1999

Petroleum Reservoir Simulation Using 3-D Finite Element Method With Parallel Implementation.

Husam M. Yaghi
Louisiana State University and Agricultural & Mechanical College

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

PETROLEUM RESERVOIR SIMULATION USING 3-D FINITE ELEMENT METHOD WITH PARALLEL IMPLEMENTATION

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Husam M. Yaghi
B.S., Southern University, 1984
M.S., Southern University, 1986
August 1999

Dedicated to my mother and father.

ACKNOWLEDGEMENTS

To Dr. John Tyler, my research major advisor and mentor, I say thank you. I hope you know I am much more grateful than words can express for all you have done. Please accept my warmest sincere thanks. Also thanks to the rest of my committee members Dr. Ted Bourgoyne, Dr. S. Sitharama Iyengar, Dr. Bert Boyce, and Dr. Robert O'Connell.

I am very grateful to my two times alma mater Southern University for its financial support and the exceptional opportunities that I was given. Special thanks to Dr. James Anderson for being my best friend and for all his assistance, also to Dr. James Cross, Dr. George Whitfield, Dr. Sane Yagi, Mr. Alonzo Johnson, and Mr. Rahman Tashakkori.

Finally, but certainly no less important, I dedicate this work to my mother and father. I express sincere gratitude and love to my brothers, sisters, and my wife for their inspiration and support. Special thanks to my loving princesses Angel, Amber, and Amira, for their patience as I devoted my life for this venture.

In conclusion, I pray the almighty for his great blessings.

TABLE OF CONTENTS

ABSTRACT

Modeling fluid flow around wellbores with conventional reservoir simulators is inaccurate because radial flow occurs in the vicinity of the wellbore and these simulators use cartesian coordinates. In this research, we present a more accurate wellbore simulation by incorporating the finite element method (FEM) to simulate the radial flow in the vicinity of the wellbore and interfacing this finite element wellbore model with an existing finite difference method (FDM) reservoir simulator. Although this technique was developed for a vertical well, it could also be used to accurately model a horizontal wellbore. This "hybrid" solution is for three dimensional - triphasic fluid flow and allows a more rigorous treatment of the near-well flow. The reservoir region, where flow geometry is linear, is simulated with the cartesian grid using finite differences.

The reservoir simulator used for this research was the US Department of Energy's Black Oil Applied Simulation Tool (BOAST II). Two problems furnished by the Department of Energy were used to test the effectiveness of our solution. The first was a single stratum three phase system. The second was a three strata three phase gas injection problem.

Finally, our stand alone model could actually be interfaced with almost any other finite difference fluid flow simulator; whether it is for petroleum reservoirs, underground water, or hazardous waste management.

CHAPTER 1. INTRODUCTION

Oil and natural gas are two of the world's most important natural resources. They are building materials of modern life. Together, oil and natural gas are called petