrix can be used to first solve for change in fracture unknowns. The change in matrix unknowns is given by:

$$\delta x_{m,i} = B4_i^{-1}(R_{m,i} - B3_i \delta x_{f,i}) \qquad [74]$$

This matrix will be much faster to solve than the original matrix and, therefore, this simulator will not require a large amount of additional time compared to a conventional model.

Discussion of some timing comparisons for different numbers of matrix sub-domains is given in the section on Program Verification. The important concept here is that additional information can be obtained at a much reduced computing time compared to the use of a standard formulation with additional grid blocks to represent the matrix.

It appears that the reduction of the equations as shown by Figures 5a through 5c is directly proportional to the number of sub-domains. For solution of a banded matrix, solution time is proportional to the number of bands squared and, therefore, it is important to reduce matrix size.

$$\begin{bmatrix} B1 & B2 & C & & & & D \\ B3 & B4 & & & & & \\ A & & B1 & B2 & C & & \\ & & B3 & B4 & & & \\ & & A & & B1 & B2 & \\ & & & & B3 & B4 & \\ E & & & & & & B5 \end{bmatrix} \cdot \begin{bmatrix} \delta x_{f1} \\ \delta x_{m1} \\ \delta x_{f2} \\ \delta x_{m2} \\ \delta x_{f3} \\ \delta x_{m3} \\ \delta x_{wL} \end{bmatrix} = \begin{bmatrix} R_{f1} \\ R_{m1} \\ R_{f2} \\ R_{m2} \\ R_{f3} \\ R_{m3} \\ R_{wL} \end{bmatrix}$$

Figure 5a. Equation Form for a Three Node, One Well System

$$\begin{bmatrix} B1'' & 0 & C & & & & 0 \\ B3 & B4 & & & & & \\ A & & B1' & 0 & C & & \\ & & B3 & B4 & & & \\ & & A & & B1' & 0 & \\ & & & & B3 & B4 & \\ E & & & & & & B5 \end{bmatrix} \cdot \begin{bmatrix} \delta x_{f1} \\ \delta x_{m1} \\ \delta x_{f2} \\ \delta x_{m2} \\ \delta x_{f3} \\ \delta x_{m3} \\ \delta x_{wL} \end{bmatrix} = \begin{bmatrix} R_{f1}'' \\ R_{m1} \\ R_{f2}' \\ R_{m3} \\ R_{f3}' \\ R_{m3} \\ R_{wL} \end{bmatrix}$$

Figure 5b. Reduced Equation Form for Double Porosity, Implicit Well Model

$$\begin{bmatrix} B1'' & C & & & & & \\ A & B1' & C & & & & \\ & A & B1' & & & & \\ B3 & & & B4 & & & \\ & B3 & & & B4 & & \\ & & B3 & & & B4 & \\ E & & & & & & B5 \end{bmatrix} \cdot \begin{bmatrix} \delta x_{f1} \\ \delta x_{f2} \\ \delta x_{f3} \\ \delta x_{m1} \\ \delta x_{m2} \\ \delta x_{m3} \\ \delta x_{wL} \end{bmatrix} = \begin{bmatrix} R_{f1}'' \\ R_{f2}' \\ R_{f3}' \\ R_{m1} \\ R_{m2} \\ R_{m3} \\ R_{wL} \end{bmatrix}$$

Figure 5c. Rearranged Equation Form for Double Porosity Model

Program Organization

Initialization

Figure 6 shows the program flow chart (the program is listed in Appendix A). Once the data have been read, the model must be intialized to ensure that the system is in gravity-capillary equlibrium as described in the section on Two Phase Flow. The initialization can be performed using a capillary pressure curve, water-oil contact pressure, water-oil contact depth, node center depths (D), and oil and water densities.

For equilibrium, there must be no flow between nodes as described by:

$$\Delta[T(\Delta p-\gamma\Delta D)]= 0 \qquad [75]$$

and no flow between fracture and matrix:

$$T_{ma}[p_{ma}-p_f-\gamma_{maf}(D_{ma}-D_f)] = 0 \qquad [76]$$

Since the density is a function of pressure, the initiliazation becomes an iterative procedure. For gravity equilibrium, the following equation applies for each phase:

$$p_i = p_{i-1} - \frac{\gamma_i+\gamma_{i-1}}{2} (D_i-D_{i-1}) \qquad [77]$$

For node one, p_{i-1} is the water-oil contact pressure and Di-1 is the water-oil contact depth. Initialization of the fracture is done first, then the first matrix sub-domain in a given node is initialized by:

$$P_{ma} = P_f - \frac{\gamma_{ma}+\gamma_f}{2} (D_{ma} - D_f) \quad [78]$$

then the other matrix blocks can be initialized as given by equation 77.

The entire initialization must be done for both oil and water. At the water-oil contact the pressure of the two phases are assumed equal. The difference in densities of the two phases will result in increasing capillary pressure away from the water-oil contact (Fig. 2). From the capillary pressure curve, saturations of the nodes can be determined.

If the calculated capillary pressure is greater than that allowed by the input capillary pressure curve, then the water pressure is calculated as:

$$P_w = P_o - Pc_{max} \quad [79]$$

The water potential gradient will no longer be zero, but because the water is at the irreducible saturation, the transmissibility, T, will be zero because water relative permeability is zero and equation 75 will still be satisfied.

A Newton-Raphson technique could be used to solve equations 77 and 78, but for oil and water systems, density is only a weak function of pressure. The linear iteration used provides rapid convergence for the density function used in this program.

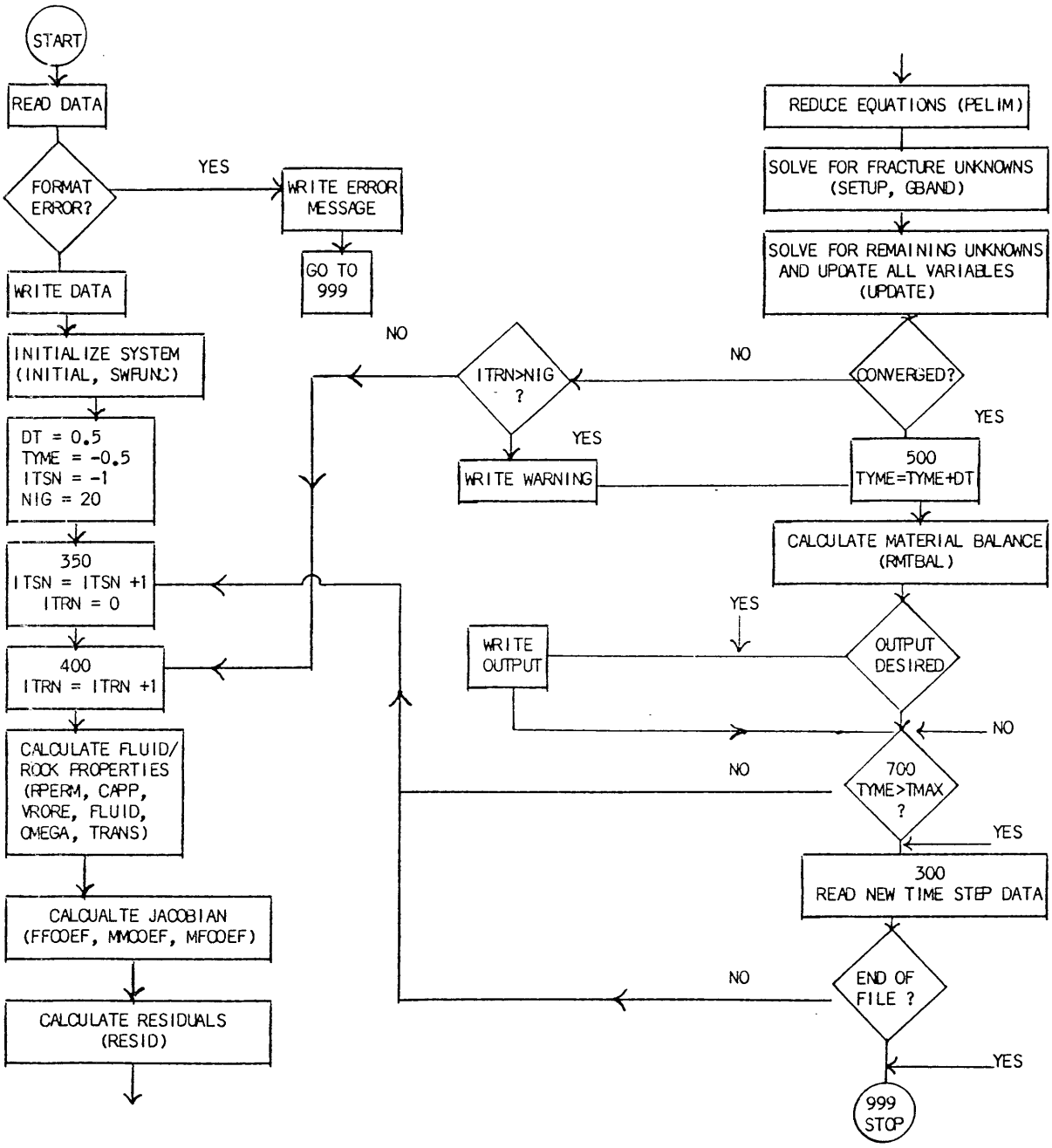

Figure 6. Program Flow Chart

To ensure equilibrium between matrix and fracture, the maximum capillary pressure in the fracture must be less than or equal to the maximum capillary pressure in the matrix.

After initialization, the model is run one time-step with no wells to ensure that the system is properly equilibrated. Once the model has been initialized, time-step information and wellbore rates can be read to begin simulation of the desired reservoir.

Fluid Properties

Fluid property calculation has been kept relatively simple for both ease of input and ease of programming; however, the correlations can fit a wide variety of actual data.

For systems with low compressibility, formation volume factor and density can be assumed to be linear functions of pressure.

$$B_\alpha = B_{\alpha b}[1 - c_\alpha(p_\alpha - p_b)] \qquad [80]$$

$$\rho_\alpha = \rho_{\alpha b}[1 + c_\alpha(p_\alpha - p_b)] \qquad [81]$$

where c_α is the fluid compressibility and the subscript b represents a base value.

The porosity is also assumed to be a linear function of pressure as determined by the rock compressibility, c_ϕ

$$\phi = \phi_b[1 + c_\phi(p_\alpha - p_b)] \qquad [82]$$

Relative permeabilities and capillary pressures are assumed to be defined by power fit equations.

Water relative permeability:

$$k_{rw} = k_{rw}(S_{or})\left[\frac{S_w - S_{wir}}{1-S_{or}-S_{wir}}\right]^{EW} \qquad S_{wir} < S_w < 1-S_{or} \qquad [83]$$

Oil relative permeability:

$$k_{ro} = k_{ro}(S_{wir})\left[\frac{1-S_{or}-S_w}{1-S_{wir}-S_{or}}\right]^{EO} \qquad S_{wir} < S_w < 1-S_{or} \qquad [84]$$

Positive Capillary Pressure:

$$P_c = P_c(S_{wir})\left[\frac{S_w(P_c=0) - S_w}{S_w(P_c=0) - S_{wir}}\right]^{EP} \qquad S_{wir} < S_w < (P_c=0) \qquad [85]$$

Negative Capillary Pressure:

$$P_c = P_c(S_{or})\left[\frac{S_w - S_w(P_c=0)}{1-S_{or}-S_w(P_{c=0})}\right]^{EM} \qquad S_w(P_c=0) < S_w < 1-S_{or} \qquad [86]$$

The terms in brackets are dimensionless saturations defined as S_D throughout the remainder of this paper. Oil and water viscosities have been assumed to be constant.

Figure 7 shows the input requirements. Card 2 through Card 8 are each used once. A separate Grid 1 Card is required for each grid block and a Grid 2 Card for each matrix sub-domain. A new Time Card can be used whenever

time-step size or well rate changes are required. A Well Card is required for each well after each Time Card. The variables are described in the nomenclature and at the start of the program listed in the Appendix.

The input data is printed by the simulator each time the program is run. Extensive data checking is not included. At the time-step frequency given on the Time Card, the simulator will print fracture and matrix pressure arrays and fracture and matrix saturtion arrays. For the wells, oil and water rates and cumulative produced or injected volumes re given. Material balances for both phases are calculated and printed for each time step as a check on the solution stability. If the maximum itrations are exceeded, a warning is printed. The print frequency can be changed by the use of a new Time Card.

To simulate a single-porosity system the spacing parameter, SIGMA, should be set to zer on all Grid 1 Cards. The program will still expect to read at least one Grid 1 Card for each grid block.

The program will continue to the maximum allowed time, even if the maximum iterations are exceeded for a time-step. It is important to check that the material balance is good before using the model results. If maximum iterations are being exceeded, time-step size or grid-block dimensions may need to be changed.

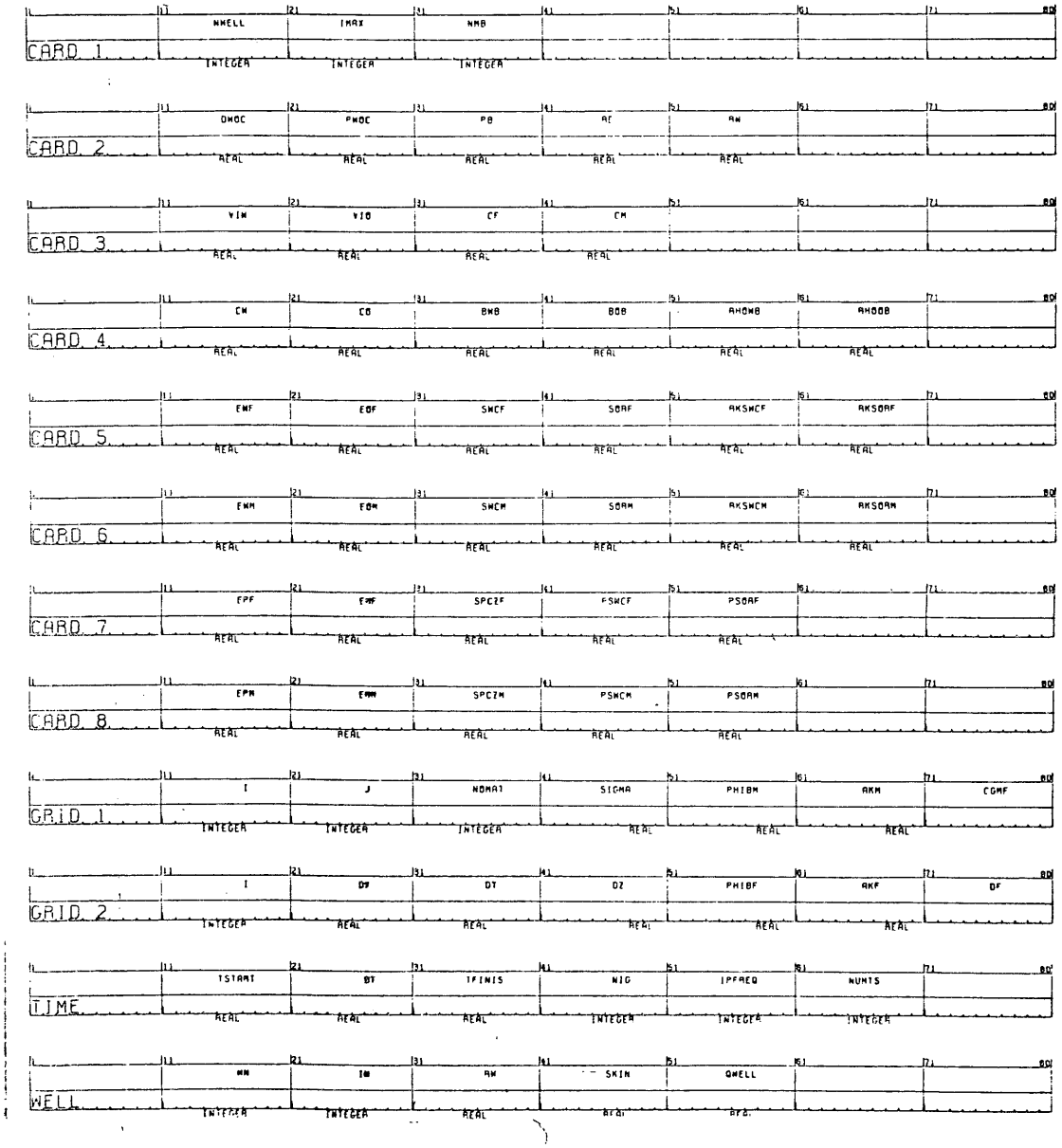

Figure. 7 Program Input Format

Program Verification

To verify the accuracy of the program, a single phase, two-porosity radial system was compared with the analytical solution for an infinite systems. Two-phase linear systems given by Gilman and Kazemi (4) were also matched with the simulator.

A single-porosity, one-dimensional, two-phase model was simulated to show the numerical dispersion effects in fully-implicit simulators. A match of a laboratory flood was made and relative solution times for multiple matrix sub-domains are discussed.

Single-Phase Radial Model

The radial system given in Table 1 was used to compare the analytical and numerical solutions of the single-phase, two-porosity system given by Equations 15, 16 and 18. In the simulator a large area was used to minimize boundary effects. The analytical solution is for an infinite system. No-flow outer boundary conditions were used in the numerical simulator, but because the flow time is short, the boundary condition was not important. Nodes were distributed exponentially and time-steps were taken exponentially to get the maximum accuracy. Figures 8 and 9 show the pressure response at the wellbore during constant rate production of

the well (pressure drawdown) and during shutin (buildup). The numerical and analytical solutions are in agreement. Reference 2 gives the analytical solution. The two-slope pressure transient plots are a characteristic of the double-porosity media. The first slope represent fracture response, and the transition to the second slope results as the matrix begins responding to the fracture flow.

The two straight lines should be parallel with the slope of the lines being directly proportional to the permeability of the system. The pressure separation between the two lines is a measure of fracture storage capacity divided by total storage capacity. The inflection point can be used to determine the fracture spacing parameter, σ. Pressure transient testing is one method to determine data for numerical simulators.

The transition curve between the two parallel straight lines cannot be accurately simulated using the standard two-porosity formulation. Use of multiple matrix sub-domains will give a more accurte pressure response as shown in a later section. The use of multiple matrix sub-domains allows transient flow within the matrix blocks. This gives a higher slope for the transitional portion of the semilog pressure-time plot. This slope should be approximately one-half the early and late time slopes for an ideal system.

Table 1

Single Phase, Double Porosity Model Data

p_i	1646.07 psi
μ	0.8 cp
c_w	3.5×10^{-6} psi $^{-1}$
B_w	0.9942 res. bbl/STB
r_w	0.34 ft
r_e*	5957.84 ft
$c_{\phi f}$	3×10^{-6} psi^{-1}
$c_{\phi ma}$	3×10^{-6} psi^{-1}
h	210 ft
k_f	1000 md
k_{ma}	1 md
ϕ_{ma}	0.1392
ϕ_f	0.02
σ	0.0016 ft^{-2}
q	100 STB/D
Δt_i**	10^{-6} days

*20 nodes were distributed exponentially between r_w and r_e using equations 44-48.

**100 times step were distributed exponentially for a 1-day buildup and a one-day drawdown using equations 49 and 50.

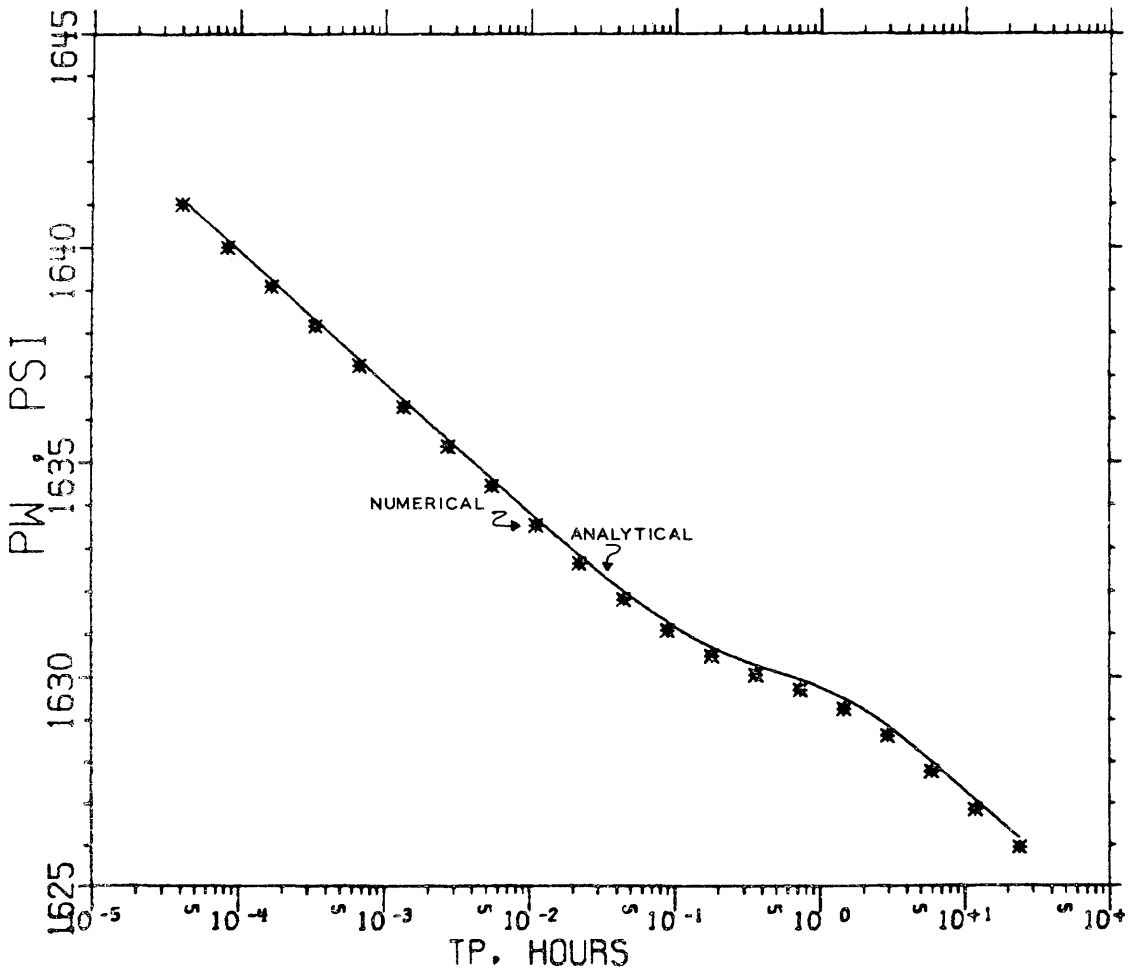

Figure 8. Analytical and Numerical Comparison for a Single-Phase Drawdown in a Double-Porosity System

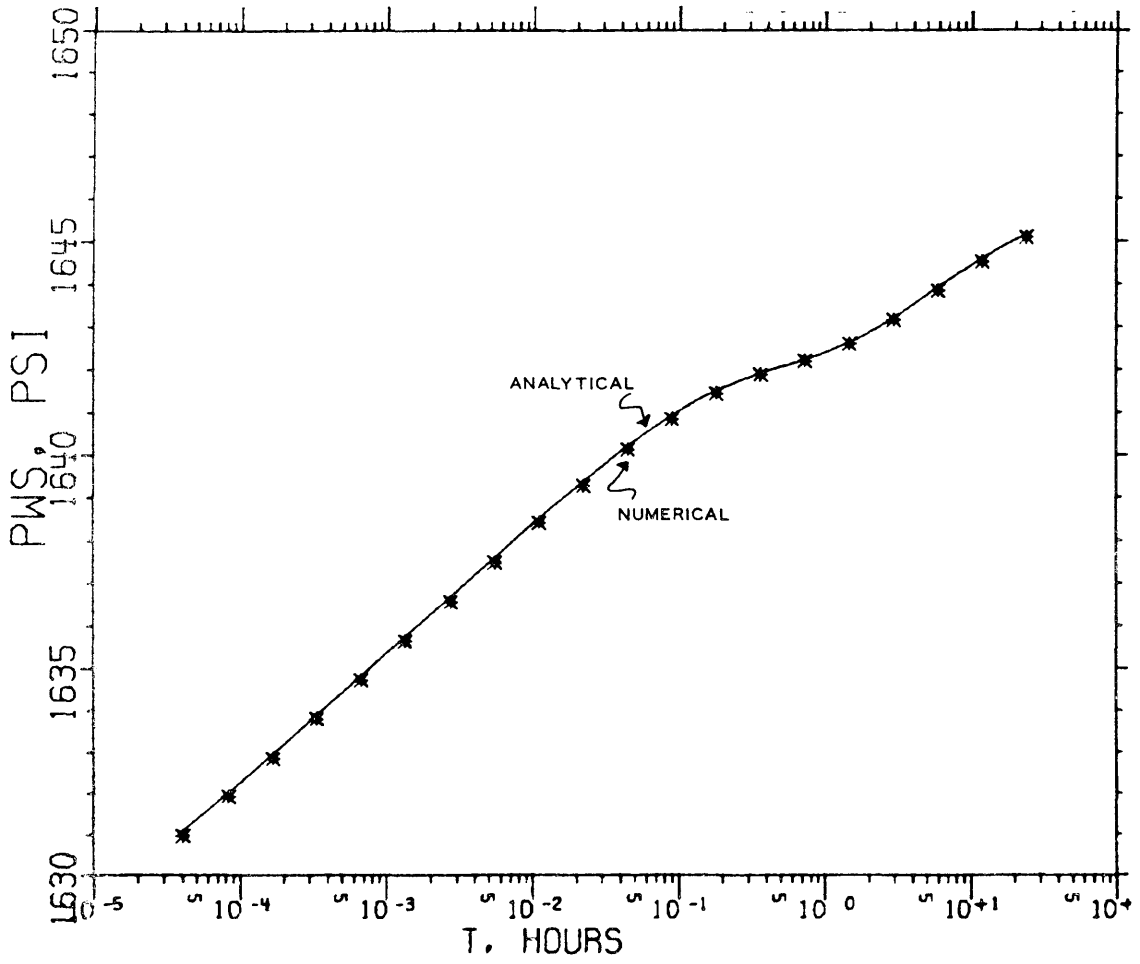

Figure 9. Analytical and Numerical Comparison for a Single-Phase Buildup in a Double-Porosity System

Linear Two-Phase Models

Linear two-porosity and single-porosity models were compared by Gilman and Kazemi (4) to show that the double-porosity model was valid. Data for vertical and horizontal double-porosity models (diagrams on Fig. 10 and 11) are given in Table 2. The simulator given in this thesis was compared to the reference to ensure that the simulator was working properly. The results should be identical since the equations are the same for single matrix sub-domains. Figures 10 and 11 show the cumulative oil produced and producing water-oil ratio (WOR) for the simulator in this thesis and the simulator of Reference 4. Equations were used for relative permeability and capillary pressure in this thesis while tables were used in the literature examples; therefore, there is some difference in the results. All other data are identical.

Buckley-Leverett Comparison

A single-porosity, linear, horizontal, two-phase model with water displacing oil was simulated by setting matrix properties to zero. The saturation distribution at 300 days for a 40 node model was compared to the analytical solution (by the well-known Buckley-Leverett Method) for the same system (Table 3 & Figure 12). The numerical solution cannot

reproduce the sharp front for the grid systems shown because of numerical dispersion effects. Use of larger time steps or large grid blocks increases the truncation error as shown in Figure 12. In the reservoir there is true physical dispersion and viscous fingering that prevents sharp fronts. The use of many more grid blocks and smaller time steps could give very close agreement between the numerical and analytical solutions.

It is interesting to show a saturation distribution for a two-porosity system with the same storage capacity. The fracture saturation has a very sharp gradient, while the matrix maintains a nearly constant profile behind the flood front. The data used is in Table 4 and saturations are plotted in Figure 12. The results are in qualitative agreement for a two-porosity analytic solution (22). Saturation distributions for a two-porosity vertical system are shown in a later section.

Capillary pressure was zero for the single-porosity models because that is the assumption used in the Buckley-Leverett solution. For the two-porosity models, a matrix capillary pressure curve greater than zero was used to ensure a large imbibition force. An exponential time function is used to describe imbibition for the analytical solution.

Table 2

Two-Phase, Double Porosity Model Data for Program Verification

P_{wmi}	1479 psi
P_{wfi}	1463 psi
S_{wmi}	0.08
S_{wfi}	0.01
μ_w	0.8 cp
μ_o	40 cp
c_ϕ	3×10^{-6} psi^{-1}
c_w	3.5×10^{-6} psi^{-1}
c_o	1.36×10^{-5} psi^{-1}
B_w	1.0 at 15 psi
B_o	1.036 at 15 psi
ρ_w	62.4 lb/ft^3
ρ_o	58.0 lb/ft^3
S_{wcf}	0.01
S_{orf}	0.0
S_{wcma}	0.08
S_{orma}	0.451
k_{rwf}	$1.0\ (S_{wfD})^{1.0}$
k_{rof}	$0.99\ (S_{orD})^{1.0}$

Continued...

Table 2 (continued)

k_{rwma}	$0.1072\ (S_{wmD})^{1.029}$
k_{roma}	$0.589\ (S_{orD})^{4.831}$
P_{cf}	$2.14\ (S_{WfD})^{1.929}$
Pc_{ma}	$13.36\ (S_{wmD})^{1.72}$
σ	$0.0002\ ft^{-2}$
ϕ_{ma}	0.1392
k_{ma}	100 md
Δx	10 ft
Δy	100 ft
Δz	102.04 ft
ϕ_f	0.02
k_f	10,000 md
r_w	0.34 ft
Q_T	500 STB/D 0-1600 days
	0 STB/D 1600-6400 days
Δt	0.01, 0-0.1 days
	0.1, 0.1-1.0 days
	1.0, 1.0-10 days
	10, 10-100 days
	100, 100-6400 days

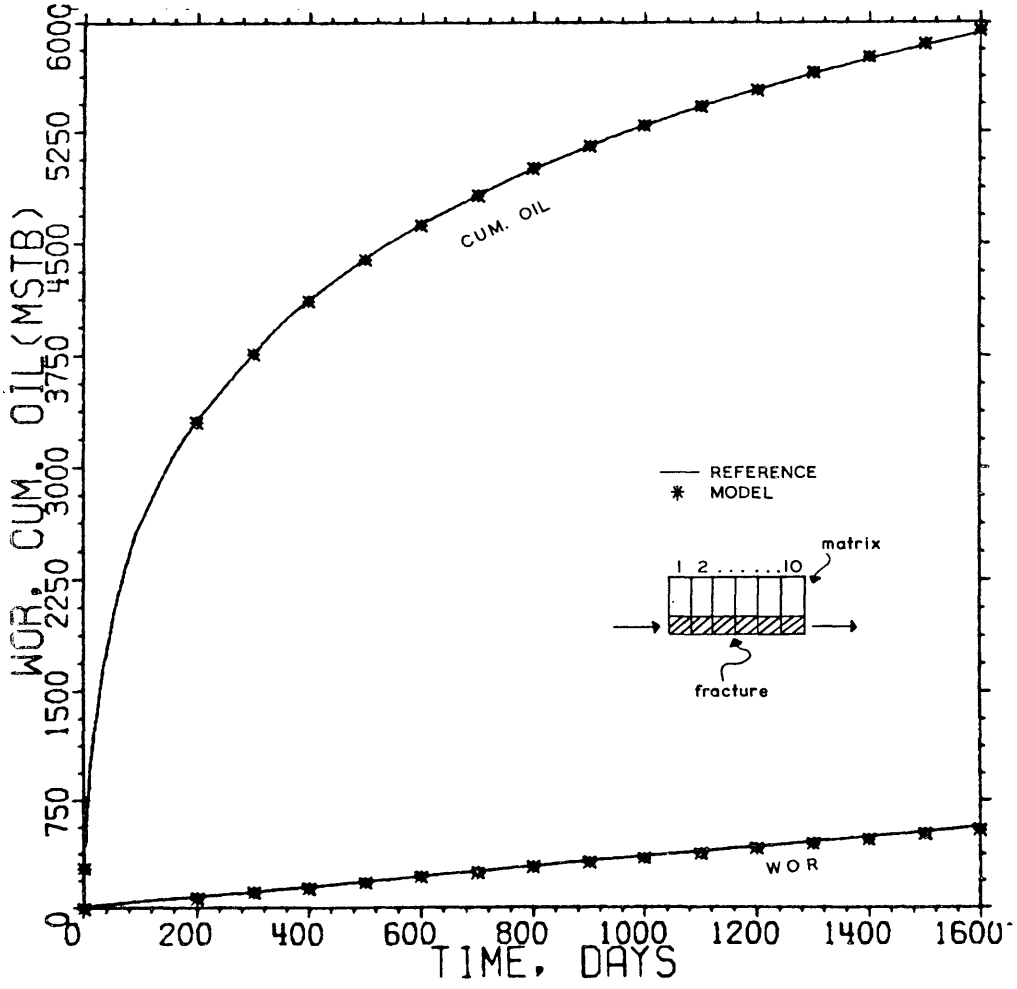

Figure 10. Comparison of Simulator with Literature Horizontal Model

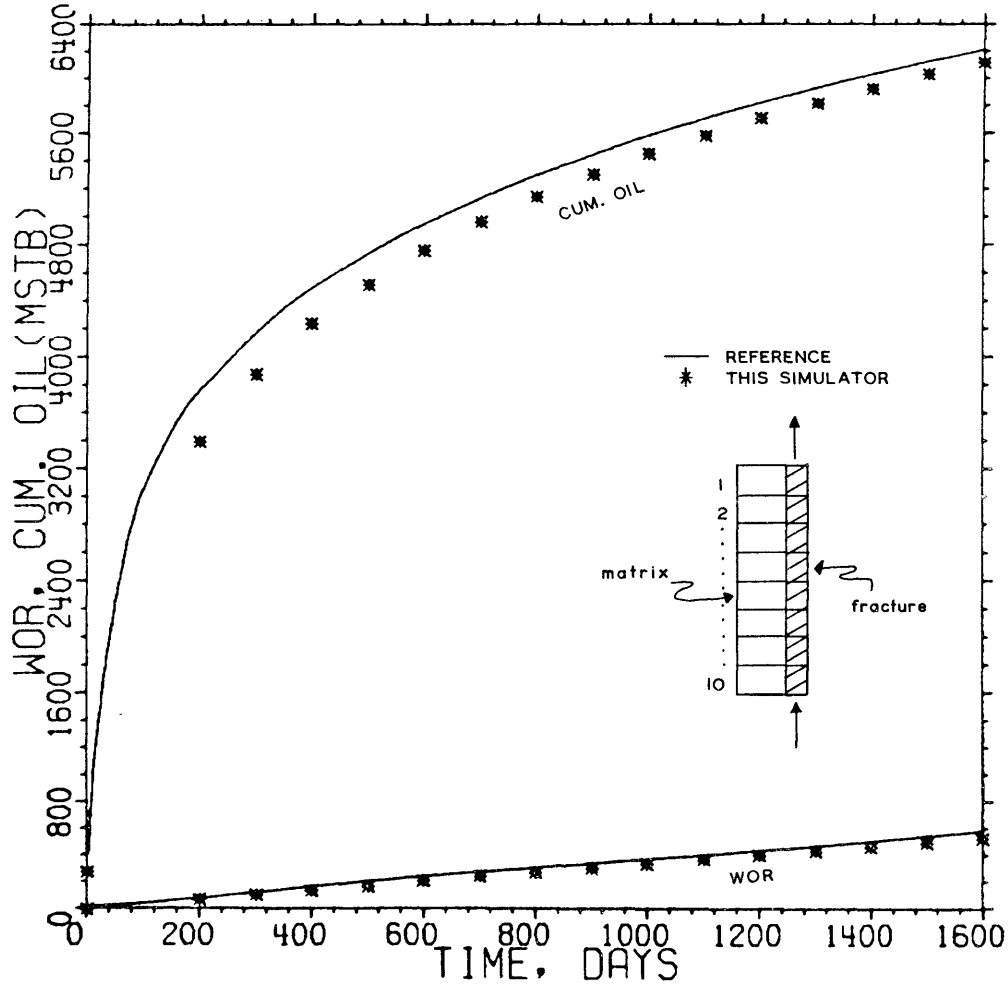

Figure 11. Comparison of Simulator with Literature Vertical Model

Table 3.

Buckley Leverett Model Data

p_{wi}	409.54 psi
S_{wi}	0.16
c_{ϕ}	3×10^{-6} psi^{-1}
$c_{w,o}$	3×10^{-6} psi^{-1}
$B_{w,o}$	1 res bbl/STB @ 15 psi
μ_o	4 cp
μ_w	1 cp
$\Delta z, \Delta y$	100 ft
k	300 md
ϕ	0.20
S_{wc}	0.16
S_{or}	0.20
k_{rw}	$0.4\ (S_{WD})^{1.2}$
k_{ro}	$0.9\ (S_{OD})^{2.0}$
q	76 STB/D
Δt	5, 50 days
Δx	10, 25 ft
r_w	0.25 ft

Table 4.

Two Porosity Buckley-Leverett Model Data

k_{rwf}	$1.0\ (S_{wDf})^{1.0}$
k_{rof}	$1.0\ (S_{oDf})^{1.0}$
S_{wcf}	0.01
S_{orf}	0.0
P_{cf}	0.0
P_{cm}	$5.0\ (S_{wDm})^{1.8}$ psi
σ	0.0016 ft^{-2}
ϕ_m	0.13
k_m	300 md
ϕ_f	0.01
Δz	100 ft
Δy	100 ft
Δx	25 ft
k_f	100000 md
P_{wmi}	409.18
P_{wfi}	414.18
S_{wmi}	0.16
S_{wfi}	0.01
Δt	5 days

Fluid properties are those in Table 3.
Matrix relative permeability is in Table 3.

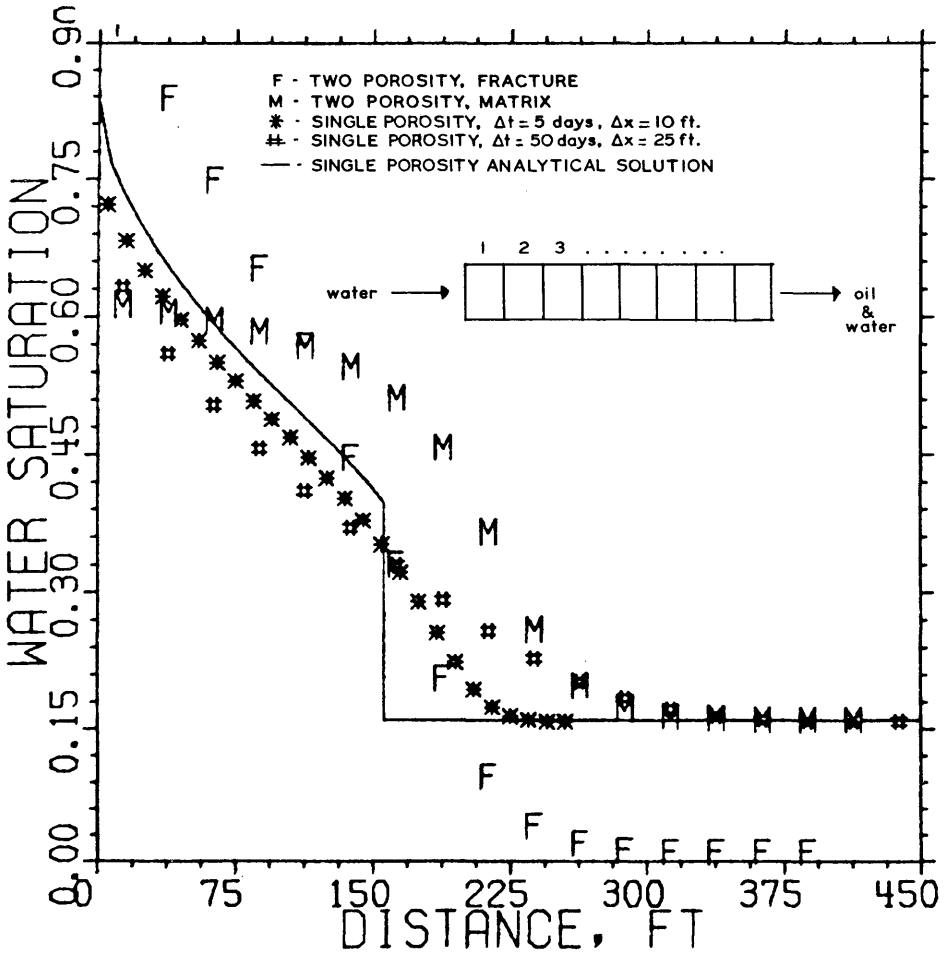

Figure 12. Comparison of Numerical Solution with Buckley-Leverett Solution

Laboratory Comparison

A laboratory core flood presented by Kazemi and Merrill (11) was simulated with the double-porosity model using one matrix block per grid node. The authors used a seven layer single-porosity model to simulate the system. The data used is that given by the authors, but adjusted for the two-porosity formulation as given in Table 5. Very small time-steps were used to reduce numerical dispersion effects. The comparison of pore volumes oil produced and producing water-oil ratio (WOR) between this model and the laboratory data are shown in Figure 13. The close agreement shows that the double-porosity formulation is valid and can use a greatly reduced grid system compared to a single-porosity formulation.

Timing Comparison

Comparison of the relative speed of this computer program with several matrix blocks compared to one block were made to show that solution time does not become prohibitive as more blocks are added. The ten node, two-porosity model shown by the diagram in Figure 10 was used in this comparison. For a single sub-domain per grid block system, there will be 40 unknowns. For four sub-domains per grid block, the number of unknowns becomes 100 or 2.5 times the number

for one block. The actual increase in time required was found to be 3.01 -- approximately the increase in number of unknowns. (Times were 1.26 and 3.79 seconds per time step for the two systems). The method of solution causes the time required to solve the equations to be proportional to the total number of matrix sub-domains. In conventional simulators, increasing the number of nodes could cause solution time to go up much more rapidly than linearly with the number of unknowns because the equation matrix bandwidth will increase and solution time is proportional to bandwidth squared. If a five layer conventional model was used to simulate the four sub-domains per grid block fracture-matrix system, the bandwidth squared would be 529 compared to 49 for the two-porosity system.Bandwidths for 2 unknowns and 2-dimensions is equal to 4*NM+3, where NM is the minimum dimension (18). The equation solution time could increase by the ratio 529:49 (10.8). In multidimensional systems it would be impractical to use a conventional simulator to model a two-porosity system because of the many grid blocks required. In systems with large matrix blocks where gravity segregation is important, it would be inefficient to use either a single-porosity model or a standard two-porosity model.

Table 5.

Model Data for Laboratory Core Flood Comparison

IMAX	10
μ_w	1 cp
μ_o	4.6 cp
c_ϕ	3×10^{-6} psi^{-1}
$c_{w,o}$	0
ρ_w	62.4 lb/ft^3
ρ_o	51.7 lb/ft^3
S_{wc}	0.43
S_{or}	0.34
k_{rwf}	$0.57(S_{wof})^{1.0}$
k_{rof}	$0.57(S_{oDf})^{1.0}$
k_{rwm}	$1.0\ (S_{wDm})^{3.0}$
k_{rom}	$1.0\ (S_{oDm})^{3.0}$
P_{cf}*	$0.75\ (S_{wDf})^{2.0}$ psi
P_{cm}*	$1.3\ (S_{wDm})^{1.0}$ psi
Pwmi	11.67 psi
σ	294.92 ft^{-2}
ϕ_m	0.1744
k_m	56 md

Continued...

Table 5 (continued)

k_f	10000 md
Δz	0.1647 ft
Δy	0.1647 ft
Δx	0.0335 ft
$q_{w1,10}$	0.000787 STB/D
r_w	0.042 ft
Δt_i	1×10^{-5} days (50 exponential steps to 0.002 days)
Δt	0.001 days, 0.01 to 0.24 days

S_{wom} and S_{wDf} are chosen so that P_{cm} = 0.9 psi at residual oil

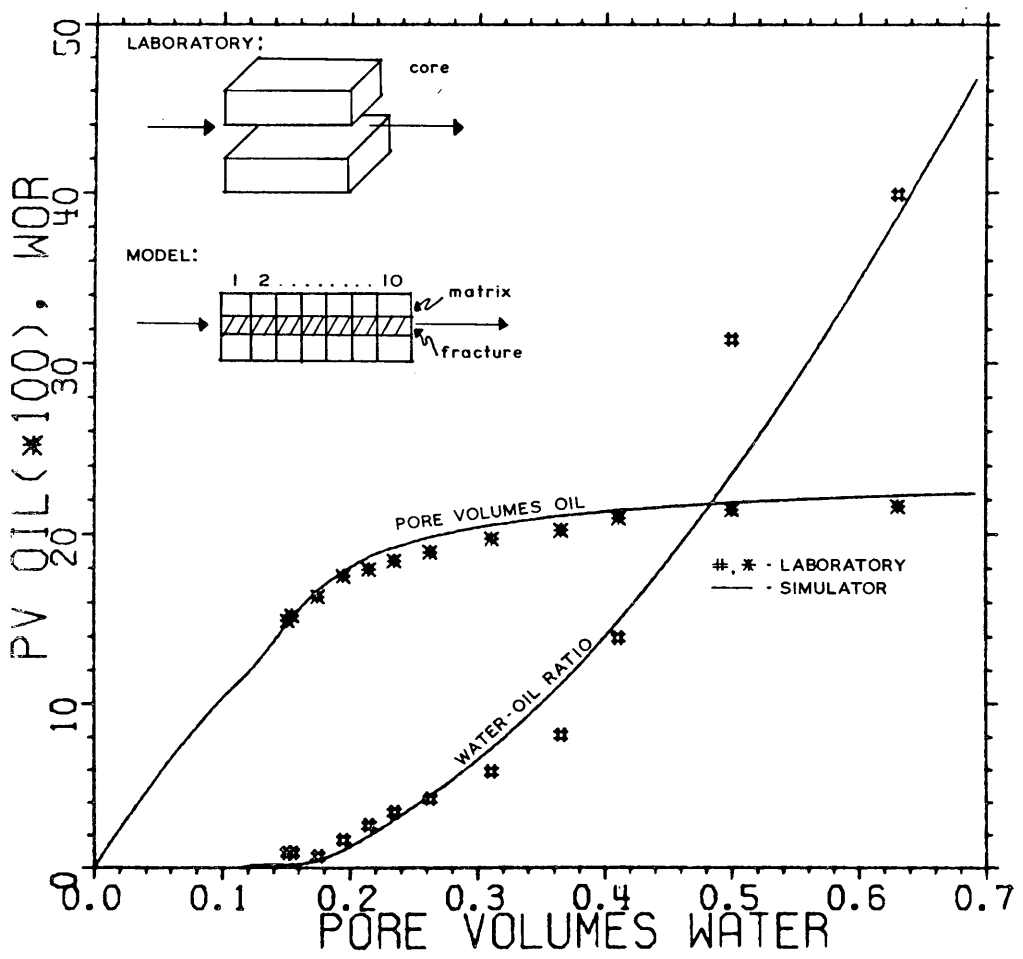

Figure 13. Comparison of Model With Kazemi's Laboratory Results

Examples of Program Use

Single Phase Transient Flow

Various authors (23, 24) have shown that the assumption of pseudo-steady state flow from matrix to fracture as is done for single matrix block systems can result in some inaccuracy when simulating pressure transient response because transient phenomenon within the matrix is not accounted for. They have shown that the transitional pressure response shown in Figures 8 and 9 should have a slope one-half that of the early and late time slopes. The one-half slope is because of the assumptions used in the double-porosity formulation, but the slope should be greater than that for the single domain system. The same system given in Table 1 was simulated using three matrix sub-domain per node in different geometric arrangements. Nested or stacked blocks (see Figure 4) were simulated. For the stacked blocks, fractures were assumed to be in one or three dimensions about the matrix block. The fracture spacing parameters were calculated using Equation 17. For one-dimensional flow from matrix to fractures, the L_y and L_z terms are omitted (4). Figure 14 shows pressure drawdown response for the different systems. The slopes of the transitional curves are greater than that for the single-subdomain system as expected. The

fracture spacing parameter varies as the number or geometry of the matrix subdomain varies. The value of the spacing parameter for each arrangement must be derived as given in Reference 4. Values for σ for the nested blocks, 1-D stacked blocks and 3-D stacked blocks are 0.00065, 0.0072, and 0.00107 ft^{-2} respectively for the outer subdomains. Those domains not connected to fractures have a spacing parameter of zero. To determine the apparent geometry and size of the matrix block in a reservoir, the simulator could be used to match a field pressure transient response. Note that only the transitional period between the early and late time straight lines is affected and thus this data is needed to determine matrix properties from a field test.

For this example, the nested matrix blocks are probably the most realistic for representing the true physics of the flow and therefore should be used when comparing the numerical solution to the analytic solution. The other geometries are shown only for comparison. The size of the matrix blocks is the same for all three examples, but the intra-matrix and matrix-fracture connection is different. For the stacked blocks, the actual pressure distribution in the matrix will not correspond to the sub-domain boundaries chosen. The two-phase examples which follow show where a stacked block representation of the matrix is useful.

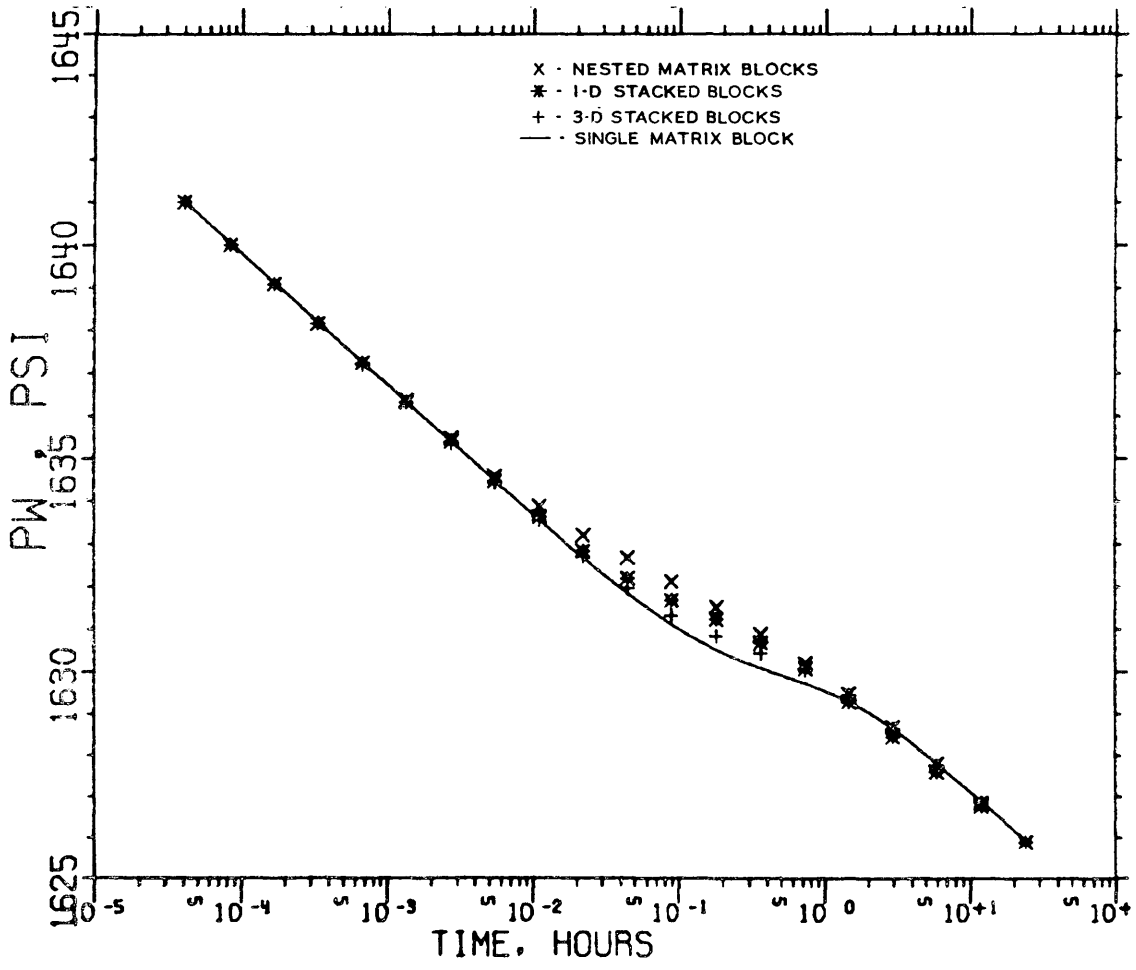

Figure 14. Pressure Drawdown for Single-Phase, Multiple Matrix Sub-Domain System

Phase Segregation in the Matrix Blocks

Horizontal System. The horizontal literature example given in Table 2 was simulated with four matrix sub-domains per node to show phase segregation in the matrix blocks. The system was simulated with four stacked blocks above the fracture. Only one block was connected to the fracture (Fig. 15) and had a spacing parameter of 0.0032 ft^{-2} as calculated from Equation 17. The saturations versus time in the top and bottom blocks of the first node are compared to a single sub-domain system in Figure 15. Fracture saturations in the first grid block are also shown. The models were not produced from 1600 to 6400 days to show the phase segregation and imbibition effects in the model. Imbibition caused by capillary forces will continue to draw water into the matrix during shutin forcing some oil into the fractures, but gravity forces will tend to cause oil to rise in the matrix away from the fractures. Because of the slow rate of fluid transfer, the system has not stabilized even after 4800 days of shutin. The multiple block systems can be used to give additional insight into the behavior of two-porosity reservoirs. Boundary conditions on the matrix blocks will also be more closely approximated for the multiple block systems.

Vertical System. The vertical literature example was also simulated using four matrix sub-domains per grid block-The saturations of the outer and inner blocks and the fracture for the first and last node are shown in Figure 16. Again, only the inner sub-domains were connected to the fracture and had the same spacing parameter as given above. In the vertical system there are no gravity forces working to distribute fluids in the sub-domains, because the sub-domains are all at the same elevation for a given node, and they are not directly connected to the matrix of the other grid blocks. The system arrangement may not be physically realistic, but it does show the effect of imbibition in the absence of gravity forces. Water continues to imbibe from the fracture into the matrix during well shutin, forcing oil into the fractures. Gravity forces in the fracture will cause oil to rise to the top of the fracture as shown by the decrease and then increase in water saturation for the bottom fracture node after the well is shut in.

Because of the similarity of the matrix saturation behavior for the vertical and horizontal systems, the imbibition force is much greater than the gravity force in this case. For oil-gas systems, results may be quite different. The standard two-porosity model could not show the phase segregation effects without using more grid blocks.

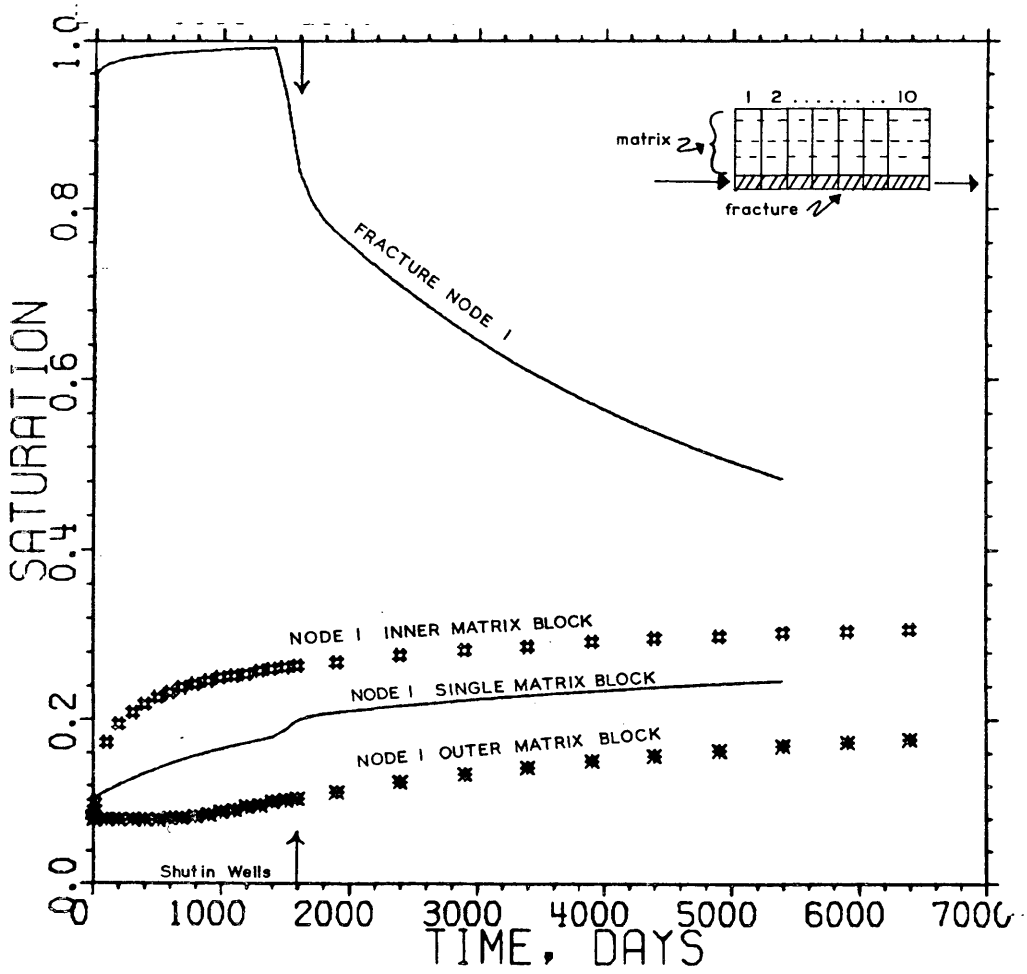

Figure 15. Saturations Versus Time for Horizontal Two-Porosity System

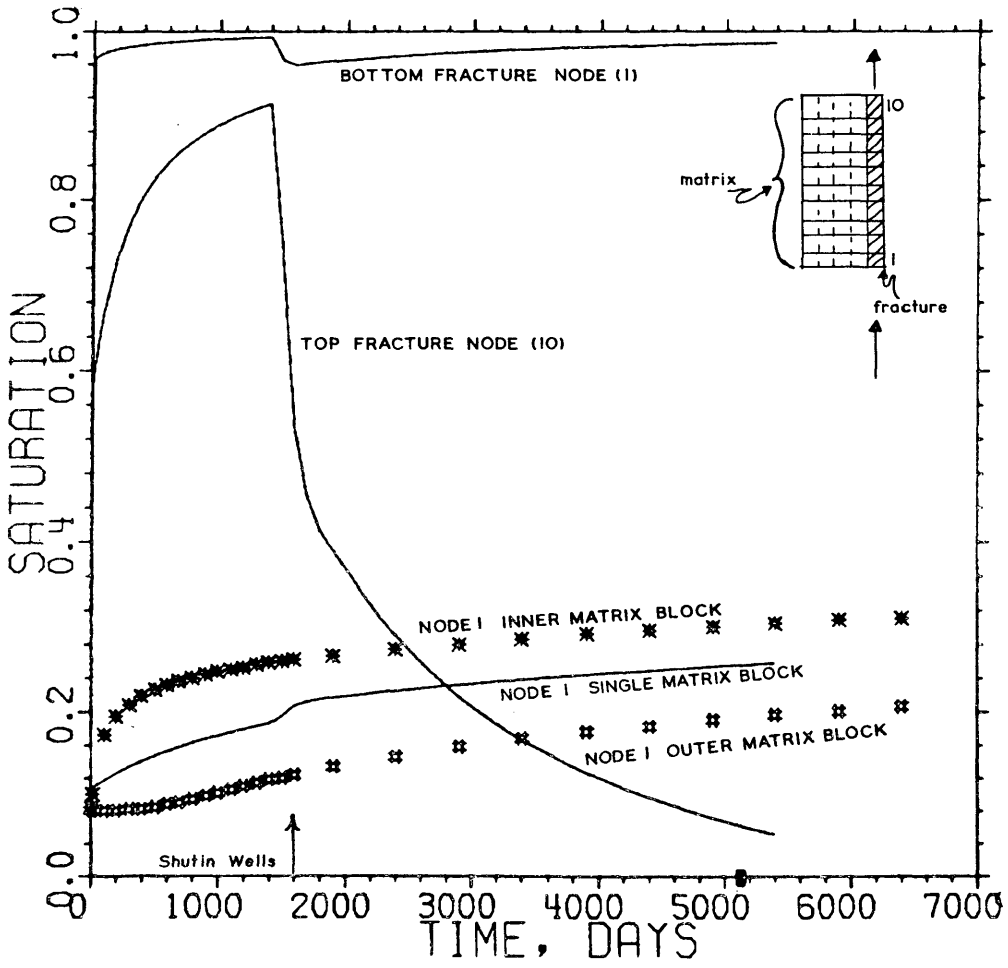

Figure 16. Saturations Versus Time for Vertical Two-Porosity System

Laboratory Simulation

The laboratory core flood described earlier was matched with one-matrix sub-domain per grid block. The matrix above and below the fracture (Fig. 13) was represented as one larger sub-domain at the same average height as the fracture. If the same system is simulated with one matrix sub-domain above and one below the fracture (to correspond to the true system geometry), the difference in saturation distribution in the top and bottom matrix rock can be seen. After 0.23 pore volumes of water are injected, the saturation distributions are as shown in Figure 17. These distributions agree with the profiles given in Reference 11. In this case there is very little additional information gained by using more matrix sub-domains because of the nature of the problem. Gravity forces are small. Using multiple matrix sub-domains can give some insight into the behavior of the fractured system as shown by the examples in this section. The finite difference method presented should also be applicable to other phenomena in fractured media such as heat flow or chemical transport.

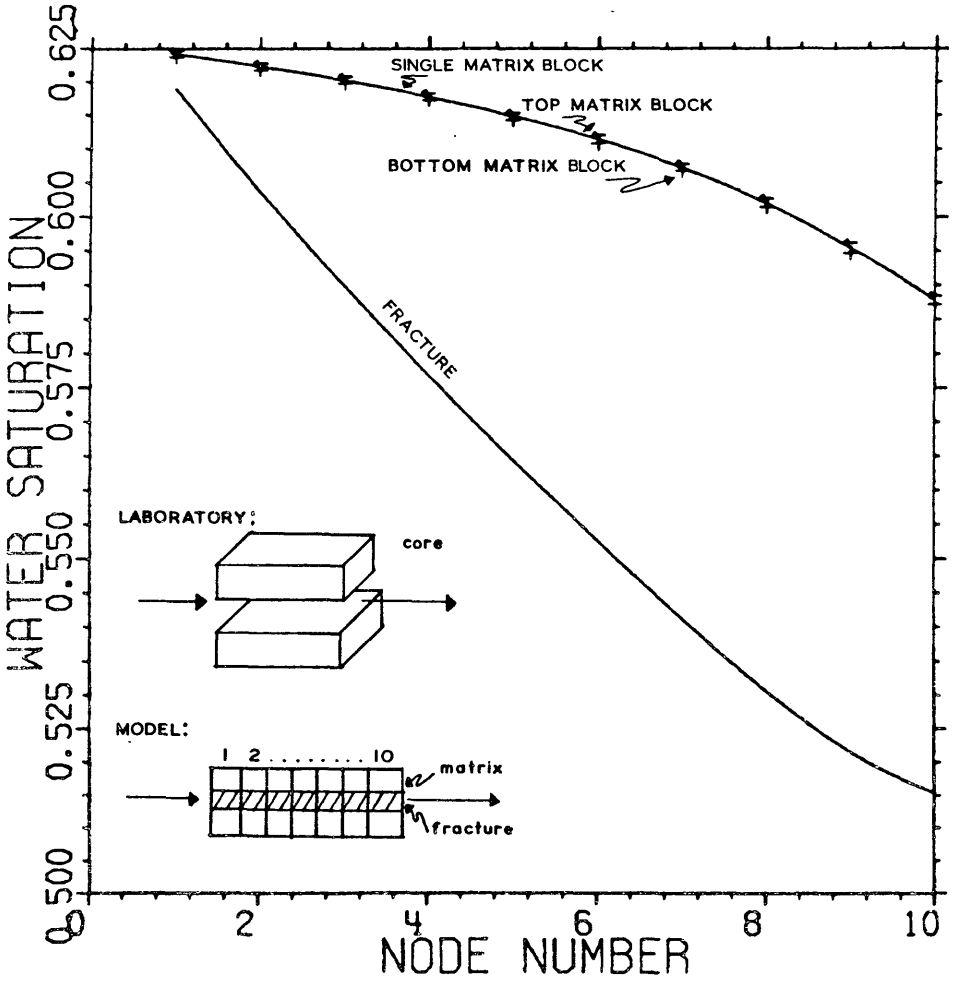

Figure 17. Saturation Distribution in Laboratory Simulation

Conclusions

Development of an efficient finite difference method for simulating unsteady state multiphase flow in the matrix blocks of naturally fractured systems led to the following conclusions:

1. The double-porosity formulation can use a greatly reduced grid system compared to a single-porosity representation of a fractured system.
2. Solution time is directly proportional to the number of matrix sub-domains.
3. The use of multiple matrix sub-domains in single-phase transient flow gives a greater semilog pressure plot slope during the transition from early to late time response than the use of a single matrix sub-domain.
4. Use of multiple matrix sub-domains can be used to show phase segregation in the matrix blocks of two-porosity systems without increasing the number of grid blocks.

Nomenclature

A	coefficient matrix
Area	flow areas between nodes; ft^2, m^2
b	right-hand side vector
B	formation volume factor; res bbl/STB, res m^3/m^3
B1	coefficient sub-matrix
B2	coefficient sub-matrix
B3	coefficient sub-matrix
B4	coefficient sub-matrix
B5	coefficient sub-matrix
c	compressibility; psi^{-1}, Pa^{-1}
C	coefficient sub-matrix
D	coefficient sub-matrix
DT	time step size variable; days
E	coefficient sub-matrix
EM	negative capillary pressure equation coefficient
EO	oil relative permeability equation coefficient
EP	positive capillary pressure equation coefficient
EW	water relative permeability equation coefficient
g	acceleration of gravity; ft/sec^2, m/sec^2
IMAX	number of nodes in x-direction
ITRN	iteration number
ITSN	time-step number
k	permeability; md, m^2
k_r	relative permeability; dimensionless
L	matrix block edge length., m
NIG	number of Newton-Raphson iterations allowed
NMB	number of matrix sub-domains per grid block
NUMTS	number of exponentially distributed time-steps
p	pressure; psi, Pa
P_c	capillary pressure; psi, Pa
PI	wellbore productivity index; md-ft, m^3
q	wellbore flow rate; STB/D, m^3/sec
q	wellbore flow rate per unit volume; day^{-1}, sec^{-1}
Q_T	total production or injection rate; STB/D, m^3/sec
r	radius; ft, m
R	residual
s	skin factor; dimensionless
S	phase saturation; fraction
t	time; days, sec.
T	transmissibility, STB/psi-day, m^3/Pa-sec
TMAX	maximum time for current time-steps; day, sec
TYME	program time variable; days
u	interstitial velocity; ft/day, m/sec
v	Darcy velocity; ft/day, m/sec
V	bulk rock volume; STB, m^3

x	vector of unknowns

Greek

γ	gravity gradient; psi/ft, Pa/m
δ	change in unknown value
Δ	$p_{i+1}-p_i$ or p_i-p_{i-1}
ΔL	distance between grid points
Δt	time step size; days, secs
Δu	radial grid exponential factor
Δu_t	exponential time-step factor
Δx	grid block size in x-direction; ft, m
Δy	grid block size in y-direction; ft, m
Δz	grid block size in z-direction; ft, m
λ	conversion factor for units used
μ	viscosity; cp, Pa•s
ρ	density, lb/ft^3, Kg/m^3
σ	fracture spacing parameter, ft^{-2}, m^{-2}
γ	matrix-fracture source term; day^{-1}, sec^{-1}
τ	matrix-fracture flow rate; STB/day, m^3/sec
ϕ	porosity; fracture
Φ	fluid potential given by Equation 2
ω	upstream weighting factor

Subscripts

1	first node, phase 1
2	second node, phase 2
b	boundary, base
D	dimensionless
e	external
f	fracture
i	node index, initial value
ir	irreducible
j	matrix block number
l	layer
ma	matrix
o	oil
r	reservoir, residual, relative
s	surface
w	water, well
WL	well
x	direction index
y	direction index
z	direction index

Greek Subscripts

α	phase
ϕ	rock

Superscripts

ℓ	iteration level

References Cited

1. Warren, J. E. and Root, P. J.: "The Behavior of Naturally Fractured Reservoirs," Soc. Pet. Eng. J., Vol. 3, No. 3, pp 245-255, (September 1963) .

2. Darcy, Henry: Les Fontaines Publiques de la Ville de Dijon, Victor Dalmot, Paris (1856).

3. Hubbert, M. King: "Darcy's Law and the Field Equations of the Flow of Underground Fluids," Petroleum Transactions of the AIME, Vol 207, pp 222-239, (1956).

4. Gilman, J. R. and Kazemi, H.: "Improvements in Simulation of Naturally Fractured Reservoirs," paper SPE 10511, presented at the Sixth SPE Symposium on Reservoir Simulation of the Society of Petroleum Engineers of AIME, New Orleans, LA. pp 293-315, (Jan. 31-Feb. 3, 1982)

5. Barenblatt, G. D., Zhelton, K. R., Kochina, I. N.: "Basic Concepts in the Theory of Homogeneous Liquids in Fissures Rocks," Journal of Applied Mathematics (USSR), Vol. 24, No. 5, pp 1286-1303 (1960).

6. Odeh, A. S.: "Unsteady State Behavior of Naturally Fractured Reservoirs," Soc. Pet. Eng. J., pp 60-65, (March 1965), Vol. 5, No. 1.

7. Kazemi, H., Merrill, L. S., Porterfield, K. L., and Zeman, P. R.: "Numerical Simulation of Water-Oil Flow in Naturally Fractured Reservoirs," Soc. Pet. Eng. J., Vol. 16, No. 6, pp 317-326, (December 1976)

8. Duguid, J. O., Lee, P.C.Y.: "Flow in Fractured Porous Media," Water Resources Research, Vol. 13, No. 3, pp 558-566, (June, 1977)

9. Witherspoon, P. A., Wang, J.S.Y., Iwai, K., Gale, J. E.: "Validity of Cubic Law for Fluid Flow in a Deformable Rock Fracture," Water Resources Research, Vol. 16, No. 6, pp 1016-1024, (Dec. 1980)

10. Rossen, R. H.: "Simulation of Naturally Fractured Reservoirs with Semi-Implicit Source Terms," Soc. Pet. Eng. J., Vol. 17, No. 3, pp 201-210, (June 1977)

11. Kazemi, H., Merrill, L. S.: "Numerical Simulation of Water Imbibition in Fractured Cores," Soc. Pet. Eng. J., Vol. 16, No. 3, pp 175-182, (June 1976)

12. Odeh, A. S.: "An Overview of Mathematical Modeling of the Behavior of Hydrocarbon Reservoirs," SIAM Review, Vol. 24, No. 3, pp 263-273, (July 1982)

13. Peaceman, D. W.: "Interpretation of Well-Block Pressures in Numerical Reservoir Simulation With Non Square Grid Blocks and Anisotropic Permeability," Paper SPE 10528, Sixth SPE Symposium on Reservoir Simulation, New Orleans, La., pp 553-570 (Feb. 1-3, 1982).

14. Settari, A., Aziz, K.: "Treatment of Nonlinear Terms in the Numerical Solution of Partial Differential Equations for Multiphase Flow in Porous Media," Int. J. Multiphase Flow, Vol. 1, pp 817-844 (1975).

15. Pruess, K., Narasimham, T. N.: "A Practical Method for Modeling Fluid and Heat Flow in Fractured Porous Media," paper SPE 10509 presented at the Sixth SPE Symposium on Reservoir Simulation of the Society of Petroleum Engineers of AIME, New Orleans, La., pp 269-284, Jan. 31-Feb. 3, 1982

16. Eica, A. S., Ershaghi, I.: "Modeling of Gas Injection in Fractured Reservoirs," paper SPE 10758 presented at the 1982 California Regional Meeting of the Society of Petroleum Engineers, San Francisco, Ca., (March 24-26, 1982).

17. Papay, J.: "Numerical Modeling of Mixed Porosity Reservoirs - Part I - Descrition of the Model", KOOLAJ FOLDGAZ Vol. 14 No. 114, pp 372-376, (December, 1982)

18. Aziz, K. and Settari, A.: Petroleum Reservoir Simulation, Applied Science Publishers, Vol. 1 (1979).

19. Thomas, L. K., Dixon, T. N., and Pierson, R. G.: "Fractured Reservoir Simulation," Soc. Pet. Eng. J., Vol. 23, No. 1, pp 42-54, (Feb. 1983).

20. Cheshire, I. A., Appleyard, J. R. and Banks, D.: "An Efficient Fully Implicit Simulator," paper EUR 179 presented at the European Offshore Petroleum Conference and Exhibition, London, England, (October 21-24, 1980).

21. Bansal, P. P., Harper, J. L., McDonald, A. E., Moreland, E. E., Odeh, A. S.: "A Strongly Coupled, Fully Implicit Three-Dimensional, Three-Phase Reservoir Simulator," paper SPE 8329 presented at the 54th Annual Fall meeting of the Society of Petroleum Engineers, (Sept. 23-26, 1979).

22. Chen, Z., and Liu, C.: "Theory of Fluid Displacement in a Medium with Double Porosity," Energy Sources, Vol. 6., No. 3, pp 193-214 (1982).

23. Sena, K. V., Reynolds, A. C., Raghavan, R.: "New Pressure Transient Analysis Methods for Naturally Fractured Reservoirs," paper SPE 10780 presented at the 1982 California Regional Meeting of the Society of Petroleum Engineers, San Francisco, Ca., (March 24-26,1982).

24. Streltsova, T. D.: "Well Pressure Behavior of a Naturally Fractured Reservoir", paper SPE 10782 presented at the 1982 California Regional Meeting of the Society of Petroleum Engineers, San Francisco, Ca., (March 24-26, 1982).

APPENDIX A

PROGRAM LISTING

```
10006000 C
10007000 C     PROGRAM CSM.  VERSION 7/27/1982.  J. R. GILMAN.   800700.
10008000 C
10009000 C      THIS IS A PROGRAM TO SIMULATE OIL-WATER FLOW IN NATURALLY
10010000 C       FRACTURED OR OTHER HETEROGENEOUS DOUBLE-POROSITY RESERVOIRS.
10011000 C       THE PRIMARY POROSITY CONTAINS THE BULK OF THE FLUID WHILE
10012000 C       THE SECONDARY POROSITY IS THE PRIMARY FLUID FLOW PATH.
10013000 C       THE PRIMARY(MATRIX) POROSITY IS NOT CONNECTED FROM GRID BLOCK
10014000 C       TO GRID BLOCK BUT IT CAN BE SUBDIVIDED INTO SEVERAL BLOCKS WITHIN
10015000 C       A GIVEN GRID BLOCK TO ALLOW TRANSIENT FLOW.
10016000 C
10017000 C
10018000 C     THE DATA REQUIRED IS AS FOLLOWS:
10018100 C      (THE CARD NAME MUST BE IN COL. 1-10. INPUT IS I10 OR F10.0
10018200 C       FORMAT WITH THE FIRST DATA IN COLUMNS 11-20.)
10019000 C
10020000 C      CARD 1 --
10021000 C
10022000 C       NWELL -- THE NUMBER OF WELLS
10023000 C       IMAX  -- THE NUMBER OF GRID BLOCKS
10024000 C       NMB   -- THE NUMBER OF MATRIX BLOCKS WITHIN A GRID BLOCK
10025000 C
10026000 C      CARD 2 --
10027000 C
10028000 C       DWOC -- DEPTH OF WATER OIL CONTACT,FT. (POSITIVE DOWN)
10029000 C       PWOC -- PRESSURE AT WATER-OIL CONTACT,PSI.
10030000 C       PB   -- BASE PRESSURE FOR FLUID PRESSURES,PSI.
10030100 C       RE   -- EXTERNAL RADIUS FOR RADIAL SYSTEMS. FT.
10030200 C       RW   -- WELLBORE RADIUS FOR RADIAL SYSTEMS. FT.
10031000 C
10032000 C      CARD 3 --
10033000 C
10034000 C       VIW -- WATER VISCOSITY. CP.
10035000 C       VIO -- OIL VISCOSITY. CP.
10036000 C       CF  -- FRACTURE ROCK COMPRESSIBILITY. 1/PSI.
10037000 C       CM  -- MATRIX ROCK COMPRESSIBILITY. 1/PSI.
10038000 C
10039000 C      CARD 4 --
10040000 C
10041000 C       CW    -- WATER COMPRESSIBILITY. 1/PSI.
10042000 C       CO    -- OIL COMPRESSIBILITY. 1/PSI.
10043000 C       BWB   -- WATER FORMATION VOLUME AT BASE PRESSURE. RES.BBL/STB
10044000 C       BOB   -- OIL FORMATION VOLUME AT BASE PRESSURE. RES.BBL./STB.
10045000 C       RHOWB -- WATER DENSITY AT BASE PRESSURE. LB/CU.FT.
10046000 C       RHOOB -- OIL DENSTIY AT BASE PRESSURE. LB/CU.FT.
10047000 C
10048000 C      CARD 5 --
10049000 C
10050000 C       EWF   -- EXPONENT FOR FRACTURE WATER REL. PERM. EQUATION
10051000 C       EOF   -- EXPONENT FOR FRACTURE OIL REL. PERM. EQUATION
10052000 C       SWCF  -- CONNATE WATER SATURATION IN FRACTURE
10053000 C       SORF  -- RESIDUAL OIL SATURATION IN FRACTURE
10054000 C       RKSWCF-- FRACTURE OIL REL. PERM. AT CONNATE WATER SATURATION
10055000 C       RKSORF-- FRACTURE WATER REL. PERM AT RESIDUAL OIL SATURATION
10056000 C
10057000 C      CARD 6 --
10058000 C
10059000 C       EWM   -- EXPONENT FOR MATRIX WATER REL. PERM. EQUATION
10060000 C       EOM   -- EXPONENT FOR MATRIX OIL REL. PERM. EQUATION
10061000 C       SWCM  -- CONNATE WATER SATURATION IN MATRIX
10062000 C       SORM  -- RESIDUAL OIL SATURATION IN MATRIX
10063000 C       RKSWCM-- MATRIX OIL REL. PERM. AT CONNATE WATER SATURATION
10064000 C       RKSORM-- MATRIX WATER REL. PERM AT RESIDUAL OIL SATURATION
10065000 C
10066000 C      CARD 7 --
10067000 C
10068000 C       EPF   -- EXPONENT FOR POSITIVE CAPILLARY PRESSURE IN FRACTURE
10069000 C       EMF   -- EXPONENT FOR NEGATIVE CAPILLARY PRESSURE IN FRACTURE
10070000 C       SPCZF -- WATER SATURATION FOR ZERO CAP. PRES. IN FRACTURE
10071000 C       PSWCF -- FRACTURE CAP. PRES. AT CONNATE WATER, PSI.
10072000 C       PSORF -- FRACTURE CAP. PRES. AT RESIDUAL OIL, PSI.
10073000 C
10074000 C      CARD 8 --
10075000 C
10076000 C       EPM   -- EXPONENT FOR POSITIVE CAPILLARY PRESSURE IN MATRIX
10077000 C       EMM   -- EXPONENT FOR NEGATIVE CAPILLARY PRESSURE IN MATRIX
10078000 C       SPCZM -- WATER SATURATION FOR ZERO CAP. PRES. IN MATRIX
10079000 C       PSWCM -- MATRIX CAP. PRES. AT CONNATE WATER. PSI.
10080000 C       PSORM -- MATRIX CAP. PRES. AT RESIDUAL OIL. PSI.
```

```
10081000 C
10082000 C     GRID1 (MUST HAVE IMAX*NMB OF THESE) --
10083000 C
10084000 C      I -- NODE NUMBER INDEX
10085000 C      J -- MATRIX BLOCK NUMBER INDEX
10086000 C      NOMAT -- INDICATES NODES WITH NO MATRIX
10087000 C      SIGMA -- FRACTURE SPACING PARAMETER, 1/ SQ. FT.
10088000 C      PHIBM -- MATRIX BASE POROSITY
10089000 C      AKM -- MATRIX PERMEABILITY, MD
10090000 C      CGMF -- GRAVITY COEFFICIENT (-0.5 < CGMF < 0.5)
10091000 C
10092000 C     GRID2 (MUST HAVE IMAX OF THESE) --
10093000 C
10094000 C      I -- NODE NUMBER INDEX
10095000 C      DX -- GRID BLOCK SIZE IN X-DIRECTION, FT.
10096000 C      DY -- GRID BLOCK SIZE IN Y-DIRECTION, FT.
10097000 C      DZ -- GRID BLOCK SIZE IN Z-DIRECTION, FT.
10098000 C      PHIBF -- FRACTURE BASE POROSITY
10099000 C      AKF -- FRACTURE PERMEABILITY, MD.
10100000 C      DF -- FRACTURE DEPTH, FT. (POSITIVE DOWN).
10101000 C
10102000 C     TIME (CAN HAVE AS MANY AS DESIRED) --
10103000 C
10104000 C      TSTART -- START TIME FOR NEW TIME STEPS, DAYS
10105000 C      DT -- TIME-STEP SIZE, DAYS
10106000 C      TFINIS -- ENDING TIME FOR CURRENT TIME STEPS, DAYS
10107000 C      NIG -- NUMBER OF NEWTON-RAPHSON ITERATIONS ALLOWED
10107500 C      IPFREQ -- TIME-STEP FREQUENCY FOR PRINTING DATA
10107600 C      NUMTS -- NUMBER OF TIME STEPS FOR EXPONENTIAL DISTRIBUTION
10108000 C
10109000 C     WELL (NEED ONE FOR EACH WELL, FOLLOWING EACH TIME CARD) --
10110000 C
10111000 C      WL -- WELL NUMBER
10112000 C      IW -- NODE NUMBER LOCATION OF WELL
10113000 C      RW -- WELLBORE RADIUS, FT.
10114000 C      SKIN -- WELLBORE SKIN FACTOR
10115000 C      QWELL -- TOTAL WELLBORE FLUID RATE, STB/DAY
10117000 C
10118000 C*************************************************************************
10119000 FILE  5=CARD,UNIT=READER
10120000 FILE  6=LINE,UNIT=PRINTER
10121000       COMMON /CONS/CW,CO,NWELL,DT,IMAX,NMB,VIC,VIW,RHOWB,RHOOB,CF,GM
10122000       COMMON /FRAC/BWF,BOF,DF,PWF,POF,PHIBF,PHIF,RKOF,RKWF,RKWPF,PCPF,
10123000      $               RKOPF,TCF,TOF,TWF,SWF,VR,WWF,WOF,RHOWF,RHOOF
10124000       COMMON /COEFF/A11,A12,A21,A22,B11,B12,B21,B22,C11,C12,C21,C22
10125000       COMMON /RHS/ RFW,RFO,RMW,RMO,RWL
10126000       COMMON /WELL/ PI,PBH,PIW,PIO,WW,W1,W2,WD,PWN
10127000       COMMON /MATRIX/BWM,BOM,DM,PWM,POM,PHIBM,PHIM,RKOM,RKWM,RKWPM,
10128000      $               RKOPM,TCM,TOM,TWM,SWM,WWM,WOM,RHOWM,RHOOM,PCPM,
10129000      $               TMF,TOMF,TWMF,WWMF,WOMF
10130000       DIMENSION A(600),D(200),X(200)
10131000       DIMENSION BWF(99),BOF(99),DF(99),PWF(99),POF(99),PCPF(99)
10132000       DIMENSION PHIBF(99),PHIF(99),RKOF(99),RKWF(99)
10133000       DIMENSION RKWPF(99),RKOPF(99),TCF(99),TOF(99),TWF(99)
10134000       DIMENSION SOF(99),SWF(99),VR(99),WWF(99),WOF(99),RHOWF(99)
10135000       DIMENSION RHOOF(99),A11(99),A12(99),A21(99),A22(99)
10136000       DIMENSION B11(99),B12(99),B21(99),B22(99),C11(99)
10137000       DIMENSION C12(99),C21(99),C22(99),WW(99),W1(99),W2(99),WD(99)
10138000       DIMENSION RFW(99),RMW(99),RFO(99),RMO(99),VPWFN(99),VPOFN(99)
10139000       DIMENSION PI(99),PBH(99),PIW(99),PIO(99),RWL(99),PWN(99)
10140000       DIMENSION BWM(99),BOM(99),DM(99),PWM(99),POM(99),RHOOM(99)
10141000       DIMENSION PHIBM(99),PHIM(99),RKOM(99),RKWM(99),PCPM(99)
10142000       DIMENSION RKWPM(99),RKOPM(99),TCM(99),TOM(99),TWM(99)
10143000       DIMENSION SOM(99),SWM(99),WWM(99),WOM(99),RHOWM(99)
10144000       DIMENSION TMF(99),TWMF(99),TOMF(99),WWMF(99),WOMF(99)
10145000       DIMENSION SIGMA(99),AKM(99),CGMF(99),NOMAT(99)
10146000       DIMENSION DX(99),DY(99),DZ(99),AKF(99),RAD(99)
10147000       DIMENSION IW(99),JW(99),RW(99),SKIN(99),QWELL(99)
10148000       DIMENSION VPWF(99),VPOF(99),VPWM(99),VPOM(99),PCF(99),PCM(99)
10149000       DIMENSION VPWFN(99),VPOFN(99),QW(99),QO(99),CUMW(99),CUMO(99)
10150000       LOGICAL CONVRG
$
```

```
10151000 C
10152000 C  READ INITIAL DATA CARDS
10153000 C
10154000       READ(5,20,ERR=800) NWELL,IMAX,NMB
10155000       WRITE(6,25) NWELL,IMAX,NMB
10156000       READ(5,30,ERR=800) DWOC,PWOC,PB,RE,RW1
10157000       WRITE(6,35) DWOC,PWOC,PB,RE,RW1
10158000       READ(5,40,ERR=800) VIW,VIO,CF,CM
10159000       WRITE(6,45) VIW,VIO,CF,CM
10160000       READ(5,50,ERR=800) CW,CO,BWB,BOB,RHOWB,RHOOB
10161000       WRITE(6,55) CW,CO,BWB,BOB,RHOWB,RHOOB
10162000       READ(5,50,ERR=800) EWF,EOF,SWCF,SORF,RKSWCF,RKSORF
10163000       WRITE(6,65) EWF,EOF,SWCF,SORF,RKSWCF,RKSORF
10164000       READ(5,50,ERR=800) EWM,EOM,SWCM,SORM,RKSWCM,RKSORM
10165000       WRITE(6,75) EWM,EOM,SWCM,SORM,RKSWCM,RKSORM
10166000       READ(5,60,ERR=800) EPF,EMF,SPCZF,PSWCF,PSORF
10167000       WRITE(6,85) EPF,EMF,SPCZF,PSWCF,PSORF
10168000       READ(5,60,ERR=800) EPM,EMM,SPCZM,PSWCM,PSORM
10169000       WRITE(6,95) EPM,EMM,SPCZM,PSWCM,PSORM
10170000    20 FORMAT(10X,3I10)
10171000    30 FORMAT(10X,5F10.0)
10172000    40 FORMAT(10X,4F10.0)
10173000    50 FORMAT(10X,6F10.0)
10174000    60 FORMAT(10X,5F10.0)
10175000    25 FORMAT(1X'CARD 1',4X,'NWELL =',I10,3X,'IMAX  =',I10,3X,
10176000      $  'NMB   =',I10,/)
10177000    35 FORMAT(1X'CARD 2',4X,'DWOC  =',F10.3,3X,'PWOC  =',F10.3,3X,
10178000      $                     'PB    =',F10.5,3X,'RE    =',F10.5,3X,
10178100      $                     'RW    =',F10.5,/)
10179000    45 FORMAT(1X'CARD 3',4X,'VIW   =',F10.5,3X,'VIO   =',F10.5,3X,
10180000      $  'CF    =',E10.5,3X,'CM    =',E10.3,/)
10181000    55 FORMAT(1X'CARD 4',4X,'CW    =',E10.5,3X,'CO    =',E10.5,3X,
10182000      $                     'BWB   =',F10.5,3X,'BOB   =',F10.5,3X,
10183000      $                     'RHOWB =',F10.5,3X,'RHOOB =',F10.5,/)
10184000    65 FORMAT(1X'CARD 5',4X,'EWF   =',F10.5,3X,'EOF   =',F10.5,3X,
10185000      $                     'SWCF  =',F10.5,3X,'SORF  =',F10.5,3X,
10186000      $                     'RKSWCF=',F10.5,3X,'RKSORF=',F10.5,/)
10187000    75 FORMAT(1X'CARD 6',4X,'EWM   =',F10.5,3X,'EOM   =',F10.5,3X,
10188000      $                     'SWCM  =',F10.5,3X,'SORM  =',F10.5,3X,
10189000      $                     'RKSWCM=',F10.5,3X,'RKSORM=',F10.5,/)
10190000    85 FORMAT(1X'CARD 7',4X,'EPF   =',F10.5,3X,'EMF   =',F10.5,3X,
10191000      $                     'SPCZF =',F10.5,3X,'PSWCF =',F10.5,3X,
10192000      $                     'PSORF =',F10.5,/)
10193000    95 FORMAT(1X'CARD 8',4X,'EPM   =',F10.5,3X,'EMM   =',F10.5,3X,
10194000      $                     'SPCZM =',F10.5,3X,'PSWCM =',F10.5,3X,
10195000      $                     'PSORM =',F10.5,/)
10196000       N = 0
10196500       NN = NMB * IMAX
10196600       IF (NN.EQ.0) GO TO 200
10197000 C
10198000 C  READ MATRIX GRID CARDS
10199000 C
10200000   100 CONTINUE
10201000       READ(5,110,ERR=800) I,J,M,VAL2,VAL3,VAL4,VAL5
10202000       N = N + 1
10203000       L = J + (I-1)*NMB
10204000       NOMAT(L) = M
10205000       SIGMA(L) = VAL2
10206000       PHIBM(L) = VAL3
10207000       AKM(L) = VAL4
10208000       CGMF(L) = VAL5
10209000       IF(N.LT.NN) GO TO 100
10210000   110 FORMAT(10X,3I10,4F10.0)
10211000       WRITE(6,115)
10212000   115 FORMAT(///40X,'NOMAT (NO MATRIX NODES)',/,120('*'))
10213000       CALL RITED(NOMAT,NMB,IMAX)
10214000       WRITE(6,125)
10215000   125 FORMAT(///40X,'SIGMA (1/FT**2)',/,120('*'))
10216000       CALL RITED(SIGMA,NMB,IMAX)
10217000       WRITE(6,135)
10218000   135 FORMAT(///40X,'PHIBM (FRACTION)',/,120('*'))
10219000       CALL RITED(PHIBM,NMB,IMAX)
10220000       WRITE(6,145)
10221000   145 FORMAT(///40X,'AKM (MD)',/,120('*'))
10222000       CALL RITED(AKM,NMB,IMAX)
10223000       WRITE(6,155)
10224000   155 FORMAT(///40X,'CGMF (DIMENSIONLESS)',/,120('*'))
10225000       CALL RITED(CGMF,NMB,IMAX)
$
```

```
C
C READ FRACTURE GRID DATA
C
      N = 0
 200  CONTINUE
      READ(5,210,ERR=800) I,VAL1,VAL2,VAL3,VAL4,VAL5,VAL6
      N = N + 1
      DX(I) = VAL1
      DY(I) = VAL2
      DZ(I) = VAL3
      VR(I) = VAL1*VAL2*VAL3/5.6146
      PHIBF(I) = VAL4
      AKF(I) = VAL5*VAL4
      DF(I) = VAL6
      IF (N.LT.IMAX) GO TO 200
      IF (RE.GT.0) CALL RADGRD(RW1,RE,IMAX,RAD,DX,DY,DZ,VR)
 210  FORMAT(10X,I10,6F10.0)
      WRITE(6,205)
 205  FORMAT(///40X,'DX (FT)',/,120('*'))
      CALL RITED(DX,1,IMAX)
      WRITE(6,215)
 215  FORMAT(///40X,'DY (FT)',/,120('*'))
      CALL RITED(DY,1,IMAX)
      WRITE(6,225)
 225  FORMAT(///40X,'DZ (FT)',/,120('*'))
      CALL RITED(DZ,1,IMAX)
      WRITE(6,235)
 235  FORMAT(///40X,'PHIBF (FRACTION)',/,120('*'))
      CALL RITED(PHIBF,1,IMAX)
      WRITE(6,245)
 245  FORMAT(///40X,'AKF (MD)',/,120('*'))
      CALL RITED(AKF,1,IMAX)
      WRITE(6,255)
 255  FORMAT(///40X,'DF (FT)',/,120('*'))
      CALL RITED(DF,1,IMAX)
C
C INTIALIZATION IS VERIFIED BY RUNNING PROGRAM WITH NO WELLS
C
      CALL INTIAL(PCF,PCM,CGMF,DZ,PB,PWOC,DWOC)
      CALL SUFUNC(IMAX,PSUCF,PSORF,SUCF,SORF,SPCZF,EPF,EMF,SUF,PCF,PCPF)
      CALL SUFUNC(NN,PSUCM,PSORM,SUCM,SORM,SPCZM,EPM,EMM,SUM,PCM,PCPM)
      DT = 0.5
      TYME = -0.5
      NIG = 20
      ITSN = -1
      IPFREQ = 1
      GO TO 350
C
C  READ TIME CARDS
C
 300  CONTINUE
      READ(5,310,ERR=800,END=999) TSTART,DT,TFINIS,NIG,IPFREQ,NUMTS
      WRITE(6,325) TSTART,DT,TFINIS,NIG,IPFREQ,NUMTS
      IF (NUMTS.GT.1) DUT=1.0/(FLOAT(NUMTS)-1)*ALOG((TFINIS-TSTART)/DT)
      DO 305 M = 1,NUELL
      READ(5,320,ERR=800) L,I,RW(I),SKIN(I),QUELL(I),PBH(I)
      WRITE(6,335) L,I,RW(I),SKIN(I),QUELL(I),PBH(I)
 305  CONTINUE
 310  FORMAT(10X,3F10.0,3I10)
 320  FORMAT(10X,2I10,4F10.0)
 325  FORMAT(///1X'TIME  ',4X,'TSTART=',F10.5,3X,'DT    =',F10.5,3X,
     $                    'TFINIS=',F10.5,3X,'NIG   =',I5,5X,
     $                    'IPFREQ =',I5,5X,'NUMTS  =',I5,/)
 335  FORMAT(1X'WELL  ',4X,'WELL #=',I10,3X,'NODE #=',I10,3X,
     $                    'RW    =',F10.5,3X,'SKIN  =',F10.5,3X,
     $                    'QUELL =',F10.5,3X,'PBH   =',F10.5,/)
C
C  BEGIN CALCULATIONS
C
 350  CONTINUE
      ITN = 0
      ITSN = ITSN + 1
 400  CONTINUE
      CALL RPERM(IMAX,EWF,EOF,SUCF,SORF,RKSUCF,RKSORF,SUF,RKWF,RKOF,
     $           RKWPF,RKOPF)
      CALL RPERM(NN,EWM,EOM,SUCM,SORM,RKSUCM,RKSORM,SUM,RKWM,RKOM,
     $           RKWPM,RKOPM)
      CALL CAPP(IMAX,PSUCF,PSORF,SUCF,SORF,SPCZF,EPF,EMF,SUF,PCF,PCPF)
      CALL CAPP(NN,PSUCM,PSORM,SUCM,SORM,SPCZM,EPM,EMM,SUM,PCM,PCPM)
$
```

```
10323000       DO 405 L=1,MN
10324000       SOM(L) = 1 - SUM(L)
10325000  405  POM(L) = PUM(L) + PCM(L)
10326000       DO 407 I=1,IMAX
10327000       SOF(I) = 1 - SUF(I)
10328000  407  POF(I) = PUF(I) + PCF(I)
10329000 C
10330000 C GET N-LEVEL PORE VOLUMES IF THIS IS FIRST ITERATION
10331000 C
10332000       IF (ITN.GT.0) GO TO 425
10333000       DO 410 L=1,MN
10334000       VPUMN(L) = VPUM(L)
10335000       VPOMN(L) = VPOM(L)
10336000  410  CONTINUE
10337000       DO 420 I=1,IMAX
10338000       VPUFN(I) = VPUF(I)
10339000       VPOFN(I) = VPOF(I)
10340000  420  CONTINUE
10341000  425  CALL FLUID(IMAX,PB,BUB,RHOUB,CU,BUF,RHOUF,PUF)
10342000       CALL FLUID(IMAX,PB,BOB,RHOOB,CO,BOF,RHOOF,POF)
10343000       CALL FLUID(MN,PB,BUB,RHOUB,CU,BUM,RHOUM,PUM)
10344000       CALL FLUID(MN,PB,BOB,RHOOB,CO,BOM,RHOOM,POM)
10345000       CALL VPORE(IMAX,1,VR,CF,PB,POF,PHIBF,PHIF,SOF,BOF,VPOF)
10346000       CALL VPORE(IMAX,1,VR,CF,PB,PUF,PHIBF,PHIF,SUF,BUF,VPUF)
10347000       CALL VPORE(IMAX,MMB,VR,CM,PB,PUM,PHIBM,PHIM,SUM,BUM,VPUM)
10348000       CALL VPORE(IMAX,MMB,VR,CM,PB,POM,PHIBM,PHIM,SOM,BOM,VPOM)
10349000 C
10350000 C  NEED TO RESET N-LEVEL PORE VOLUMES AT TIME ZERO
10351000 C
10352000       IF (ITN+TYME.GT.0) GO TO 445
10352100       WIPM = 0.0
10352200       WIPF = 0.0
10352300       OIPM = 0.0
10352400       OIPF = 0.0
10353000       DO 430 L=1,MN
10354000       VPUMN(L) = VPUM(L)
10355000       VPOMN(L) = VPOM(L)
10355100       WIPM = WIPM + VPUM(L)
10355200       OIPM = OIPM + VPOM(L)
10356000  430  CONTINUE
10357000       DO 440 I=1,IMAX
10358000       VPUFN(I) = VPUF(I)
10359000       VPOFN(I) = VPOF(I)
10359100       WIPF = WIPF + VPUF(I)
10359200       OIPF = OIPF + VPOF(I)
10360000  440  CONTINUE
10361000  445  CALL OMEGA(IMAX,PUF,RHOUF,DF,WUF)
10362000       CALL OMEGA(IMAX,POF,RHOOF,DF,WOF)
10363000       CALL OMEGA(MN,PUM,RHOUM,DM,WUM)
10364000       CALL OMEGA(MN,POM,RHOOM,DM,WOM)
10365000       CALL OMEGAI(IMAX,MMB,PUF,PUM,RHOUF,RHOUM,DF,DM,WUFF)
10366000       CALL OMEGAI(IMAX,MMB,POF,POM,RHOOF,RHOOM,DF,DM,WOFF)
10367000       CALL TRANS(0,IMAX,TCF,TUF,AKF,RKUF,WUF,VIW,BUF,DX,DY,DZ,RAD,RW1)
10368000       CALL TRANS(0,IMAX,TCF,TOF,AKF,RKOF,WOF,VIO,BOF,DX,DY,DZ,RAD,RW1)
10369000       CALL TRANS(MMB,IMAX,TCM,TUM,AKM,RKUM,WUM,VIW,BUM,DX,DY,DZ,RAD,RW1)
10370000       CALL TRANS(MMB,IMAX,TCM,TOM,AKM,RKOM,WOM,VIO,BOM,DX,DY,DZ,RAD,RW1)
10371000       CALL TRANSI(MMB,IMAX,TMF,TUMF,AKM,RKUM,RKUF,WUMF,VIW,BUF,BUM,
10372000      $            VR,SIGMA)
10373000       CALL TRANSI(MMB,IMAX,TMF,TOMF,AKM,RKOM,RKOF,WOMF,VIO,BOF,BOM,
10374000      $            VR,SIGMA)
10375000       CALL UTRANS(IMAX,DX,DY,DZ,RW,VIW,VIO,BUF,BOF,SKIN,RKUF,RKOF,
10376000      $                  AKF,GWELL,WW,PI,PIW,PIO,RWL,PBH,PWH,PWF,RAD(1))
10377000       CALL FFCOEF
10378000       CALL MMCOEF
10379000       CALL MFCOEF
10380000       CALL RESID(MMB,IMAX,RFW,RMW,RHOUF,RHOUM,PUF,PUM,DF,DM,TUF,TUM,
10381000      $           TUMF,UPUF,UPUFN,UPUM,UPUMN,PIW,PBH,PWH,DT)
10382000       CALL RESID(MMB,IMAX,RFO,RMO,RHOOF,RHOOM,POF,POM,DF,DM,TOF,TOM,
10383000      $           TOMF,VPOF,VPOFN,VPOM,VPOMN,PIO,PBH,PWH,DT)
10383010 C
10383020 C   THE FOLLOWING SUBROUTINES ARE COMMENTED OUT ON PURPOSE.
10383030 C   THE SUBROUTINES ARE USED ONLY FOR DEBUGGING.
10383040 C
10383100 C     CALL RDATA
10383500 C     CALL DEBUG
8
```

```
10384000       CALL PELIM(IMAX,NMB)
10385000 C     CALL DEBUG
10386000       CALL SETUP(A,D,IMAX)
10387000       CALL GBAND(A,D,X,2*IMAX,3,.001,IERR,0)
10388000       CALL UPDATE(X,PWF,SWF,PWM,SWM,QUELL,QW,QO,NMB,IMAX,CONVRG)
10389000       ITN = ITN + 1
10390000       IF (CONVRG) GO TO 500
10390500       CONVRG = .TRUE.
10391000       IF(ITN.LT.NIG) GO TO 400
10392000       WRITE(6,450) ITSN
10393000  450  FORMAT(//1X,'MAXIMUM ITERATIONS USED AT TIME STEP ',I5,//)
10394000  500  CONTINUE
10395000       TYME = TYME + DT
10396000       CALL RMTBAL(WMBT,WIPF,WPWF,WPWFN,QW,WIPM,WPWM,WPWMN,CUMW,CUMO,
10397000      $            OMBT,OIPF,OPOF,OPOFN,QO,OIPM,OPOM,OPOMN,NMB,IMAX,DT)
10397005       IF (MOD(ITSN,IPFREQ).NE.0) GO TO 700
10397010 C
10397020 C  WRITE TIME STEP INFORMATION
10397030 C
10397040       WRITE(6,600) ITSN,TYME,DT,ITN
10397050  600  FORMAT(1X,'TIME STEP # ',I3,14X,'TIME= ',F13.4,' DAYS',6X,'DELTA',
10397060     $' TIME= ',F13.4,' DAYS',10X,'NUMBER OF ITERATIONS= ',I3,///)
10397070       WRITE(6,610) WMBT, OMBT
10397080  610  FORMAT(40X,'WATER',18X,'OIL',/,1X,'MATERIAL BALANCE(FRACTION)',9X,
10397090     $E13.4,9X,E13.4,////)
10397100       IF (TYME.GT.0.0) WRITE(6,620)
10397110  620  FORMAT(44X,'W E L L   R E P O R T',//)
10397120       DO 650 I = 1,IMAX
10397125       IF (RW(I).EQ.0) GO TO 650
10397130       IF (QUELL(I).NE.0) WC = QW(I)/QUELL(I)
10397140       IF (CUMW(I).LE.1.0)
10397145      $WRITE(6,640) I,QO(I),QW(I),QUELL(I),PBH(I),WC,CUMW(I),CUMO(I)
10397150       IF (CUMW(I).GT.1.0)
10397160      $WRITE(6,630) I,QO(I),QW(I),QUELL(I),PBH(I),WC,CUMW(I),CUMO(I)
10397170  630  FORMAT(8X,'OIL RATE',6X,'WATER RATE',6X,'TOTAL RATE',5X,'PRESSURE'
10397180     $,9X,'WATER CUT',6X,'CUM. WATER',7X,'CUM. OIL',/,1X,'NODE',3X,
10397190     $'(STB/DAY)',6X,'(STB/DAY)',7X,'(STB/DAY)',7X,'(PSI)',10X,
10397200     $'(FRACTION)',9X,'(STB)',11X,'(STB)',/,1X,I3,4(2X,F13.4),3(3X,
10397210     $F13.4),///)
10397230  640  FORMAT(8X,'OIL RATE',6X,'WATER RATE',6X,'TOTAL RATE',5X,'PRESSURE'
10397240     $,9X,'WATER CUT',6X,'CUM. WATER',7X,'CUM. OIL',/,1X,'NODE',3X,
10397250     $'(STB/DAY)',6X,'(STB/DAY)',7X,'(STB/DAY)',7X,'(PSI)',10X,
10397260     $'(FRACTION)',9X,'(STB)',11X,'(STB)',/,1X,I3,4(2X,E13.6),3(3X,
10397270     $E13.6),///)
10397280  650  CONTINUE
10398000 C
10399000 C WRITE PRESSURES AND SATURATIONS
10400000 C
10401000       WRITE(6,507)
10402000  507  FORMAT(///40X,'SWM (FRACTION)',/,120('='))
10403000       CALL RITED(SWM,NMB,IMAX)
10404000       WRITE(6,517)
10405000  517  FORMAT(///40X,'SWF (FRACTION)',/,120('='))
10406000       CALL RITED(SWF,1,IMAX)
10407000       WRITE(6,527)
10408000  527  FORMAT(///40X,'PWM (PSI)',/,120('='))
10409000       CALL RITED(PWM,NMB,IMAX)
10410000       WRITE(6,537)
10411000  537  FORMAT(///40X,'PWF (PSI)',/,120('='))
10412000       CALL RITED(PWF,1,IMAX)
10413000       WRITE(6,547)
10414000  547  FORMAT(///40X,'POM (PSI)',/,120('='))
10415000       CALL RITED(POM,NMB,IMAX)
10416000       WRITE(6,557)
10417000  557  FORMAT(///40X,'POF (PSI)',/,120('='))
10418000       CALL RITED(POF,1,IMAX)
10418500  700  CONTINUE
10418600       IF (NUMTS.GT.1) DT = (TYME-TSTART)*(EXP(DUT)-1.0)
10419000       IF(TYME+0.0001.LT.TFINIS) GO TO 350
10420000       IF(TYME+0.0001.GE.TFINIS) GO TO 300
10421000       GO TO 999
10422000  800  WRITE(6,810)
10423000  810  FORMAT(2X,'FORMAT ERROR ON DATA CARD')
10424000  999  STOP
10424010       END
$
```

```
10424020 C
10424030 C*************************************************************************
10424040 C
10424050 C               RADGRD -- CALCULATES GRID FOR RADIAL SYSTEMS
10424060 C
10424070 C*************************************************************************
10424080 C
10424090       SUBROUTINE RADGRD(RW,RE,N,R,DX,DY,DZ,V)
10424100       DIMENSION R(1),DX(1),DY(1),DZ(1),V(1)
10424110 C
10424115       DU = ALOG(RE/RW) / N
10424120       DO 50 I=1,N
10424130       R(I) = RW * EXP(FLOAT(I)*DU)
10424140       IF (I.GT.1) GO TO 10
10424150       DX(I) = R(I) - RW
10424160       DY(I) = 3.1416*(R(I)**2-RW**2) / DX(I)
10424170       GO TO 20
10424180    10 DX(I) = R(I) - R(I-1)
10424190       DY(I) = 3.1416*(R(I)**2-R(I-1)**2) / DX(I)
10424200    20 V(I) = DX(I) * DY(I) * DZ(I) / 5.6146
10424210    50 CONTINUE
10424220       RETURN
10424230       END
$
```

```
10425005 C
10425010 C*************************************************************************
10425015 C
10425020 C                  INTIAL -- INTIALIZATION SUBROUTINE
10425025 C
10425030 C*************************************************************************
10425035 C
10425040       SUBROUTINE INTIAL(PCF,PCM,CGMF,DZ,PB,PWOC,DWOC)
10425045       DIMENSION PCF(1),PCM(1),CGMF(1),DZ(1)
10425050       COMMON /CONS/CW,CO,NUELL,DT,IMAX,NMB,VIO,VIW,RHOWB,RHOOB,CF,CM
10425055       COMMON /FRAC/BWF,BOF,DF,PWF,POF,PHIBF,PHIF,RKOF,RKWF,RKWPF,PCPF,
10425060      $             RKOPF,TCF,TOF,TWF,SWF,VR,UWF,UOF,RHOWF,RHOOF
10425065       COMMON /MATRIX/BWM,BOM,DM,PWM,POM,PHIBM,PHIM,RKOM,RKWM,RKWPM,
10425070      $              RKOPM,TCM,TOM,TWM,SWM,UWM,UOM,RHOWM,RHOOM,PCPM,
10425075      $              TMF,TOMF,TWMF,UWMF,UOMF
10425080       DIMENSION BWF(99),BOF(99),DF(99),PWF(99),POF(99),PCPF(99)
10425085       DIMENSION PHIBF(99),PHIF(99),RKOF(99),RKWF(99)
10425090       DIMENSION RKWPF(99),RKOPF(99),TCF(99),TOF(99),TWF(99)
10425095       DIMENSION RHOOF(99),SWF(99),VR(99),UWF(99),UOF(99),RHOWF(99)
10425100       DIMENSION BWM(99),BOM(99),DM(99),PWM(99),POM(99),RHOOM(99)
10425105       DIMENSION PHIBM(99),PHIM(99),RKOM(99),RKWM(99),PCPM(99)
10425110       DIMENSION RKWPM(99),RKOPM(99),TCM(99),TOM(99),TWM(99)
10425115       DIMENSION SWM(99),UWM(99),UOM(99),RHOWM(99)
10425118       LOGICAL CONVRG
10425120 C
10425122       NN = IMAX*NMB
10425125       DO  10 I=1,IMAX
10425130       POF(I) = PWOC
10425135       PWF(I) = PWOC
10425140       DO  10 J=1,NMB
10425145       L = J +(I-1)*NMB
10425150       PWM(L) = PWOC
10425155       DM(L) = DF(I) + CGMF(L)*DZ(I)
10425160       POM(L) = PWOC
10425165    10 CONTINUE
10425170    20 CONTINUE
10425175       CONVRG = .TRUE.
10425180       CALL FLUID(IMAX,PB,BWB,RHOWB,CW,BWF,RHOWF,PWF)
10425185       CALL FLUID(IMAX,PB,BOB,RHOOB,CO,BOF,RHOOF,POF)
10425190       CALL FLUID(NN,PB,BWB,RHOWB,CW,BWM,RHOWM,PWM)
10425195       CALL FLUID(NN,PB,BOB,RHOOB,CO,BOM,RHOOM,POM)
10425200       DO 30 I=1,IMAX
10425202       IF (I.EQ.IMAX) GO TO 25
10425205       IF (I.EQ.1) PWF(I) = PWOC + RHOWF(I)/144*(DF(I)-DWOC)
10425210       AVGD = (RHOWF(I+1)+RHOWF(I)) / 288
10425215       DP = PWF(I+1) - PWF(I) - AVGD*(DF(I+1)-DF(I))
10425220       IF (ABS(DP).GT.0.0001) CONVRG = .FALSE.
10425225       PWF(I+1) = PWF(I+1) - DP
10425227       IF (I.GT.1) GO TO 22
10425230       POF(I) = PWOC + RHOOF(I)/144*(DF(I)-DWOC)
10425232       PCF(I) = POF(I) - PWF(I)
10425235    22 AVGD = (RHOOF(I+1)+RHOOF(I)) / 288
10425240       DP = POF(I+1) - POF(I) - AVGD*(DF(I+1)-DF(I))
10425245       IF (ABS(DP).GT.0.0001) CONVRG = .FALSE.
10425250       POF(I+1) = POF(I+1) - DP
10425255       PCF(I+1) = POF(I+1) - PWF(I+1)
10425260    25 DO 30 J=1,NMB
10425265       L = J + (I-1)*NMB
10425270       AVGD = (RHOWM(L)+RHOWF(I)) / 288
10425275       DP = PWM(L) - PWF(I) - AVGD*(DM(L)-DF(I))
10425280       IF (ABS(DP).GT.0.0001) CONVRG = .FALSE.
10425285       PWM(L) = PWM(L) - DP
10425290       AVGD = (RHOOM(L)+RHOOF(I)) / 288
10425300       DP = POM(L) - POF(I) - AVGD*(DM(L)-DF(I))
10425305       IF (ABS(DP).GT.0.0001) CONVRG = .FALSE.
10425310       POM(L) = POM(L) - DP
10425315       PCM(L) = POM(L) - PWM(L)
10425320    30 CONTINUE
10425325       IF (CONVRG) RETURN
10425330       NITN = NITN + 1
10425335       IF (NITN.LT.20) GO TO 20
10425340       RETURN
10425345       END
$
```

```
C
C***********************************************************************
C                                                                      *
C          SUFUNC -- CAPILLARY PRESSURE CALCULATION                    *
C                                                                      *
C***********************************************************************
C
C        THIS SUBROUTINE CALCULATES WATER SATURATION FOR EACH
C         NODE AS A FUNCTION OF CAPILLARY PRESSURE. POWER-FIT
C         EQUATIONS ARE USED TO DESCRIBE THE CAPILLARY PRESSURE
C         FUNCTION.  CAPILLARY PRESSURE CURVES ARE ASSUMED TO BE
C         IMBIBITION CURVES AND EXTEND FROM A MAXIMUM VALUE AT
C         CONNATE WATER TO A MINIMUM VALUE (ZERO OR NEGATIVE) AT
C         RESIDUAL OIL. SEPARATE POWER-FIT EQUATIONS ARE USED
C         ABOVE AND BELOW THE ZERO CAPILLARY PRESSURE POINT.
C
C        PARAMETERS:
C
C          N -- THE NUMBER OF NODES.
C          PSWC -- CAPILLARY PRESSURE AT CONNATE WATER. MUST BE ZERO
C                  OR POSITIVE.
C          PSOR -- CAPILLARY PRESSURE AT RESIDUAL OIL. MUST BE ZERO
C                  OR NEGATIVE.
C          SWC -- CONNATE WATER SATURATION.
C          SOR -- RESIDUAL OIL SATURATION.
C          SPCZ -- SATURATION AT THE ZERO CAPILLARY PRESSURE POINT.
C          EP -- EXPONENT FOR THE POSITIVE PORTION OF THE CAPILLARY
C                PRESSURE CURVE.
C          EM -- EXPONENT FOR THE NEGATIVE PORTION OF THE CAPILLARY
C                PRESSURE CURVE.
C          SW -- WATER SATURATION VALUE.
C          PC -- CAPILLARY PRESSURE VALUE.
C          PCP -- DERIVATIVE OF CAPILLARY PRESSURE.
C
C***********************************************************************
C
      SUBROUTINE SWFUNC(N,PSWC,PSOR,SWC,SOR,SPCZ,EP,EM,SW,PC,PCP)
      DIMENSION PC(1),SW(1),PCP(1)
C
      DP = SPCZ - SWC
      IF (DP.EQ.0.0) DP = 9999.0
      DM = 1-SOR-SPCZ
      IF (DM.EQ.0.0) DM = 9999.0
      DO 30 I=1,N
      IF (PC(I).LT.PSWC) GO TO 20
C   WATER IS AT CONNATE SATURATION.
      PC(I) = PSWC
      SW(I) = SWC
      GO TO 30
   20 IF (PC(I).GT.PSOR) GO TO 21
C   WATER SATURATION IS 1-SOR (RESIDUAL OIL SATURATION)
      PC(I) = PSOR
      SW(I) = 1 - SOR
      GO TO 30
   21 IF (PC(I).LT.0) GO TO 22
C   WATER SATURATION IS BETWEEN CONNATE SATURATION AND THE ZERO
C     CAPILLARY PRESSURE SATURATION
      SW(I) = SPCZ - (PC(I)/PSWC)**(1/EP)*(DP)
      GO TO 30
C   WATER SATURATION IS BETWEEN THE ZERO CAPILLARY SATURATION AND
C     1 - SOR (RESIDUAL OIL SATURATION)
   22 SW(I) = SPCZ + (PC(I)/PSOR)**(1/EM)*(DM)
   30 CONTINUE
      RETURN
      END
$
```

```
10426000 C
10427000 C*************************************************************************
10428000 C                                                                        *
10429000 C              RPERM -- RELATIVE PERMEABILITY CALCULATION                *
10430000 C                                                                        *
10431000 C*************************************************************************
10432000 C
10433000 C        CALCULATION OF RELATIVE PERMEABILITIES AND DERIVATIVES
10434000 C         OF RELATIVE PERMEABILITY FOR EACH NODE
10435000 C         USING POWER-FIT EQUATION PARAMETERS AND THE WATER
10436000 C         SATURATION FOR EACH NODE.
10437000 C
10438000 C        PARAMETERS:
10439000 C
10440000 C         N -- NUMBER OF NODES
10441000 C         EW -- EXPONENT FOR WATER RELATIVE PERMEABILITY EQUATION
10442000 C         EO -- EXPONENT FOR OIL RELATIVE PERMEABILITY EQUATION
10443000 C         SWC -- CONNATE WATER SATURATION
10444000 C         SOR -- RESIDUAL OIL SATURATION
10445000 C         RKSWC -- OIL RELATIVE PERMEABILITY AT CONNATE WATER
10446000 C         RKSOR -- WATER RELATIVE PERMEABILITY AT RESIDUAL OIL
10447000 C         SW -- WATER SATURATION
10448000 C         RKW -- WATER RELATIVE PERMEABILITY
10449000 C         RKO -- OIL RELATIVE PERMEABILITY
10450000 C         RKWP -- DERIVATIVE OF WATER RELATIVE PERMEABILITY
10451000 C         RKOP -- DERIVATIVE OF OIL RELATIVE PERMEABILITY
10452000 C
10453000 C*************************************************************************
10454000 C
10455000       SUBROUTINE RPERM(N,EW,EO,SWC,SOR,RKSWC,RKSOR,SW,RKW,RKO,RKWP,RKOP)
10456000       DIMENSION SW(1),RKW(1),RKO(1),RKWP(1),RKOP(1)
10457000
10458000       DENOM = 1 - SOR - SWC
10459000       DO 30 I = 1,N
10460000       IF (SW(I).GT.SWC) GO TO 20
10461000 C   WATER IS AT CONNATE SATURATION.
10463000       RKW(I) = 0
10464000       RKO(I) = RKSWC
10465000       RKWP(I) = 0
10466000       RKOP(I) = -EO*RKSWC/DENOM
10467000       GO TO 30
10468000    20 IF (SW(I).LT.1-SOR) GO TO 21
10469000 C   WATER SATURATION IS 1-SOR (RESIDUAL OIL SATURATION)
10471000       RKW(I) = RKSOR
10472000       RKO(I) = 0
10473000       RKWP(I) = EW*RKSOR/DENOM
10474000       RKOP(I) = 0
10475000       GO TO 30
10476000    21 RKW(I) = RKSOR*((SW(I)-SWC)/DENOM)**EW
10477000       RKO(I) = RKSWC*((1-SOR-SW(I))/DENOM)**EO
10478000       RKWP(I) = EW*RKSOR/DENOM*((SW(I)-SWC)/DENOM)**(EW-1)
10479000       RKOP(I) = -EO*RKSWC/DENOM*((1-SOR-SW(I))/DENOM)**(EO-1)
10480000    30 CONTINUE
10481000       RETURN
10482000       END
$
```

```
10483000 C
10484000 C*************************************************************************
10485000 C                                                                        *
10486000 C            CAPP -- CAPILLARY PRESSURE CALCULATION                      *
10487000 C                                                                        *
10488000 C*************************************************************************
10489000 C
10490000 C       THIS SUBROUTINE CALCULATES CAPILLARY PRESSURE FOR EACH
10491000 C        NODE AS A FUNCTION OF WATER SATURATION. POWER-FIT
10492000 C        EQUATIONS ARE USED TO DESCRIBE THE CAPILLARY PRESSURE
10493000 C        FUNCTION.  CAPILLARY PRESSURE CURVES ARE ASSUMED TO BE
10494000 C        IMBIBITION CURVES AND EXTEND FROM A MAXIMUM VALUE AT
10495000 C        CONNATE WATER TO A MINIMUM VALUE (ZERO OR NEGATIVE) AT
10496000 C        RESIDUAL OIL.  SEPARATE POWER-FIT EQUATIONS ARE USED
10497000 C        ABOVE AND BELOW THE ZERO CAPILLARY PRESSURE POINT.
10498000 C
10499000 C       PARAMETERS:
10500000 C
10501000 C         N -- THE NUMBER OF NODES.
10502000 C         PSUC -- CAPILLARY PRESSURE AT CONNATE WATER. MUST BE ZERO
10503000 C                 OR POSITIVE.
10504000 C         PSOR -- CAPILLARY PRESSURE AT RESIDUAL OIL. MUST BE ZERO
10505000 C                 OR NEGATIVE.
10506000 C         SUC -- CONNATE WATER SATURATION.
10507000 C         SOR -- RESIDUAL OIL SATURATION.
10508000 C         SPCZ -- SATURATION AT THE ZERO CAPILLARY PRESSURE POINT.
10509000 C         EP -- EXPONENT FOR THE POSITIVE PORTION OF THE CAPILLARY
10510000 C               PRESSURE CURVE.
10511000 C         EM -- EXPONENT FOR THE NEGATIVE PORTION OF THE CAPILLARY
10512000 C               PRESSURE CURVE.
10513000 C         SW -- WATER SATURATION VALUE.
10514000 C         PC -- CAPILLARY PRESSURE VALUE.
10515000 C         PCP -- DERIVATIVE OF CAPILLARY PRESSURE.
10516000 C
10517000 C*************************************************************************
10518000 C
10519000       SUBROUTINE CAPP(N,PSUC,PSOR,SUC,SOR,SPCZ,EP,EM,SW,PC,PCP)
10520000       DIMENSION PC(1),SW(1),PCP(1)
10521000 C
10521100       DP = SPCZ - SUC
10521200       IF (DP.EQ.0.0) DP = 9999.0
10521300       DM = 1-SOR-SPCZ
10521400       IF (DM.EQ.0.0) DM = 9999.0
10522000       DO 30 I=1,N
10523000       IF (SW(I).GT.SUC) GO TO 20
10524000 C   WATER IS AT CONNATE SATURATION.
10525000       PC(I) = PSUC
10526000       PCP(I) = -EP*PSUC/(DP)
10527000       GO TO 30
10528000    20 IF (SW(I).LT.1-SOR) GO TO 21
10529000 C   WATER SATURATION IS 1-SOR (RESIDUAL OIL SATURATION)
10530000       PC(I) = PSOR
10531000       PCP(I) = EM*PSOR/(DM)
10532000       GO TO 30
10533000    21 IF (SW(I).GT.SPCZ) GO TO 22
10534000 C   WATER SATURATION IS BETWEEN CONNATE SATURATION AND THE ZERO
10535000 C      CAPILLARY PRESSURE SATURATION
10536000       PC(I) = PSUC*((SPCZ-SW(I))/(DP))**EP
10537000       PCP(I) = -EP*PSUC/(DP)*((SPCZ-SW(I))/(DP))**(EP-1)
10538000       GO TO 30
10539000 C   WATER SATURATION IS BETWEEN THE ZERO CAPILLARY SATURATION AND
10540000 C      1 - SOR (RESIDUAL OIL SATURATION)
10541000    22 PC(I) = PSOR*((SW(I)-SPCZ)/(DM))**EM
10542000       PCP(I) = EM*PSOR/(DM)*((SW(I)-SPCZ)/(DM))**(EM-1)
10543000    30 CONTINUE
10544000       RETURN
10545000       END
$
```

```
C
C***********************************************************************
C                                                                      *
C                  FLUID -- CALUCLATION OF FLUID PROPERTIES            *
C                                                                      *
C***********************************************************************
C
C         THIS SUBROUTINE CALCULATES DENSITIES AND FORMATION
C          VOLUME FACTORS AS FUNCTIONS OF PRESSURE.  FLUID
C          COMPRESSIBILITY IS ASSUMED CONSTANT AND THERFORE
C          THE FLUID PROPERTIES VARY LINEARLY WITH PRESSURE.
C
C         PARAMETERS:
C
C          N -- NUMBER OF NODES
C          PB -- BASE PRESSURE VALUE
C          BB -- FORMATION VOLUME FACTOR AT BASE PRESSURE
C          RHOB -- DENSITY AT BASE PRESSURE
C          CF -- FLUID COMPRESSIBILITY
C          BF -- FORMATION VOLUME FACTOR
C          RHO -- DENSITY
C          P -- FLUID PRESSURE
C
C***********************************************************************
C
      SUBROUTINE FLUID(N,PB,BB,RHOB,CF,BF,RHO,P)
      DIMENSION BF(1),RHO(1),P(1)
C
      DO 10 I=1,N
      BF(I) = BB*(1 - CF*(P(I)-PB))
      RHO(I)= RHOB*(1 + CF*(P(I)-PB))
   10 CONTINUE
      RETURN
      END
$
```

```
C
C*********************************************************************
C                                                                    *
C          UPORE  -- PORE VOLUME CALCULATION                         *
C                                                                    *
C*********************************************************************
C
C        THIS SUBROUTINE CALCULATES POROSITY AS A FUNCTION OF FLUID
C         PRESSURE. ROCK COMPRESSIBILITY IS ASSUMED TO BE CONSTANT
C         AND THEREFORE POROSITY INCREASES LINEARLY WITH PRESSURE.
C         PORE VOLUMES ARE CALCULATED BY MULTIPLYING NODE VOLUME
C         BY POROSITY AND SATURATION.
C
C        PARAMETERS:
C
C         N -- NUMBER OF NODES
C         M -- NUMBER OF MATRIX BLOCKS
C         C -- ROCK COMPRESSIBILITY
C         VR -- NODE VOLUME
C         PB -- BASE FLUID PRESSURE
C         P -- FLUID PRESSURE
C         PHIB -- POROSITY AT BASE FLUID PRESSURE
C         PHI -- POROSITY
C         S -- PHASE SATURATION OF NODE
C         B -- FORMATION VOLUME FACTOR
C         VP -- PORE VOLUME OF NODE
C
C*********************************************************************
C
      SUBROUTINE UPORE(N,M,VR,C,PB,P,PHIB,PHI,S,B,VP)
      DIMENSION VR(1),P(1),PHI(1),VP(1),PHIB(1),S(1),B(1)
C
      DO 10 I=1,N
      DO 10 J=1,M
      L = J + (I-1)*M
      V = VR(I) / FLOAT(M)
      PHI(L) = PHIB(L)*(1 + C*(P(L)-PB))
      VP(L) = PHI(L) * V * S(L) / B(L)
   10 CONTINUE
      RETURN
      END
$
```

```
C
C***********************************************************************
C                                                                      *
C             OMEGA -- CALCULATE UPSTREAM WEIGHTING FACTORS            *
C                                                                      *
C***********************************************************************
C
C            THIS SUBROUTINE DETERMINES WHICH NODE (I OR I-1) IS
C             UPSTREAM. THE NODE WITH THE LARGEST FLUID POTENTIAL
C             IS CONSIDERED THE UPSTREAM NODE.  A 'U' VALUE OF 1
C             FOR NODE 'I' MEANS FLUID WILL FLOW INTO THIS NODE
C             FROM NODE 'I-1'. A '0' VALUE MEANS FLUID WILL FLOW
C             OUT.  RELATIVE PERMEABILITY BETWEEN TWO NODES IS
C             DETERMINED BY THE UPSTREAM NODE SATURATION.
C
C            PARAMETERS:
C             N -- NUMBER OF NODES
C             P -- FLUID PRESSURE
C             RHO -- FLUID DENSITY
C             D -- DEPTH OF NODE (POSITIVE DOWN)
C             U -- WEIGHTING FACTOR. A VALUE OF 1 MEANS FLUID FLOWS
C                  IN FROM NODE I-1. A VALUE OF 0 MEANS FLUID FLOWS
C                  INTO NODE I-1.
C
C***********************************************************************
C
      SUBROUTINE OMEGA(N,P,RHO,D,U)
      DIMENSION P(1),RHO(1),U(1),D(1)
C
      IF (N.EQ.1) RETURN
      DO 10 I=2,N
      AVGGRD = (RHO(I) + RHO(I-1))/(2*144)
      PTI = P(I) - AVGGRD * D(I)
      PTIM = P(I-1) - AVGGRD*D(I-1)
      T = PTIM - PTI
      U(I) = SIGN(0.5,T) + 0.5
   10 CONTINUE
      RETURN
      END
```

```
C
C************************************************************************
C                                                                       *
C           OMEGAI -- CALCULATE UPSTREAM WEIGHTING FACTORS             *
C                                                                       *
C************************************************************************
C
C           THIS SUBROUTINE DETERMINES WHICH POROSITY (F OR M) IS
C            UPSTREAM. THE POROSITY WITH THE LARGEST FLUID POTENTIAL
C            IS CONSIDERED THE UPSTREAM NODE.  A 'W' VALUE OF 1
C            FOR POROSITY 'F' MEANS FLUID WILL FLOW IN
C            FROM POROSITY 'M'. A '0' VALUE MEANS FLUID WILL FLOW
C            OUT.  RELATIVE PERMEABILITY BETWEEN TWO POROSITIES IS
C            DETERMINED BY THE UPSTREAM SATURATION.
C
C           PARAMETERS:
C            N -- NUMBER OF NODES
C            M -- NUMBER OF MATRIX BLOCKS
C            PF -- PRESSURE OF FRACTURE
C            PM -- PRESSURE OF MATRIX
C            RHOF -- FRACTURE FLUID DENSITY
C            RHOM -- MATRIX FLUID DENSITY
C            DF -- DEPTH OF FRACTURE (POSITIVE DOWN)
C            DM -- DEPTH OF MATRIX (POSITIVE DOWN)
C            W -- WEIGHTING FACTOR. A VALUE OF 1 MEANS FLUID FLOWS
C                 INTO FRACTURE. A VALUE OF 0 MEANS FLUID FLOWS
C                 INTO MATRIX.
C
C************************************************************************
C
      SUBROUTINE OMEGAI(N,M,PF,PM,RHOF,RHOM,DF,DM,W)
      DIMENSION PF(1),PM(1),RHOF(1),RHOM(1),DF(1),DM(1),W(1)
C
      DO 10 I=1,N
      DO 10 J=1,M
      L = J + (I-1)*M
      AVGGRD = (RHOF(I) + RHOM(L))/(2*144)
      PTF = PF(I) - AVGGRD*DF(I)
      PTM = PM(L) - AVGGRD*DM(L)
      T = PTM - PTF
      W(L) = SIGN(0.5,T) + 0.5
   10 CONTINUE
      RETURN
      END
$
```

```
$
10704000 C
10705000 C*************************************************************************
10706000 C                                                                        *
10707000 C                    TRANS -- TRANSMISSIBILITY CALCULATION               *
10708000 C                                                                        *
10709000 C*************************************************************************
10710000 C
10711000 C             THIS SUBROUTINE CALCULATES TRANSMISSIBLITIES BETWEEN
10712000 C              NODE 'I' AND NODE 'I-1'. THE TRANSMIBILITY VALUE
10713000 C              BETWEEN THE TWO NODES IS STORED IN THE 'I' NODE
10714000 C              LOCATION. ARITHMETIC AVERAGE AREA AND HARMONIC
10715000 C              AVERAGE PERMEABILITY ARE USED FOR THE TRANSMISSIBITY
10716000 C              CALCULATION. UPSTREAM FLUID PROPERTIES ARE USED.
10717000 C
10718000 C             PARAMETERS:
10719000 C
10720000 C              M -- NUMBER OF CONNECTED MATRIX BLOCKS
10721000 C              N -- NUMBER OF NODES
10722000 C              TC -- CONSTANT PORTION OF TRANSMISSIBILITY
10723000 C              T -- TRANSMISSIBILTY.
10724000 C              AK -- ABSOLUTE PERMEABILITY
10725000 C              RK -- RELATIVE PERMEABILITY
10726000 C              W -- UPSTREAM WEIGHTING FACTOR
10727000 C              VIS -- VISCOSITY
10728000 C              B -- FORMATION VOLUME FACTOR
10729000 C              DX -- NODE SIZE IN X DIRECTION
10730000 C              DY -- NODE SIZE IN Y DIRECTION
10731000 C              DZ -- NODE SIZE IN Z DIRECTION
10732000 C
10733000 C*************************************************************************
10734000 C
10735000       SUBROUTINE TRANS(M,N,TC,T,AK,RK,W,VIS,B,DX,DY,DZ,RAD,RW)
10736000       DIMENSION W(1),TC(1),T(1),AK(1),RK(1),B(1),DX(1),DY(1),DZ(1),
10736100      $          RAD(1)
10737000 C
10738000       I1 = 2
10739000       IF (M.EQ.1) RETURN
10740000       IF (M.GT.0) I1 = 1
10741000       DO 30 I=I1,M
10742000       DO 30 J=2,N
10743000       L = J + (I-1)*M
10744000       IF (M.EQ.0) GO TO 20
10745000       AREA = DX(I)*DY(I)
10746000       DL = DZ(I) / FLOAT(M)
10747000       HK = 2*AK(L)*AK(L-1) / (AK(L) + AK(L-1))
10748000       GO TO 25
10748100   20  IF (RAD(I).GT.0) GO TO 22
10748200       AREA = (DZ(I)*DY(I) + DZ(I-1)*DY(I-1))
10750000       DL = (DX(I) + DX(I-1))
10750050       HK = DL * AK(I) * AK(I-1) / (DX(I-1)*AK(I) + DX(I)*AK(I-1))
10750100       GO TO 25
10750200   22  AREA = 3.1416 * RAD(I-1) * (DZ(I)+DZ(I-1))
10750250       IF (I.GT.2) GO TO 23
10750300       DL = SQRT(RAD(I)*RAD(I-1)) - SQRT(RAD(I-1)*RW)
10750350       HK = DL*AK(I)*AK(I-1)/((SQRT(RAD(I)*RAD(I-1))-RAD(I-1))*AK(I-1)
10750360      $     + (RAD(I-1)-SQRT(RAD(I-1)*RW))*AK(I))
10750370       GO TO 25
10750400   23  DL = SQRT(RAD(I)*RAD(I-1)) - SQRT(RAD(I-1)*RAD(I-2))
10751000       HK = DL*AK(I)*AK(I-1)/((SQRT(RAD(I)*RAD(I-1))-RAD(I-1))*AK(I-1)
10751100      $     + (RAD(I-1)-SQRT(RAD(I-1)*RAD(I-2)))*AK(I))
10752000   25  II = I
10753000       IF (M.GT.0) II = L
10754000       RMOB = (W(II)*RK(II-1)/B(II-1) + (1-W(II))*RK(II)/B(II))/VIS
10755000       TC(II) = 0.001127*HK*AREA/DL
10756000       T(II) = TC(II) * RMOB
10757000   30  CONTINUE
10758000       RETURN
10759000       END
$
```

```
C
C**********************************************************************
C                                                                     *
C                    TRANSI -- TRANSMISSIBILITY CALCULATION           *
C                                                                     *
C**********************************************************************
C
C                 THIS SUBROUTINE CALCULATES TRANSMISSIBLITIES BETWEEN
C                 FRACTURE AND MATRIX. UPSTREAM FLUID PROPERTIES ARE USED.
C
C                 PARAMETERS:
C
C                  M -- NUMBER OF CONNECTED MATRIX BLOCKS
C                  N -- NUMBER OF NODES
C                  TC -- CONSTANT PORTION OF TRANSMISSIBILITY
C                  T -- TRANSMISSIBILTY.
C                  AK -- ABSOLUTE PERMEABILITY OF MATRIX
C                  RKF -- RELATIVE PERMEABILITY OF FRAC TURE
C                  RKM -- RELATIVE PERMEABILITY OF MATRIX
C                  W -- UPSTREAM WEIGHTING FACTOR
C                  VIS -- FLUID VISCOSITY
C                  BF -- FRACTURE FORMATION VOLUME FACTOR
C                  BM -- MATRIX FORMATION VOLUME FACTOR
C                  V -- NODE VOLUME
C                  SIGMA -- SPACING PARAMETER
C
C**********************************************************************
C
      SUBROUTINE TRANSI(M,N,TC,T,AK,RKM,RKF,W,VIS,BF,BM,V,SIGMA)
      DIMENSION TC(1),T(1),AK(1),RKF(1),RKM(1),W(1),BF(1),BM(1),
     $          V(1),SIGMA(1)
C
      DO 10 I=1,N
      DO 10 J=1,M
      L = J + (I-1)*M
      RMOB = ((1-W(L))*RKF(I)/BF(I) + W(L)*RKM(L)/BM(L)) / VIS
      TC(L) = 0.006328*V(I)*AK(L)*SIGMA(L)/ FLOAT(M)
      T(L) = TC(L)*RMOB
   10 CONTINUE
      RETURN
      END
$
```

```
10801000 C
10802000 C*************************************************************************
10803000 C              WTRANS -- WELLBORE TRANSMISSIBILITY CALCULATION
10804000 C
10805000 C*************************************************************************
10806000 C
10807000 C             THIS SUBROUTINE CALCULATES PRODUCTIVITY INDICES FOR
10808000 C             OIL AND WATER AT THE WELLBORE. FOR INJECTION RELATIVE
10809000 C             PERMEABILITY FOR INJECTION FLUID(WATER) IS ASSUMED TO
10810000 C             BE EQUAL TO THE SUM OF WATER AND OIL RELATIVE PERM. OF
10811000 C             THE NODE.  WELLBORE RESIDUAL IS ALSO CALCULATED.
10812000 C
10813000 C             PARAMETERS:
10814000 C
10815000 C              N -- NUMBER OF NODES
10816000 C              DX -- GRID BLOCK SIZE IN X-DIRECTION, FT.
10817000 C              DY -- GRID BLOCK SIZE IN Y-DIRECTION, FT.
10818000 C              DZ -- GRID BLOCK SIZE IN Z-DIRECTION, FT.
10819000 C              RW -- WELLBORE RADIUS, FT.
10820000 C              VIW -- WATER VISCOSITY, CP.
10821000 C              VIO -- OIL VISCOSITY, CP.
10822000 C              BW -- WATER FORMATION VOLUME FACTOR, RES.BBL./STB
10823000 C              BO -- OIL FORMATION VOLUME FACTOR, RES. BBL./STB
10824000 C              S -- SKIN FACTOR, DIMENSIONLESS
10825000 C              RKW -- WATER RELATIVE PERMEABILITY
10826000 C              RKO -- OIL RELATIVE PERMEABILITY
10827000 C              AK -- ABSOLUTE PERMEABILITY, MD.
10828000 C              QWELL -- WELLBORE RATE, STB/D.
10829000 C              WW -- WELLBORE WEIGHTING FACTOR (1 FOR INJECTION)
10830000 C              PI -- WELLBORE PRODUCTIVITY CONSTANT
10831000 C              PIW -- WATER WELLBORE PRODUCTIVITY (TRANSMISSIBILITY)
10832000 C              PIO -- OIL WELLBORE PRODUCTIVITY (TRANSMISSIBILITY)
10833000 C              RWL -- WELLBORE RESIDUAL
10834000 C              PBH -- WELLBORE PRESSURE
10835000 C              PUN -- WELL NODE PRESSURE
10836000 C
10837000 C*************************************************************************
10838000 C
10839000       SUBROUTINE WTRANS(N,DX,DY,DZ,RW,VIW,VIO,BW,BO,S,RKW,RKO,
10840000      $                  AK,QWELL,WW,PI,PIW,PIO,RWL,PBH,PUN,PUF,R1)
10841000       DIMENSION DX(1),DY(1),DZ(1),RW(1),BW(1),BO(1),S(1),RKW(1),
10842000      $           RKO(1),AK(1),QWELL(1),WW(1),PI(1),PIW(1),PIO(1),RWL(1),
10843000      $           PBH(1),PUN(1),PUF(1)
10844000
10845000       DO 10 I=1,N
10846000       IF (R1.EQ.0) R1 = SQRT(DX(I)*DY(I)/3.6146)
10847000       IF (RW(I).EQ.0) GO TO 10
10848000       PI(I) = 0.00708*DZ(I)*AK(I)/(R1*R1/(R1*R1-RW(I)*RW(I))*
10849000      $        ALOG(R1/RW(I)) - 0.5 + S(I))
10850000       WW(I) = SIGN(0.5,QWELL(I)) + 0.5
10851000       WMOB = (RKW(I) + WW(I)*RKO(I))/ BW(I) / VIW
10852000       OMOB = RKO(I) / BO(I) / VIO * (1-WW(I))
10853000       PIW(I) = PI(I)*WMOB
10854000       PIO(I) = PI(I)*OMOB
10855000       IF (PUN(I).EQ.0) PUN(I) = PUF(I)
10856000       IF (PBH(I).EQ.0) PBH(I) = QWELL(I)/(PIO(I)+PIW(I)) + PUN(I)
10857000       RWL(I) = QWELL(I)-(PIO(I)+PIW(I))*(PBH(I)-PUN(I))
10858000    10 CONTINUE
10859000       RETURN
10860000       END
@
```

```
10862000 C*************************************************************************
10863000 C                                                                        *
10864000 C            FFCOEF  -- CALCULATES TERMS FOR JACOBIAN MATRIX             *
10865000 C                                                                        *
10866000 C*************************************************************************
10867000 C
10868000 C      THIS SUBROUTINE CALCULATES PARTIAL DERIVATIVES OF FRACTURE
10869000 C        RESIDUALS WITH RESPECT TO FRACTURE VARIABLES FOR USE IN THE
10870000 C        JACOBIAN MATRIX.  WELLBORE DERIVATIVES ARE ALSO CALCULATED.
10871000 C         THE COEFFICIENTS CALCULATED ARE AS FOLLOWS:
10872000 C
10873000 C           A11 -- PARTIAL DERIVATIVE OF FRACTURE WATER RESIDUAL WITH
10874000 C                  RESPECT TO FRACTURE WATER PRESSURE OF NODE I-1.
10875000 C           A12 -- PARTIAL DERIVATIVE OF FRACTURE WATER RESIDUAL WITH
10876000 C                  RESPECT TO FRACTURE WATER SATURATION OF NODE I-1.
10877000 C           A21 -- PARTIAL DERIVATIVE OF FRACTURE OIL RESIDUAL WITH
10878000 C                  RESPECT TO FRACTURE WATER PRESSURE OF NODE I-1.
10879000 C           A22 -- PARTIAL DERIVATIVE OF FRACTURE OIL RESIDUAL WITH
10880000 C                  RESPECT TO FRACTURE WATER SATURATION OF NODE I-1.
10881000 C           B11 -- PARTIAL DERIVATIVE OF FRACTURE WATER RESIDUAL WITH
10882000 C                  RESPECT TO FRACTURE WATER PRESSURE OF NODE I.
10883000 C           B12 -- PARTIAL DERIVATIVE OF FRACTURE WATER RESIDUAL WITH
10884000 C                  RESPECT TO FRACTURE WATER SATURATION OF NODE I.
10885000 C           B21 -- PARTIAL DERIVATIVE OF FRACTURE OIL RESIDUAL WITH
10886000 C                  RESPECT TO FRACTURE WATER PRESSURE OF NODE I.
10887000 C           B22 -- PARTIAL DERIVATIVE OF FRACTURE OIL RESIDUAL WITH
10888000 C                  RESPECT TO FRACTURE WATER SATURATION OF NODE I.
10889000 C           C11 -- PARTIAL DERIVATIVE OF FRACTURE WATER RESIDUAL WITH
10890000 C                  RESPECT TO FRACTURE WATER PRESSURE OF NODE I+1.
10891000 C           C12 -- PARTIAL DERIVATIVE OF FRACTURE WATER RESIDUAL WITH
10892000 C                  RESPECT TO FRACTURE WATER SATURATION OF NODE I+1.
10893000 C           C21 -- PARTIAL DERIVATIVE OF FRACTURE OIL RESIDUAL WITH
10894000 C                  RESPECT TO FRACTURE WATER PRESSURE OF NODE I+1.
10895000 C           C22 -- PARTIAL DERIVATIVE OF FRACTURE OIL RESIDUAL WITH
10896000 C                  RESPECT TO FRACTURE WATER SATURATION OF NODE I+1.
10897000 C           W1 -- PARTIAL DERIVATIVE OF WELLBORE RESIDUAL WITH RESPECT
10898000 C                 TO FRACTURE WATER PRESSURE OF NODE I.
10899000 C           W2 -- PARTIAL DERIVATIVE OF WELLBORE RESIDUAL WITH RESPECT
10900000 C                 TO FRACTURE WATER SATRUATION OF NODE I.
10901000 C           WD -- PARTIAL DERIVATIVE OF WELLBORE RESIDUAL WITH RESPECT
10902000 C                 TO WELLBORE PRESSURE.
10903000 C
10904000 C           SEE THE MAIN PROGRAM FOR DEFINITION OF OTHER VARIABLES.
10905000 C
10906000 C*************************************************************************
10907000 C
10908000       SUBROUTINE FFCOEF
10909000       COMMON /CONS/CW,CO,GRAV,DT,IMAX,NMB,VIO,VIW,RHOWB,RHOOB,CF,CR
10910000       COMMON /FRAC/BWF,BOF,DF,PWF,POF,PHIBF,PHIF,RKOF,RKWF,RKWPF,PCPF,
10911000      $             RKOPF,TCF,TOF,TWF,SWF,WR,WWF,WOF,RHOWF,RHOOF
10912000       COMMON /COEFF/A11,A12,A21,A22,B11,B12,B21,B22,C11,C12,C21,C22
10913000       COMMON /WELL/ PI,PBH,PIW,PIO,WW,W1,W2,WD,PWH
10914000       COMMON /MATRIX/BWM,BOM,DM,PWM,POM,PHIBM,PHIM,RKOM,RKWM,RKWPM,
10915000      $              RKOPM,TCM,TOM,TWM,SWM,WWM,WOM,RHOWM,RHOOM,PCPM,
10916000      $              TMF,TOMF,TWMF,WWMF,WOMF
10917000       DIMENSION BWF(99),BOF(99),DF(99),PWF(99),POF(99),PCPF(99)
10918000       DIMENSION PHIBF(99),PHIF(99),RKOF(99),RKWF(99)
10919000       DIMENSION RKWPF(99),RKOPF(99),TCF(99),TOF(99),TWF(99)
10920000       DIMENSION SWF(99),WR(99),WWF(99),WOF(99),RHOWF(99)
10921000       DIMENSION RHOOF(99),A11(99),A12(99),A21(99),A22(99)
10922000       DIMENSION B11(99),B12(99),B21(99),B22(99),C11(99)
10923000       DIMENSION C12(99),C21(99),C22(99),WW(99),W1(99),W2(99),WD(99)
10924000       DIMENSION PI(99),PBH(99),PIW(99),PIO(99),PWH(99)
10925000       DIMENSION BWM(99),BOM(99),DM(99),PWM(99),POM(99),RHOOM(99)
10926000       DIMENSION PHIBM(99),PHIM(99),RKOM(99),RKWM(99),PCPM(99)
10927000       DIMENSION RKWPM(99),RKOPM(99),TCM(99),TOM(99),TWM(99)
10928000       DIMENSION SWM(99),WWM(99),WOM(99),RHOWM(99)
10929000       DIMENSION TMF(99),TWMF(99),TOMF(99),WWMF(99),WOMF(99)
10930000
10931000       DO 30 I=1,IMAX
10932000       B11(I) = 0
10933000       B12(I) = 0
10934000       B21(I) = 0
10935000       B22(I) = 0
10936000 C
10937000 C   CALCULATE CONTRIBUTIONS FROM I-1 NODES
10938000 C
10939000       IF (I.EQ.1) GO TO 21
$
```

```
10940000        AVGD = (RHOUF(I) + RHOUF(I-1)) / 288
10941000        DP1 = PUF(I)-PUF(I-1)-AVGD*(DF(I)-DF(I-1))
10942000        A11(I) =  TCF(I)*UUF(I)*(RKUF(I-1)/VIU)*(CU/BUF(I-1))*DP1
10943000       $          -TUF(I)*(1+RHOUB*CU/288*(DF(I)-DF(I-1)))
10944000        A12(I) = TCF(I)*UUF(I)*RKUPF(I-1)/VIU/BUF(I-1)*DP1
10945000        B11(I) =  TCF(I)*(1-UUF(I))*RKUF(I)/VIU*CU/BUF(I)*DP1
10946000       $          +TUF(I)*(1-RHOUB*CU/288*(DF(I)-DF(I-1)))
10947000        B12(I) =  TCF(I)*(1-UUF(I))*RKUPF(I)/VIU/BUF(I)*DP1
10948000        AVGD = (RHOOF(I) + RHOOF(I-1)) / 288
10949000        DPO1 = POF(I)-POF(I-1)-AVGD*(DF(I)-DF(I-1))
10950000        A21(I) =  TCF(I)*UOF(I)*(RKOF(I-1)/VIO)*(CO/BOF(I-1))*DPO1
10951000       $          -TOF(I)*(1+RHOOB*CO/288*(DF(I)-DF(I-1)))
10952000        A22(I) = TCF(I)*UOF(I)*RKOPF(I-1)/VIO/BOF(I-1)*DPO1
10953000       $          -TOF(I)*PCPF(I-1)
10954000        B21(I) =  TCF(I)*(1-UOF(I))*RKOF(I)/VIO*CO/BOF(I)*DPO1
10955000       $          +TOF(I)*(1-RHOOB*CO/288*(DF(I)-DF(I-1)))
10956000        B22(I) = TCF(I)*(1-UOF(I))*RKOPF(I)/VIO/BOF(I-1)*DPO1
10957000       $          + TOF(I)*PCPF(I)
10958000 C
10959000 C   CALCULATE CONTRIBUTIONS FROM I+1 NODES
10960000 C
10961000    21  IF (I.EQ.IMAX) GO TO 22
10962000        AVGD = (RHOUF(I+1) + RHOUF(I)) / 288
10963000        DP1 = PUF(I+1)-PUF(I)-AVGD*(DF(I+1)-DF(I))
10964000        C11(I) = -TCF(I+1)*(1-UUF(I+1))*(RKUF(I+1)/VIU)*(CU/BUF(I+1))*DP1
10965000       $          -TUF(I+1)*(1-RHOUB*CU/288*(DF(I+1)-DF(I)))
10966000        C12(I) = -TCF(I+1)*(1-UUF(I+1))*RKUPF(I+1)/VIU/BUF(I+1)*DP1
10967000        B11(I) = B11(I) -TCF(I+1)*UUF(I+1)*(RKUF(I)/VIU*CU/BUF(I)
10968000       $          *DP1) + TUF(I+1)*(1+RHOUB*CU/288*(DF(I+1)-DF(I)))
10969000        B12(I) = B12(I) - TCF(I+1)*UUF(I+1)*RKUPF(I)/VIU/BUF(I)
10970000       $          *DP1
10971000        AVGD = (RHOOF(I+1) + RHOOF(I)) / 288
10972000        DPO1 = POF(I+1)-POF(I)-AVGD*(DF(I+1)-DF(I))
10973000        C21(I) = -TCF(I+1)*(1-UOF(I+1))*(RKOF(I+1)/VIO)*(CO/BOF(I+1))*DPO1
10974000       $          -TOF(I+1)*(1+RHOOB*CO/288*(DF(I+1)-DF(I)))
10975000        C22(I) = -TCF(I+1)*(1-UOF(I+1))*RKOPF(I+1)/VIO/BOF(I+1)*DPO1
10976000       $          -TOF(I+1)*PCPF(I+1)
10977000        B21(I) = B21(I) -TCF(I+1)*UOF(I+1)*RKOF(I)/VIO*CO/BOF(I)
10978000       $          *DPO1 + TOF(I+1)*(1+RHOOB*CO/288*(DF(I+1)-DF(I)))
10979000        B22(I) = B22(I) - TCF(I+1)*UOF(I+1)*RKOPF(I)/VIO/BOF(I)*DPO1
10980000       $          +TOF(I+1)*PCPF(I)
10981000 C
10982000 C   ADD WELLBORE AND ACCUMULATION TERMS
10983000 C
10984000    22  CONTINUE
10985000        B11(I) = B11(I) - (UU(I)*RKOF(I)+RKUF(I))/VIU*CU/BUF(I)*(PBH(I)-
10986000       $          PUN(I))*PI(I) + PIU(I) + VR(I)/DT*SUF(I)*(PHIBF(I)*CF
10987000       $          +PHIF(I)*CU) / BUF(I)
10988000        B12(I) = B12(I) - PI(I)*(UU(I)*RKOPF(I)+RKUPF(I))*(PBH(I)-PUN(I))
10989000       $          /VIU/BUF(I) + VR(I)/DT*PHIF(I)/BUF(I)
10990000        B21(I) = -PI(I)*(1-UU(I))*RKOF(I)/VIO*CO/BOF(I)*(PBH(I)-PUN(I))
10991000       $          + PIO(I) + UR(I)*(1-SUF(I))/DT*(PHIBF(I)*CF+PHIF(I)
10992000       $          *CO)/BOF(I) + B21(I)
10993000        B22(I) =  -PI(I)*(1-UU(I))*RKOPF(I)/VIO/BOF(I)*(PBH(I)-PUN(I))
10994000       $          - VR(I)/DT*PHIF(I)/BOF(I)  + B22(I)
10995000        U1(I) =  PI(I)*((RKUF(I)+UU(I)*RKOF(I))*CU/VIU/BUF(I)+(1-UU(I))*
10996000       $         RKOF(I)*CO/BOF(I)/VIO) * (PBH(I)-PUN(I))-PIO(I)-PIU(I)
10997000        U2(I) =  PI(I)*((RKUPF(I)+UU(I)*RKOPF(I))/VIU/BUF(I)+(1-UU(I))*
10998000       $         RKOPF(I)/BOF(I)/VIO) * (PBH(I)-PUN(I))
10999000        UD(I) =  (PIU(I) + PIO(I))
11000000 C
11001000 C   NOW ADD CONTRIBUTION FROM MATRIX BLOCKS
11002000 C
11003000        DO 30 J=1,NMB
11004000        L = J + (I-1)*NMB
11005000        AVGD = (RHOUF(I) + RHOUM(L)) / 288
11006000        DPFM = PUF(I) - PUM(L) - AVGD*(DF(I) -DM(L))
11007000        B11(I) = B11(I) + TMF(L)*(1-UUMF(L))*(RKUF(I)/VIU*CU/BUF(I))*
11008000       $          DPFM + TUMF(L)*(1 + RHOUB*CU/288*(DF(I)-DM(L)))
11009000        B12(I) = B12(I) + TMF(L)*(1-UUMF(L))*RKUPF(I)/VIU/BUF(I)*DPFM
11010000        AVGD = (RHOOF(I) + RHOOM(L)) / 288
11011000        DPFM = POF(I) - POM(L) - AVGD*(DF(I) -DM(L))
11012000        B21(I) = B21(I) + TMF(L)*(1-UOMF(L))*(RKOF(I)/VIO*CO/BOF(I))*
11013000       $          DPFM + TOMF(L)*(1 + RHOOB*CO/288*(DF(I)-DM(L)))
11014000        B22(I) = B22(I) + TMF(L)*(1-UOMF(L))*RKOPF(I)/VIO/BOF(I)*DPFM
11015000       $          + TOMF(L)*PCPF(I)
11016000    30  CONTINUE
11017000        RETURN
11018000        END
$
```

```
11019000 C*************************************************************************
11020000 C                                                                        *
11021000 C            MMCOEF  -- CALCULATES TERMS FOR JACOBIAN MATRIX             *
11022000 C                                                                        *
11023000 C*************************************************************************
11024000 C
11025000 C     THIS SUBROUTINE CALCULATES PARTIAL DERIVATIVES OF MATRIX
11026000 C       RESIDUALS WITH RESPECT TO MATRIX VARIABLES FOR USE IN THE
11027000 C       JACOBIAN MATRIX.  THE COEFFICIENTS CALCULATED ARE AS FOLLOWS:
11028000 C
11029000 C          AM11 -- PARTIAL DERIVATIVE OF MATRIX WATER RESIDUAL WITH
11030000 C                  RESPECT TO MATRIX WATER PRESSURE OF NODE I-1.
11031000 C          AM12 -- PARTIAL DERIVATIVE OF MATRIX WATER RESIDUAL WITH
11032000 C                  RESPECT TO MATRIX WATER SATURATION OF NODE I-1.
11033000 C          AM21 -- PARTIAL DERIVATIVE OF MATRIX OIL RESIDUAL WITH
11034000 C                  RESPECT TO MATRIX WATER PRESSURE OF NODE I-1.
11035000 C          AM22 -- PARTIAL DERIVATIVE OF MATRIX OIL RESIDUAL WITH
11036000 C                  RESPECT TO MATRIX WATER SATURATION OF NODE I-1.
11037000 C          BM11 -- PARTIAL DERIVATIVE OF MATRIX WATER RESIDUAL WITH
11038000 C                  RESPECT TO MATRIX WATER PRESSURE OF NODE I.
11039000 C          BM12 -- PARTIAL DERIVATIVE OF MATRIX WATER RESIDUAL WITH
11040000 C                  RESPECT TO MATRIX WATER SATURATION OF NODE I.
11041000 C          BM21 -- PARTIAL DERIVATIVE OF MATRIX OIL RESIDUAL WITH
11042000 C                  RESPECT TO MATRIX WATER PRESSURE OF NODE I.
11043000 C          BM22 -- PARTIAL DERIVATIVE OF MATRIX OIL RESIDUAL WITH
11044000 C                  RESPECT TO MATRIX WATER SATURATION OF NODE I.
11045000 C          CM11 -- PARTIAL DERIVATIVE OF MATRIX WATER RESIDUAL WITH
11046000 C                  RESPECT TO MATRIX WATER PRESSURE OF NODE I+1.
11047000 C          CM12 -- PARTIAL DERIVATIVE OF MATRIX WATER RESIDUAL WITH
11048000 C                  RESPECT TO MATRIX WATER SATURATION OF NODE I+1.
11049000 C          CM21 -- PARTIAL DERIVATIVE OF MATRIX OIL RESIDUAL WITH
11050000 C                  RESPECT TO MATRIX WATER PRESSURE OF NODE I+1.
11051000 C          CM22 -- PARTIAL DERIVATIVE OF MATRIX OIL RESIDUAL WITH
11052000 C                  RESPECT TO MATRIX WATER SATURATION OF NODE I+1.
11053000 C
11054000 C          SEE THE MAIN PROGRAM FOR DEFINITION OF OTHER VARIABLES.
11055000 C
11056000 C*************************************************************************
11057000 C
11058000       SUBROUTINE MMCOEF
11059000       COMMON /CONS/CW,CO,NWELL,DT,IMAX,NMB,UIO,UIW,RHOUB,RHOOB,CF,CM
11060000       COMMON /MATRIX/BWM,BOM,DM,PWM,POM,PHIBM,PHIM,RKOM,RKWM,RKWPM,
11061000      $               RKOPM,TCM,TOM,TWM,SWM,WWM,WOM,RHOWM,RHOOM,PCPM,
11062000      $               TMF,TOMF,TWMF,WWMF,WOMF
11063000       COMMON /COEF2/AM11,AM12,AM21,AM22,BM11,BM12,BM21,BM22,CM11,CM12,
11064000      $              CM21,CM22
11065000       COMMON /FRAC/BWF,BOF,DF,PWF,POF,PHIBF,PHIF,RKOF,RKWF,RKWPF,PCPF,
11066000      $             RKOPF,TCF,TOF,TWF,SWF,WR,WWF,WOF,RHOWF,RHOOF
11067000       DIMENSION BWF(99),BOF(99),DF(99),PWF(99),POF(99),PCPF(99)
11068000       DIMENSION PHIBF(99),PHIF(99),RKOF(99),RKWF(99)
11069000       DIMENSION RKWPF(99),RKOPF(99),TCF(99),TOF(99),TWF(99)
11070000       DIMENSION SWF(99),WR(99),WWF(99),WOF(99),RHOWF(99)
11071000       DIMENSION RHOOF(99),AM11(99),AM12(99),AM21(99),AM22(99)
11072000       DIMENSION BM11(99),BM12(99),BM21(99),BM22(99),CM11(99)
11073000       DIMENSION CM12(99),CM21(99),CM22(99),WW(99),W1(99),W2(99),WD(99)
11074000       DIMENSION PI(99),PBH(99),PIW(99),PIO(99)
11075000       DIMENSION BWM(99),BOM(99),DM(99),PWM(99),POM(99),RHOOM(99)
11076000       DIMENSION PHIBM(99),PHIM(99),RKOM(99),RKWM(99),PCPM(99)
11077000       DIMENSION RKWPM(99),RKOPM(99),TCM(99),TOM(99),TWM(99)
11078000       DIMENSION SWM(99),WWM(99),WOM(99),RHOWM(99)
11079000       DIMENSION TMF(99),TWMF(99),TOMF(99),WWMF(99),WOMF(99)
11080000
11081000       DO 30 L=1,NMB*IMAX
11082000       BM11(L) = 0
11083000       BM12(L) = 0
11084000       BM21(L) = 0
11085000       BM22(L) = 0
11086000       IF (NMB.EQ.1) GO TO 32
$
```

```
11087000 C
11088000 C   CALCULATE CONTRIBUTIONS FROM L-1 NODES
11089000 C
11090000       IF (MOD(L,NMB).EQ.1) GO TO 21
11091000       AVGD = (RHOUM(L) + RHOUM(L-1)) / 288
11092000       DP1 = PUM(L)-PUM(L-1)-AVGD*(DM(L)-DM(L-1))
11093000       AM11(L) =  TCM(L)*UUM(L)*(RKUM(L-1)/VIU)*(CU/BUM(L-1))*DP1
11094000      $          -TUM(L)*(1+RHOUB*CU/288*(DM(L)-DM(L-1)))
11095000       AM12(L) = TCM(L)*UUM(L)*RKUPM(L-1)/VIU/BUM(L-1)*DP1
11096000       BM11(L) =  TCM(L)*(1-UUM(L))*RKUM(L)/VIU*CU/BUM(L)*DP1
11097000      $          +TUM(L)*(1-RHOUB*CU/288*(DM(L)-DM(L-1)))
11098000       BM12(L) = TCM(L)*(1-UUM(L))*RKUPM(L)/VIU/BUM(L)*DP1
11099000       AVGD = (RHOOM(L) + RHOOM(L-1)) / 288
11100000       DPO1 = POM(L)-POM(L-1)-AVGD*(DM(L)-DM(L-1))
11101000       AM21(L) =  TCM(L)*UOM(L)*(RKOM(L-1)/VIO)*(CO/BOM(L-1))*DPO1
11102000      $          -TOM(L)*(1+RHOOB*CO/288*(DM(L)-DM(L-1)))
11103000       AM22(L) = TCM(L)*UOM(L)*RKOPM(L-1)/VIO/BOM(L-1)*DPO1
11104000      $          - TOM(L)*PCPM(L-1)
11105000       BM21(L) =  TCM(L)*(1-UOM(L))*RKOM(L)/VIO*CO/BOM(L)*DPO1
11106000      $          +TOM(L)*(1-RHOOB*CO/288*(DM(L)-DM(L-1)))
11107000       BM22(L) = TCM(L)*(1-UOM(L))*RKOPM(L)/VIO/BOM(L)*DPO1
11108000      $           + TOM(L)*PCPM(L)
11109000 C
11110000 C   CALCULATE CONTRIBUTIONS FROM L+1 NODES
11111000 C
11112000    21 IF (MOD(L,NMB).EQ.0) GO TO 22
11113000       AVGD = (RHOUM(L+1) + RHOUM(L)) / 288
11114000       DP1 = PUM(L+1)-PUM(L)-AVGD*(DM(L+1)-DM(L))
11115000       CM11(L) = -TCM(L+1)*(1-UUM(L+1))*(RKUM(L+1)/VIU)*(CU/BUM(L+1))
11116000      $          *DP1-TUM(L+1)*(1-RHOUB*CU/288*(DM(L+1)-DM(L)))
11117000       CM12(L) = TCM(L+1)*(1-UUM(L+1))*RKUPM(L+1)/VIU/BUM(L+1)*DP1
11118000       BM11(L) = BM11(L) -TCM(L+1)*UUM(L+1)*(RKUM(L)/VIU*CU/BUM(L)
11119000      $          *DP1) + TUM(L+1)*(1+RHOUB*CU/288*(DM(L+1)-DM(L)))
11120000       BM12(L) = BM12(L) - TCM(L+1)*UUM(L+1)*RKUPM(L+1)/VIU/BUM(L+1)
11121000      $          *DP1
11122000       AVGD = (RHOOM(L+1) + RHOOM(L)) / 288
11123000       DPO1 = POM(L+1)-POM(L)-AVGD*(DM(L+1)-DM(L))
11124000       CM21(L) = -TCM(L+1)*(1-UOM(L+1))*(RKOM(L+1)/VIO)*(CO/BOM(L+1))
11125000      $          *DPO1 -TOM(L+1)*(1+RHOOB*CO/288*(DM(L+1)-DM(L)))
11126000       CM22(L) = -TCM(L+1)*(1-UOM(L+1))*RKOPM(L+1)/VIO/BOM(L+1)*DPO1
11127000      $           - TOM(L+1)*PCPM(L+1)
11128000       BM21(L) = BM21(L) -TCM(L+1)*UOM(L+1)*RKOM(L)/VIO*CO/BOM(L)
11129000      $          *DPO1 + TOM(L+1)*(1+RHOOB*CO/288*(DM(L+1)-DM(L)))
11130000       BM22(L) = BM22(L) - TCM(L+1)*UOM(L+1)*RKOPM(L)/VIO*BOM(L)*DPO1
11131000      $           + TOM(L+1)*PCPM(L)
11132000 C
11133000 C   ADD ACCUMULATION TERMS
11134000 C
11135000    22 CONTINUE
11136000       I = L / NMB + 1
11137000       IF (MOD(L,NMB).EQ.0) I=I-1
11138000       BM11(L)=  VR(I)/NMB/DT*SUM(L)*(PHIBM(L)*CM+PHIM(L)*CU)/BUM(L) +
11139000      $          BM11(L)
11140000       BM12(L) = BM12(L) +   VR(I)/NMB/DT*PHIM(L)/BUM(L)
11141000       BM21(L) = BM21(L) +  VR(I)/NMB*(1-SUM(L))/DT*(PHIBM(L)*CM+PHIM(L)
11142000      $          *CO)/BOM(L)
11143000       BM22(L) = BM22(L) - VR(I)/NMB/DT*PHIM(L)/BOM(L)
11144000 C
11145000 C    NOW ADD CONTRIBUTION FROM FRACTURES
11146000 C
11147000       AVGD = (RHOUF(I) + RHOUM(L)) / 288
11148000       DPMF = PUF(I) - PUM(L) - AVGD*(DF(I) -DM(L))
11149000       BM11(L) = BM11(L) - TMF(L)*UUMF(L)*(RKUM(L)/VIU*CU/BUM(L))*
11150000      $          DPMF + TUMF(L)*(1 - RHOUB*CU/288*(DF(I)-DM(L)))
11151000       BM12(L) = BM12(L) - TMF(L)*UUMF(L)*RKUPM(L)/VIU/BUM(L)*DPMF
11152000       AVGD = (RHOOF(I) + RHOOM(L)) / 288
11153000       DPMF = POF(I) - POM(L) - AVGD*(DF(I) -DM(L))
11154000       BM21(L) = BM21(L) - TMF(L)*UOMF(L)*(RKOM(L)/VIO*CO/BOM(L))*
11155000      $          DPMF + TOMF(L)*(1 - RHOOB*CO/288*(DF(I)-DM(L)))
11156000       BM22(L) = BM22(L) - TMF(L)*UOMF(L)*RKOPM(L)/VIO/BOM(L)*DPMF
11156500      $          +TOMF(L)*PCPM(L)
11157000    30 CONTINUE
11158000       RETURN
11159000       END
@
```

```
C**********************************************************************
C                                                                     *
C           MFCOEF  -- CALCULATES TERMS FOR JACOBIAN MATRIX           *
C                                                                     *
C**********************************************************************
C
C     THIS SUBROUTINE CALCULATES PARTIAL DERIVATIVES OF MATRIX
C       RESIDUALS WITH RESPECT TO FRACTURE VARIABLES AND PARTIAL
C       DERIVATIVES OF FRACTURE RESIDUALS WITH RESPECT TO MATRIX
C      VARIABLES. THESE TERMS ARE THE LOWER-LEFT AND UPPER-RIGHT OF THE
C       JACOBIAN MATRIX. THE COEFFICIENTS CALCULATED ARE AS FOLLOWS:
C
C         B(I,J2) -- PARTIAL DERIVATIVE OF FRACTURE WATER RESIDUAL WITH
C                RESPECT TO MATRIX WATER PRESSURE OR WATER SATURATION
C         B(I,JP) -- PARTIAL DERIVATIVE OF FRACTURE OIL RESIDUAL WITH
C                RESPECT TO MATRIX WATER PRESSURE OR WATER SATURATION
C         B(J2,I) -- PARTIAL DERIVATIVE OF MATRIX WATER RESIDUAL WITH
C                RESPECT TO FRACTURE WATER PRESSURE OR WATER SATURATION
C         B(JP,I) -- PARTIAL DERIVATIVE OF MATRIX OIL RESIDUAL WITH
C                RESPECT TO FRACTURE WATER PRESSURE OR WATER SATURATION
C
C          SEE THE MAIN PROGRAM FOR DEFINITION OF OTHER VARIABLES.
C
C**********************************************************************
C
      SUBROUTINE MFCOEF
      COMMON /CONS/CW,CO,NWELL,DT,IMAX,NMB,VIO,VIW,RHOWB,RHOOB,CF,CM
      COMMON /FRAC/BWF,BOF,DF,PWF,POF,PHIBF,PHIF,RKOF,RKWF,RKWPF,PCPF,
     $             RKOPF,TCF,TOF,TWF,SWF,VR,WWF,WOF,RHOWF,RHOOF
      COMMON /COEF3/ B
      COMMON /WELL/ PI,PBH,PIW,PIO,WW,W1,W2,WD,PWN
      COMMON /MATRIX/BWM,BOM,DM,PWM,POM,PHIBM,PHIM,RKOM,RKWM,RKWPM,
     $             RKOPM,TCM,TOM,TWM,SWM,WWM,WOM,RHOWM,RHOOM,PCPM,
     $             TMF,TOMF,TWMF,WWMF,WOMF
      DIMENSION BWF(99),BOF(99),DF(99),PWF(99),POF(99),PCPF(99)
      DIMENSION PHIBF(99),PHIF(99),RKOF(99),RKWF(99)
      DIMENSION RKWPF(99),RKOPF(99),TCF(99),TOF(99),TWF(99)
      DIMENSION SWF(99),VR(99),WWF(99),WOF(99),RHOWF(99)
      DIMENSION RHOOF(99),B(25,25,50),WW(99)
      DIMENSION PI(99),PBH(99),PIW(99),PIO(99),W1(99),W2(99),WD(99)
      DIMENSION BWM(99),BOM(99),DM(99),PWM(99),POM(99),RHOOM(99)
      DIMENSION PHIBM(99),PHIM(99),RKOM(99),RKWM(99),PCPM(99)
      DIMENSION RKWPM(99),RKOPM(99),TCM(99),TOM(99),TWM(99)
      DIMENSION SWM(99),WWM(99),WOM(99),RHOWM(99)
      DIMENSION TMF(99),TWMF(99),TOMF(99),WWMF(99),WOMF(99)

      DO 10 I=1,IMAX
      DO 10 J=1,NMB
      L = J + (I-1)*NMB
      J2 = 2*J + 1
      JP = 2*J + 2
```
8

```
11211000 C
11212000 C  CALCULATE DERIVATIVE OF FRACTURE RESIDUALS WITH RESPECT TO MATRIX
11213000 C     VARIABLES.
11214000 C
11215000       AUGD = (RHOUF(I) + RHOUM(L)) / 288
11216000       DPMF = PUF(I) - PUM(L) - AUGD*(DF(I) -DM(L))
11217000       B(1,J2,I) = TMF(L)*UUMF(L)*RKUM(L)/VIU*CU/BUM(L)*DPMF
11218000      $          - TUMF(L)*(1-RHOUB*CU/288*(DF(I)-DM(L)))
11219000       B(1,JP,I) = TMF(L)*UUMF(L)*RKUPM(L)/VIU/BUM(L)*DPMF
11220000       AUGD = (RHOOF(I) + RHOOM(L)) / 288
11221000       DPMF = POF(I) - POM(L) - AUGD*(DF(I)-DM(L))
11222000       B(2,J2,I) = TMF(L)*UOMF(L)*RKOM(L)/VIO*CO/BOM(L)*DPMF
11223000      $          - TOMF(L)*(1-RHOOB*CO/288*(DF(I)-DM(L)))
11224000       B(2,JP,I) = TMF(L)*UOMF(L)*RKOPM(L)/VIO/BOM(L)*DPMF
11225000      $          - TOMF(L)*PCPM(L)
11226000 C
11227000 C  CALCULATE DERIVATIVE OF MATRIX RESIDUALS WITH RESPECT TO FRACTURE
11228000 C     VARIABLES.
11229000 C
11230000       AUGD = (RHOUF(I) + RHOUM(L)) / 288
11231000       DPMF = PUF(I) - PUM(L) - AUGD*(DF(I) -DM(L))
11232000       B(J2,1,I) = -TMF(L)*(1-UUMF(L))*RKUF(I)/VIU*CU/BUF(I)*DPMF
11233000      $          - TUMF(L)*(1+RHOUB*CU/288*(DF(I)-DM(L)))
11234000       B(J2,2,I) = -TMF(L)*(1-UUMF(L))*RKUPF(I)/VIU/BUF(I)*DPMF
11235000       AUGD = (RHOOF(I) + RHOOM(L)) / 288
11236000       DPMF = POF(I) - POM(L) - AUGD*(DF(I) -DM(L))
11237000       B(JP,1,I) = -TMF(L)*(1-UOMF(L))*RKOF(I)/VIO*CO/BOF(I)*DPMF
11238000      $          - TOMF(L)*(1+RHOOB*CO/288*(DF(I)-DM(L)))
11239000       B(JP,2,I) = -TMF(L)*(1-UOMF(L))*RKOPF(I)/VIO/BOF(I)*DPMF
11240000      $          - TOMF(L)*PCPF(I)
11241000    10 CONTINUE
11242000       RETURN
11243000       END
@
```

```
11244000 C
11245000 C*************************************************************************
11246000 C                                                                        *
11247000 C                RESID -- CALCULATES RESIDUAL R.H.S. VECTOR              *
11248000 C                                                                        *
11249000 C*************************************************************************
11250000 C
11251000 C         THIS SUBROUTINE CALCULATES THE RIGHT-HAND-SIDE VETOR. THE
11252000 C          TERMS IN THIS VECTOR ARE THE RESIDUALS CALCULATED FROM
11253000 C          THE LAST ITERATION VALUES.
11254000 C
11255000 C        PARAMETERS:
11256000 C
11257000 C          M -- NUMBER OF CONNECTED MATRIX BLOCKS
11258000 C          N -- NUMBER OF NODES
11259000 C          R1 -- RESIDUAL OF FRACTURE POROSITY
11260000 C          R2 -- RESIDUAL OF MATRIX POROSITY
11261000 C          RHO1 -- DENSITY IN FRACTURE POROSITY
11262000 C          RHO2 -- DENSITY IN MATRIX POROSITY
11263000 C          P1 -- PRESSURE IN FRACTURE POROSITY
11264000 C          P2 -- PRESSURE IN MATRIX POROSITY
11265000 C          D1 -- DEPTH OF FRACTURE POROSITY
11266000 C          D2 -- DEPTH OF MATRIX POROSITY
11267000 C          T1 -- TRANSMISSIBILITY BETWEEN NODES IN FRACTURE POROSITY
11268000 C          T2 -- TRANSMISSIBILITY BETWEEN NODES IN MATRIX POROSITY
11269000 C          T12 -- TRANSMISSIBILITY BETWEEN FRACTURE AND MATRIX POROSITY
11270000 C          U -- PORE VOLUME
11271000 C          UN -- PORE VOLUME AT LAST TIME STEP
11272000 C
11273000 C*************************************************************************
11274000 C
11275000       SUBROUTINE RESID(M,N,R1,R2,RHO1,RHO2,P1,P2,D1,D2,T1,T2,T12,U,UN,
11276000      $                 U2,U2N,PI,PBH,PUN,DT)
11277000       DIMENSION R1(1),RHO1(1),RHO2(1),P1(1),P2(1),D1(1),D2(1),T1(1),
11278000      $          R2(1),T2(1),T12(1),U(1),UN(1),PI(1),PBH(1),PUN(1),
11279000      $          U2(1),U2N(1)
11280000
11281000       DO 40 I=1,M
11282000       R1(I) = 0
11283000       IF (I.EQ.N) GO TO 21
11284000       AVGD = (RHO1(I+1)+RHO1(I)) / 288
11285000       R1(I) = T1(I+1)*(P1(I+1)-P1(I)-AVGD*(D1(I+1)-D1(I)))
11286000    21 IF (I.EQ.1) GO TO 22
11287000       AVGD = (RHO1(I)+RHO1(I-1)) / 288
11288000       R1(I) = R1(I) - T1(I)*(P1(I)-P1(I-1)-AVGD*(D1(I)-D1(I-1)))
11289000    22 R1(I) = R1(I) - (U(I) - UN(I)) / DT  +   PI(I)*(PBH(I)-PUN(I))
11290000       DO 40 J=1,M
11291000       L = J + (I-1)*M
11292000       AVGD = (RHO1(I)+RHO2(L)) / 288
11293000       R1(I) = R1(I) + T12(L)*(P2(L)-P1(I)-AVGD*(D2(L)-D1(I)))
11294000       R2(L) = T12(L)*(P2(L)-P1(I)-AVGD*(D2(L)-D1(I)))
11295000       IF (M.EQ.1) GO TO 33
11296000       IF (MOD(L,M).EQ.0) GO TO 32
11297000       AVGD = (RHO2(L+1)+RHO2(L)) / 288
11298000       R2(L) = T2(L+1)*(P2(L+1)-P2(L)-AVGD*(D2(L+1)-D2(L)))+R2(L)
11299000    32 IF (MOD(L,M).EQ.1) GO TO 33
11300000       AVGD = (RHO2(L)+RHO2(L-1)) / 288
11301000       R2(L) = R2(L) - T2(L)*(P2(L)-P2(L-1)-AVGD*(D2(L)-D2(L-1)))
11302000    33 R2(L) = R2(L) - (U2(L) - U2N(L)) / DT
11303000    40 CONTINUE
11304000       RETURN
11305000       END
$
```

```
C
C***********************************************************************
C
C             PELIM -- ELIMINATES MATRIX TERMS FROM JACOBIAN
C
C***********************************************************************
C
      SUBROUTINE PELIM(IMAX,NMB)
      COMMON /RHS/ RFW.RFO,RMW,RMO.RWL
      COMMON /COEFF/A11.A12.A21.A22.B11.B12.B21.B22.C11.C12.C21.C22
      COMMON /COEF2/AM11.AM12.AM21.AM22.BM11.BM12.BM21.BM22.CM11.CM12.
     $            CM21.CM22
      COMMON /COEF3/ B
      COMMON /WELL/ PI.PBH.PIW.PIO.WW.W1.W2.WD.PWN
      DIMENSION RFW(99).RMW(99).RFO(99).RMO(99).RWL(99)
      DIMENSION A11(99).A12(99).A21(99).A22(99)
      DIMENSION B11(99).B12(99).B21(99).B22(99).C11(99)
      DIMENSION C12(99).C21(99).C22(99)
      DIMENSION AM11(99).AM12(99).AM21(99).AM22(99)
      DIMENSION BM11(99).BM12(99).BM21(99).BM22(99).CM11(99)
      DIMENSION CM12(99).CM21(99).CM22(99)
      DIMENSION B(25.25.50).WW(99)
      DIMENSION PI(99).PBH(99).PIW(99).PIO(99).W1(99).W2(99).WD(99)
C
      MC = 2*NMB + 3
      DO 30 I=1.IMAX
      B(1.1.I) = B11(I)
      B(2.1.I) = B21(I)
      B(1.2.I) = B12(I)
      B(2.2.I) = B22(I)
      B(1.MC.I) = RFW(I)
      B(2.MC.I) = RFO(I)
      DO 10 J=1.NMB
      L = J +(I-1)*NMB
      IF (BM11(L).EQ.0.0) BM11(L) = 1.0
      IF (BM22(L).EQ.0.0) BM22(L) = 1.0
      J2 = J * 2
      J2P1 = J2 + 1
      J2P2 = J2 + 2
      B(J2P1.MC.I) = RMW(L)
      B(J2P2.MC.I) = RMO(L)
      B(J2P1.J2P1.I) = BM11(L)
      B(J2P2.J2P1.I) = BM21(L)
      B(J2P1.J2P2.I) = BM12(L)
      B(J2P2.J2P2.I) = BM22(L)
      IF (J.EQ.1) GO TO 5
      B(J2P1.J2-1.I) = AM11(L)
      B(J2P2.J2-1.I) = AM21(L)
      B(J2P1.J2.I)   = AM12(L)
      B(J2P2.J2.I)   = AM22(L)
    5 IF (J.EQ.NMB) GO TO 10
      B(J2P1.J2+3.I) = CM11(L)
      B(J2P2.J2+3.I) = CM21(L)
      B(J2P1.J2+4.I) = CM12(L)
      B(J2P2.J2+4.I) = CM22(L)
   10 CONTINUE
```

```
11361000       DO 30 KK = 3,MC-1
11362000       K = MC - KK + 2
11363000       D = B(K,K,I)
11364000       DO 15 M = 1,MC
11365000       B(K,M,I) = B(K,M,I) / D
11366000    15 CONTINUE
11367000       DO 20 II = 1,K-1
11368000       KI = K - II
11369000       Z = B(KI,K,I)
11370000       DO 20 J = 1,MC
11371000       B(KI,J,I) = B(KI,J,I) - ZXB(K,J,I)
11372000    20 CONTINUE
11373000       B11(I) = B(1,1,I)
11373100       B21(I) = B(2,1,I)
11373200       B12(I) = B(1,2,I)
11373300       B22(I) = B(2,2,I)
11373400       RFU(I) = B(1,MC,I)
11373500       RFO(I) = B(2,MC,I)
11374000       IF (UD(I).EQ.0) GO TO 30
11375000       V1 = -PIU(I) / UD(I)
11376000       V2 = -PIO(I) / UD(I)
11377000       B11(I) = B11(I) - V1*U1(I)
11378000       B12(I) = B12(I) - V1*U2(I)
11379000       B21(I) = B21(I) - V2*U1(I)
11380000       B22(I) = B22(I) - V2*U2(I)
11381000       RFU(I) = RFU(I) - V1*RUL(I)
11382000       RFO(I) = RFO(I) - V2*RUL(I)
11383000    30 CONTINUE
11384000       RETURN
11385000       END
$
```

```
                                                                      $
11386000 C*************************************************************************
11387000 C
11388000 C            SETUP -- LOADS MATRIX FOR GBAND
11389000 C
11390000 C*************************************************************************
11391000 C
11392000       SUBROUTINE SETUP(A,D,N)
11393000       DIMENSION A(1),D(1)
11394000       COMMON /COEFF/A11,A12,A21,A22,B11,B12,B21,B22,C11,C12,C21,C22
11395000       COMMON /RHS/ RFW,RFO,RMW,RMO,RUL
11396000       DIMENSION A11(99),A12(99),A21(99),A22(99)
11397000       DIMENSION B11(99),B12(99),B21(99),B22(99),C11(99)
11398000       DIMENSION C12(99),C21(99),C22(99)
11399000       DIMENSION RFW(99),RMW(99),RFO(99),RMO(99),RUL(99)
11400000 C
11401000       IC = 0
11402000       DO 40 I=1,N
11402100       IF (B11(I).EQ.0.0) B11(I) = 1.0
11402200       IF (B22(I).EQ.0.0) B22(I) = 1.0
11403000       J1 = 1
11404000       J2 = 7
11405000       IF(I.LT.3) J1 = 6-2*I
11406000       IF(I.EQ.N) J2 = 5
11407000       DO 20 J=J1,J2
11408000       IC = IC + 1
11409000       GO TO(20,11,12,13,14,15,16) J
11410000    11 A(IC) = A11(I)
11411000       GO TO 20
11412000    12 A(IC) = A12(I)
11413000       GO TO 20
11414000    13 A(IC) = B11(I)
11415000       GO TO 20
11416000    14 A(IC) = B12(I)
11417000       GO TO 20
11418000    15 A(IC) = C11(I)
11419000       GO TO 20
11420000    16 A(IC) = C12(I)
11421000    20 CONTINUE
11422000       J1 = 1
11423000       IF(I.EQ.1) J1=3
11424000       IF(I.GT.N-2) J2=4+2*(N-I)
11425000       DO 30 J=J1,J2
11426000       IC = IC + 1
11427000       GO TO(21,22,23,24,25,26,30) J
11428000    21 A(IC) = A21(I)
11429000       GO TO 30
11430000    22 A(IC) = A22(I)
11431000       GO TO 30
11432000    23 A(IC) = B21(I)
11433000       GO TO 30
11434000    24 A(IC) = B22(I)
11435000       GO TO 30
11436000    25 A(IC) = C21(I)
11437000       GO TO 30
11438000    26 A(IC) = C22(I)
11439000    30 CONTINUE
11440000       I2 = I*2
11441000       I2M = I2-1
11442000       D(I2M) = RFW(I)
11443000       D(I2) = RFO(I)
11444000    40 CONTINUE
11445000       RETURN
11446000       END
```

```
C
C**********************************************************************
C                                                                     *
C                   QBAND -- MATRIX INVERSION ROUTINE                 *
C                                                                     *
C**********************************************************************
C
C             SOLUTION OF SYSTEM OF EQUATIONS WITH BAND MATRIX BY STANDARD
C              ELIMINATION WITHOUT PIVOTING.
C
C             PARAMETERS.
C
C              A -- ONE-DIMENSIONAL ARRAY CONTAINING THE BAND OF THE MATRIX
C                   STORED BY ROWS. THEN REQUIRED DIMENSION OF A IS
C                   N*(2*M+1) - M*M - M.
C              M -- NUMBER OF DIAGONALS ABOVE THE MAIN DIAGONAL. THE NUMBER
C                   OF DIAGONALS BELOW THE MAIN DIAGONAL IF ALSO M, THERE-
C                   FORE THE TOTAL BANDWIDTH IS 2M+1.
C              N -- NUMBER OF EQUATIONS.
C              D -- RIGHT-HAND-SIDE VECTOR.
C              X -- SOLUTION VECTOR
C              EPS -- THE ELEMENT ON THE MAIN DAIGONAL IS COMPARED TO EPS
C                     DURING ELIMINATION. IF IT IS SMALLER, THE VALUE OF
C                     THE COUNTER IERR IS INCREMENTED.
C              IERR -- NUMBER OF TIMES THE ELEMENT ON THE MAIN DIAGONAL WAS
C                      SMALLER THAN EPS.
C              IFRST -- IF IFRST = 0 THEN THE MATRIX IS INVERTED AND THE
C                       INVERSE IS STORED IN PLACE OF THE ORIGINAL MATRIX.
C                       IF IFRST IS GREATER THAN 0, THE ROUTINE ASSUMES
C                       THAT THE MATRIX HAS BEEN INVERTED AND WILL CALCU-
C                       LATE A SOLUTION CORRESPONDING TO THE NEW R.H.S.
C                       VECTOR. THIS OPTION IS USEFUL WHEN SOLVING REPEAT-
C                       EDLY THE SAME EQUATIONS WITH DIFFERENT R.H.S.
C                       VECTORS.
C
C**********************************************************************
C
      SUBROUTINE QBAND(A,D,X,N,M,EPS,IERR,IFRST)
      DIMENSION A(1),D(1),X(1)
C
      IERR = 0
      J = 1
      DO 10 I = 1,N
      IE = M
      IF(I+M-N) 21,21,22
   22 IE = N-I
   21 IEAUX = M
      IF(I-M) 23,23,24
   23 IEAUX = I
   24 IE1 = IE + IEAUX
      MBIG = IE
      J1 = J+IE1
      J2 = J1
      IF(IFRST.GT.0) GO TO 27
      IF(ABS(A(J))-EPS) 25,25,27
   25 IERR = IERR + 1
   27 IF(MBIG) 10,10,26
   26 DO 20 JO = 1,MBIG
      S = A(J1) / A(J)
      IF (IFRST.GT.0) GO TO 35
      DO 30 K = 1,MBIG
      J1K = J1 + K
      JK = J + K
   30 A(J1K) = A(J1K) - A(JK)*S
   35 CONTINUE
      IAUX = JO + I
      D(IAUX) = D(IAUX) - D(I)*S
      IE = M
      IF (IAUX+M-N) 31,31,32
   32 IE = N - IAUX
   31 IEAUX = M
      IF (IAUX-M) 33,33,34
   33 IEAUX = IAUX
   34 IE1 = IE + IEAUX
   20 J1 = J1 + IE1
   10 J = J2 + 1
$
```

```
11523000          J = J - M - 1
11524000          MP1 = N + 1
11525000          DO 40 IIIMU = 1,N
11526000          I = MP1 - IIIMU
11527000          IE = M
11528000          IF (I+M-N) 41,41,42
11529000       42 IE = N - I
11530000       41 MBIG = IE
11531000          X(I) = D(I)
11532000          IF (MBIG) 44,44,43
11533000       43 DO 50 K=1,MBIG
11534000          IK = I + K
11535000          JK = J + K
11536000       50 X(I) = X(I) - X(IK)*A(JK)
11537000       44 X(I) = X(I)/A(J)
11538000          IE = M
11539000          IF (I+M-MP1) 51,51,52
11540000       52 IE = MP1 - I
11541000       51 IEAUX = M
11542000          IF(I-1-M) 53,53,54
11543000       53 IEAUX = I - 1
11544000       54 IE1 = IE + IEAUX
11545000          J = J - IE1 - 1
11546000       40 CONTINUE
11547000          RETURN
11548000          END
$
```

```
11549000 C
11550000 C*************************************************************************
11551000 C
11552000 C      UPDATE - SOLVES FOR MATRIX UNKNOWNS AND UPDATES ALL VARIABLES
11553000 C
11554000 C*************************************************************************
11555000 C
11556000       SUBROUTINE UPDATE(X,PUF,SUF,PUN,SUN,QUELL,QU,QO,M,N,CONVRG)
11557000       DIMENSION X(1),PUF(1),SUF(1),PUN(1),SUN(1),QU(1),QO(1),QUELL(1)
11558000       COMMON /RHS/ RFU,RFO,RNU,RNO,RUL
11559000       COMMON /UELL/ PI,PBH,PIU,PIO,UU,U1,U2,UD,PUN
11560000       COMMON /COEF3/ B
11561000       DIMENSION B(25,25,50),PUN(99),UU(99),Y(99)
11562000       DIMENSION RFU(99),RNU(99),RFO(99),RNO(99),RUL(99)
11563000       DIMENSION PI(99),PBH(99),PIU(99),PIO(99),U1(99),U2(99),UD(99)
11564000       LOGICAL CONVRG
11565000 C
11566000       DO 40 I=1,M
11567000       I2M = 2*I-1
11568000       I2 = 2*I
11569000       PUF(I) = PUF(I) + X(I2M)
11570000       SUF(I) = SUF(I) + X(I2)
11570100       IF (SUF(I).LT.0.0) SUF(I) = 0.0
11570200       IF (SUF(I).GT.1.0) SUF(I) = 1.0
11571000       IF (ABS(X(I2M)).GT.0.001) CONVRG = .FALSE.
11572000       IF (ABS(X(I2)).GT.0.0001) CONVRG = .FALSE.
11573000       IF (UD(I).EQ.0) GO TO 15
11574000       DP = (RUL(I)-U1(I)*X(I2M)-U2(I)*X(I2)) / UD(I)
11575000       IF (ABS(DP).GT.0.001) CONVRG = .FALSE.
11576000       PBH(I) = PBH(I) + DP
11577000       PUN(I) = PUF(I)
11578000       GO TO 17
11579000    15 CONTINUE
11580000       PBH(I) = PUF(I)
11581000       PUN(I) = PUF(I)
11582000    17 CONTINUE
11582500       IF (UD(I).NE.0) PBH(I) = QUELL(I) / UD(I) + PUN(I)
11583000       QO(I) = PIO(I) * (PBH(I)-PUN(I))
11584000       QU(I) = PIU(I) * (PBH(I)-PUN(I))
11586000       MC = 2*M + 3
11588000       Y(1) = X(I2M)
11589000       Y(2) = X(I2)
11590000       DO 20 K = 3,MC-1
11591000       DO 22 J = 1,K-1
11591500    22 B(K,MC,I) = B(K,MC,I) - B(K,J,I)*Y(J)
11592000       Y(K) = B(K,MC,I)
11593000    20 CONTINUE
11594000       DO 40 J=1,M
11595000       L = J  + (I-1)*M
11596000       J2 = J*2
11597000       PUN(L) = Y(J2+1) + PUN(L)
11597100       IF (ABS(Y(J2+1)).GT.0.001) CONVRG = .FALSE.
11598000       SUN(L) = Y(J2+2) + SUN(L)
11598010       IF (SUN(L).LT.0.0) SUN(L) = 0.0
11598020       IF (SUN(L).GT.1.0) SUN(L) = 1.0
11598100       IF (ABS(Y(J2+2)).GT.0.0001) CONVRG = .FALSE.
11599000    40 CONTINUE
11600000       RETURN
11601000       END
$
```

```
C
C**********************************************************************
C
C                    RMTBAL -- MATERIAL BALANCE CALCULATION
C
C**********************************************************************
C
      SUBROUTINE RMTBAL(UMBT,UIPF,UPUF,UPUFN,QU,UIPM,UPUM,UPUMN,CUMU,
     $            CUMO,OMBT,OIPF,UPOF,UPOFN,QO,OIPM,UPOM,UPOMN,M,N,DT)
      DIMENSION QO(1),QU(1),UPUF(1),UPUFN(1),UPOF(1),UPOFN(1)
      DIMENSION UPUM(1),UPUMN(1),UPOM(1),UPOMN(1),CUMU(1),CUMO(1)
C
      UIPFN = UIPF
      UIPF = 0
      OIPFN = OIPF
      OIPF = 0
      UIPMN = UIPM
      UIPM = 0
      OIPMN = OIPM
      OIPM = 0
      QONET = 0
      QUNET = 0
      DO 10 I=1,N
      UIPF = UIPF + UPUF(I)
      OIPF = OIPF + UPOF(I)
      CUMU(I) = CUMU(I) + QU(I)*DT
      CUMO(I) = CUMO(I) + QO(I)*DT
      QUNET = QUNET + QU(I)*DT
      QONET = QONET + QO(I)*DT
   10 CONTINUE
      DO 20 L=1,M*N
      UIPM = UIPM + UPUM(L)
      OIPM = OIPM + UPOM(L)
   20 CONTINUE
      IF (QUNET.NE.0) GO TO 30
      IF (UIPFN+UIPMN.EQ.0) GO TO 32
      UMBT = (UIPF-UIPFN+UIPM-UIPMN) / (UIPFN + UIPMN)
      GO TO 32
   30 UMBT = (UIPF-UIPFN+UIPM-UIPMN-QUNET) / QUNET
   32 IF(QONET.NE.0) GO TO 35
      IF (OPIFN+OPIMN.EQ.0) GO TO 40
      OMBT = (OIPF-OIPFN+OIPM-OIPMN) / (OIPFN + OIPMN)
      GO TO 40
   35 OMBT = (OIPF-OIPFN+OIPM-OIPMN-QONET) / QONET
   40 CONTINUE
      RETURN
      END
$
```

```
11643000 C
11644000 C*************************************************************************
11645000 C
11646000 C                    RITED -- OUTPUT SUBROUTINE
11647000 C
11648000 C*************************************************************************
11649000 C
11650000       SUBROUTINE RITED(X,MY,MX)
11651000       DIMENSION X(1)
11652000
11653000       L1 = 0
11654000       L2 = 0
11655000       L3 = 0
11656000       MAXLP = MX / 8 + 1
11657000       DO 50 M=1,MAXLP
11658000       L1 = L2 + 1
11659000       L2 = MX
11660000       IF (L1+7.LT.MX) L2 = L1+7
11661000       L3 = L2 - L1 + 1
11662000       WRITE(6,100) (I, I=L1,L2)
11663000       DO 20 J=1,MY
11664000    20 WRITE(6,120) J,(X(J+(I-1)*MY), I=L1,L2)
11665000       WRITE(6,130)
11666000    50 CONTINUE
11667000   100 FORMAT(2X,'I ',1X,8(7X,I3,4X))
11668000   120 FORMAT(2X,I2,1X,8(1X,F13.4))
11669000   130 FORMAT(/)
11670000       RETURN
11671000       END
$
```

```
11672000 C
11673000 C*************************************************************************
11674000 C
11675000 C                    RDATA -- PRINTS DEBUG DATA
11676000 C
11677000 C*************************************************************************
11678000 C
11679000       SUBROUTINE RDATA
11680000       COMMON /CONS/CW,CO,MUELL,DT,IMAX,NMB,UIO,UIW,RHOUB,RHOOB,CF,CM
11681000       COMMON /FRAC/BUF,BOF,DF,PUF,POF,PHIBF,PHIF,RKOF,RKUF,RKUPF,PCPF,
11682000      $             RKOPF,TCF,TOF,TUF,SUF,UR,UUF,UOF,RHOUF,RHOOF
11685000       COMMON /UELL/ PI,PBH,PIW,PIO,UW,U1,U2,UD,PWN
11686000       COMMON /MATRIX/BUM,BOM,DM,PUM,POM,PHIBM,PHIM,RKOM,RKUM,RKUPM,
11687000      $             RKOPM,TCM,TOM,TUM,SUM,UUM,UOM,RHOUM,RHOOM,PCPM,
11688000      $             TMF,TOMF,TUMF,UUMF,UOMF
11694000       DIMENSION BUF(99),BOF(99),DF(99),PUF(99),POF(99),PCPF(99)
11695000       DIMENSION PHIBF(99),PHIF(99),RKOF(99),RKUF(99)
11696000       DIMENSION RKUPF(99),RKOPF(99),TCF(99),TOF(99),TUF(99)
11697000       DIMENSION SOF(99),SUF(99),UR(99),UUF(99),UOF(99),RHOUF(99)
11700000       DIMENSION RHOOF(99),UW(99),U1(99),U2(99),UD(99)
11701000       DIMENSION UPWFN(99),UPOFN(99)
11702000       DIMENSION PI(99),PBH(99),PIW(99),PIO(99),RWL(99),PWN(99)
11703000       DIMENSION BUM(99),BOM(99),DM(99),PUM(99),POM(99),RHOOM(99)
11704000       DIMENSION PHIBM(99),PHIM(99),RKOM(99),RKUM(99),PCPM(99)
11705000       DIMENSION RKUPM(99),RKOPM(99),TCM(99),TOM(99),TUM(99)
11706000       DIMENSION SOM(99),SUM(99),UUM(99),UOM(99),RHOUM(99)
11707000       DIMENSION TMF(99),TUMF(99),TOMF(99),UUMF(99),UOMF(99)
11713000
11714000       WRITE(6,527)
11715000  527  FORMAT(///40X,'RKOF ',/,120('='))
11716000       CALL RITED(RKOF,1,IMAX)
11717000       WRITE(6,537)
11718000  537  FORMAT(///40X,'RKUF ',/,120('='))
11718500       CALL RITED(RKUF,1,IMAX)
11719100       WRITE(6,550)
11719200  550  FORMAT(///40X,'RKOPF ',/,120('='))
11719300       CALL RITED(RKOPF,NMB,IMAX)
11719400       WRITE(6,555)
11719500  555  FORMAT(///40X,'RKUPF ',/,120('='))
11719600       CALL RITED(RKUPF,NMB,IMAX)
11720000       WRITE(6,547)
11721000  547  FORMAT(///40X,'PHIF ',/,120('='))
11722000       CALL RITED(PHIF,1,IMAX)
11723000       WRITE(6,557)
11724000  557  FORMAT(///40X,'RHOUF ',/,120('='))
11725000       CALL RITED(RHOUF,1,IMAX)
11726000       WRITE(6,627)
11727000  627  FORMAT(///40X,'RHOOF ',/,120('='))
11728000       CALL RITED(RHOOF,1,IMAX)
11729000       WRITE(6,637)
11730000  637  FORMAT(///40X,'UOF ',/,120('='))
11731000       CALL RITED(UOF,1,IMAX)
11732000       WRITE(6,647)
11733000  647  FORMAT(///40X,'UUF ',/,120('='))
11734000       CALL RITED(UUF,1,IMAX)
11735000       WRITE(6,657)
11736000  657  FORMAT(///40X,'BUF ',/,120('='))
11737000       CALL RITED(BUF,1,IMAX)
11738000       WRITE(6,727)
11739000  727  FORMAT(///40X,'BOF ',/,120('='))
11740000       CALL RITED(BOF,1,IMAX)
11741000       WRITE(6,737)
11742000  737  FORMAT(///40X,'PCPF ',/,120('='))
11743000       CALL RITED(PCPF,1,IMAX)
11744000       WRITE(6,747)
11745000  747  FORMAT(///40X,'UR ',/,120('='))
11746000       CALL RITED(UR,1,IMAX)
11747000       WRITE(6,757)
11748000  757  FORMAT(///40X,'TCF ',/,120('='))
11749000       CALL RITED(TCF,1,IMAX)
11750000       WRITE(6,828)
11751000  828  FORMAT(///40X,'TUF ',/,120('='))
11752000       CALL RITED(TUF,1,IMAX)
11753000       WRITE(6,838)
11754000  838  FORMAT(///40X,'TOF ',/,120('='))
11755000       CALL RITED(TOF,1,IMAX)
$
```

```
11756000       WRITE(6,848)
11757000  848  FORMAT(///40X,'TOMF ',/,120('*'))
11758000       CALL RITED(TOMF,NMB,IMAX)
11759000       WRITE(6,858)
11760000  858  FORMAT(///40X,'TUMF ',/,120('*'))
11761000       CALL RITED(TUMF,NMB,IMAX)
11762000       WRITE(6,929)
11763000  929  FORMAT(///40X,'TMF ',/,120('*'))
11764000       CALL RITED(TMF,NMB,IMAX)
11765000       WRITE(6,939)
11766000  939  FORMAT(///40X,'RKOM ',/,120('*'))
11767000       CALL RITED(RKOM,NMB,IMAX)
11768000       WRITE(6,949)
11769000  949  FORMAT(///40X,'RKUM ',/,120('*'))
11770000       CALL RITED(RKUM,NMB,IMAX)
11770100       WRITE(6,950)
11770200  950  FORMAT(///40X,'RKOPM ',/,120('*'))
11770300       CALL RITED(RKOPM,NMB,IMAX)
11770400       WRITE(6,955)
11770500  955  FORMAT(///40X,'RKUPM ',/,120('*'))
11770600       CALL RITED(RKUPM,NMB,IMAX)
11771000       WRITE(6,959)
11772000  959  FORMAT(///40X,'RHOOM ',/,120('*'))
11773000       CALL RITED(RHOOM,NMB,IMAX)
11774000       WRITE(6,1029)
11775000 1029  FORMAT(///40X,'RHOUM ',/,120('*'))
11776000       CALL RITED(RHOUM,NMB,IMAX)
11777000       WRITE(6,1039)
11778000 1039  FORMAT(///40X,'UUM ',/,120('*'))
11779000       CALL RITED(UUM,NMB,IMAX)
11780000       WRITE(6,1049)
11781000 1049  FORMAT(///40X,'UOM ',/,120('*'))
11782000       CALL RITED(UOM,NMB,IMAX)
11783000       WRITE(6,1059)
11784000 1059  FORMAT(///40X,'PHIM ',/,120('*'))
11785000       CALL RITED(PHIM,NMB,IMAX)
11786000       WRITE(6,1129)
11787000 1129  FORMAT(///40X,'UUMF ',/,120('*'))
11788000       CALL RITED(UUMF,NMB,IMAX)
11789000       WRITE(6,1139)
11790000 1139  FORMAT(///40X,'UOMF ',/,120('*'))
11791000       CALL RITED(UOMF,NMB,IMAX)
11792000       WRITE(6,1149)
11793000 1149  FORMAT(///40X,'PCPM ',/,120('*'))
11794000       CALL RITED(PCPM,NMB,IMAX)
11795000       WRITE(6,1159)
11796000 1159  FORMAT(///40X,'TUM ',/,120('*'))
11797000       CALL RITED(TUM,NMB,IMAX)
11798000       WRITE(6,1229)
11799000 1229  FORMAT(///40X,'BUM ',/,120('*'))
11800000       CALL RITED(BUM,NMB,IMAX)
11801000       WRITE(6,1239)
11802000 1239  FORMAT(///40X,'BOM ',/,120('*'))
11803000       CALL RITED(BOM,NMB,IMAX)
11804000       WRITE(6,1249)
11805000 1249  FORMAT(///40X,'TOM ',/,120('*'))
11806000       CALL RITED(TOM,NMB,IMAX)
11807000       WRITE(6,1259)
11808000 1259  FORMAT(///40X,'TCM ',/,120('*'))
11809000       CALL RITED(TCM,NMB,IMAX)
11811100       WRITE(6,1269)
11811200 1269  FORMAT(///40X,'UU ',/,130('*'))
11811300       CALL RITED(UU,1,IMAX)
11812000       WRITE(6,1339)
11813000 1339  FORMAT(///40X,'PI ',/,130('*'))
11814000       CALL RITED(PI,1,IMAX)
11815000       WRITE(6,1349)
11816000 1349  FORMAT(///40X,'PIU ',/,130('*'))
11817000       CALL RITED(PIU,1,IMAX)
11818000       WRITE(6,1359)
11819000 1359  FORMAT(///40X,'PIO ',/,130('*'))
11820000       CALL RITED(PIO,1,IMAX)
11821000       RETURN
11822000       END
$
```

```
11823000 C
11824000 C*************************************************************************
11825000 C
11826000 C                    DEBUG -- PRINTS DEBUG DATA
11827000 C
11828000 C*************************************************************************
11829000 C
11830000       SUBROUTINE DEBUG
11831000       COMMON /CONS/CW,CO,NUELL,DT,IMAX,NMB,VIO,VIW,RHOWB,RHOOB,CF,CM
11834000       COMMON /COEFF/A11,A12,A21,A22,B11,B12,B21,B22,C11,C12,C21,C22
11835000       COMMON /RHS/ RFW,RFO,RMW,RMO,RWL
11836000       COMMON /WELL/ PI,PBH,PIW,PIO,UW,U1,U2,UD,PWN
11840000       COMMON /COEF2/AM11,AM12,AM21,AM22,BM11,BM12,BM21,BM22,CM11,CM12,
11841000      $              CM21,CM22
11841100       COMMON /COEF3/ B
11841200       DIMENSION B(25,25,50)
11842000       DIMENSION AM11(99),AM12(99),AM21(99),AM22(99)
11843000       DIMENSION BM11(99),BM12(99),BM21(99),BM22(99),CM11(99)
11844000       DIMENSION CM12(99),CM21(99),CM22(99)
11849000       DIMENSION A11(99),A12(99),A21(99),A22(99)
11850000       DIMENSION B11(99),B12(99),B21(99),B22(99),C11(99)
11851000       DIMENSION C12(99),C21(99),C22(99),UW(99),U1(99),U2(99),UD(99)
11852000       DIMENSION RFW(99),RMW(99),RFO(99),RMO(99)
11853000       DIMENSION PI(99),PBH(99),PIW(99),PIO(99),RWL(99),PWN(99)
11864000
11865000       WRITE(6,527)
11866000  527  FORMAT(///40X,'A11 ',/,120('*'))
11867000       CALL RITED(A11,1,IMAX)
11868000       WRITE(6,537)
11869000  537  FORMAT(///40X,'A12 ',/,120('*'))
11870000       CALL RITED(A12,1,IMAX)
11871000       WRITE(6,547)
11872000  547  FORMAT(///40X,'A21 ',/,120('*'))
11873000       CALL RITED(A21,1,IMAX)
11874000       WRITE(6,557)
11875000  557  FORMAT(///40X,'A22 ',/,120('*'))
11876000       CALL RITED(A22,1,IMAX)
11877000       WRITE(6,627)
11878000  627  FORMAT(///40X,'B11 ',/,120('*'))
11879000       CALL RITED(B11,1,IMAX)
11880000       WRITE(6,637)
11881000  637  FORMAT(///40X,'B12 ',/,120('*'))
11882000       CALL RITED(B12,1,IMAX)
11883000       WRITE(6,647)
11884000  647  FORMAT(///40X,'B21 ',/,120('*'))
11885000       CALL RITED(B21,1,IMAX)
11886000       WRITE(6,657)
11887000  657  FORMAT(///40X,'B22 ',/,120('*'))
11888000       CALL RITED(B22,1,IMAX)
11889000       WRITE(6,727)
11890000  727  FORMAT(///40X,'C11 ',/,120('*'))
11891000       CALL RITED(C11,1,IMAX)
11892000       WRITE(6,737)
11893000  737  FORMAT(///40X,'C12 ',/,120('*'))
11894000       CALL RITED(C12,1,IMAX)
11895000       WRITE(6,747)
11896000  747  FORMAT(///40X,'C21 ',/,120('*'))
11897000       CALL RITED(C21,1,IMAX)
11898000       WRITE(6,757)
11899000  757  FORMAT(///40X,'C22 ',/,120('*'))
11900000       CALL RITED(C22,1,IMAX)
11913000       WRITE(6,929)
11914000  929  FORMAT(///40X,'BM11 ',/,120('*'))
11915000       CALL RITED(BM11,NMB,IMAX)
11916000       WRITE(6,939)
11917000  939  FORMAT(///40X,'BM12 ',/,120('*'))
11918000       CALL RITED(BM12,NMB,IMAX)
11919000       WRITE(6,949)
11920000  949  FORMAT(///40X,'BM21 ',/,120('*'))
11921000       CALL RITED(BM21,NMB,IMAX)
11922000       WRITE(6,959)
11923000  959  FORMAT(///40X,'BM22 ',/,120('*'))
11924000       CALL RITED(BM22,NMB,IMAX)
11925000       WRITE(6,1029)
11926000 1029  FORMAT(///40X,'AM11 ',/,120('*'))
11927000       CALL RITED(AM11,NMB,IMAX)
$
```

```
11928000       WRITE(6,1039)
11929000 1039 FORMAT(///40X,'AM12 ',/,120('*'))
11930000       CALL RITED(AM12,NMB,IMAX)
11931000       WRITE(6,1049)
11932000 1049 FORMAT(///40X,'AM21 ',/,120('*'))
11933000       CALL RITED(AM21,NMB,IMAX)
11934000       WRITE(6,1059)
11935000 1059 FORMAT(///40X,'AM22 ',/,120('*'))
11936000       CALL RITED(AM22,NMB,IMAX)
11937000       WRITE(6,1129)
11938000 1129 FORMAT(///40X,'CM11 ',/,120('*'))
11939000       CALL RITED(CM11,NMB,IMAX)
11940000       WRITE(6,1139)
11941000 1139 FORMAT(///40X,'CM12 ',/,120('*'))
11942000       CALL RITED(CM12,NMB,IMAX)
11943000       WRITE(6,1149)
11944000 1149 FORMAT(///40X,'CM21 ',/,120('*'))
11945000       CALL RITED(CM21,NMB,IMAX)
11946000       WRITE(6,1159)
11947000 1159 FORMAT(///40X,'CM22 ',/,120('*'))
11948000       CALL RITED(CM22,NMB,IMAX)
11949000       WRITE(6,1229)
11950000 1229 FORMAT(///40X,'RFW ',/,120('*'))
11951000       CALL RITED(RFW,1,IMAX)
11952000       WRITE(6,1239)
11953000 1239 FORMAT(///40X,'RFO ',/,120('*'))
11954000       CALL RITED(RFO,1,IMAX)
11955000       WRITE(6,1249)
11956000 1249 FORMAT(///40X,'RMW ',/,120('*'))
11957000       CALL RITED(RMW,NMB,IMAX)
11958000       WRITE(6,1259)
11959000 1259 FORMAT(///40X,'RMO ',/,120('*'))
11960000       CALL RITED(RMO,NMB,IMAX)
11960100       WRITE(6,1269)
11960200 1269 FORMAT(///40X,'RWL ',/,120('*'))
11960300       CALL RITED(RWL,1,IMAX)
11961000       WRITE(6,1339)
11962000 1339 FORMAT(///40X,'W1 ',/,130('*'))
11963000       CALL RITED(W1,1,IMAX)
11964000       WRITE(6,1349)
11965000 1349 FORMAT(///40X,'W2 ',/,130('*'))
11966000       CALL RITED(W2,1,IMAX)
11967000       WRITE(6,1359)
11968000 1359 FORMAT(///40X,'WD ',/,130('*'))
11969000       CALL RITED(WD,1,IMAX)
11969100       WRITE(6,1369)
11969300 1369 FORMAT(///40X,'B ',/,130('*'))
11969400       DO 1400 I=1,IMAX
11969500 1400 CALL RITED(B(1,1,I),20,2*NMB+3)
11970000       RETURN
11971000       END
$
```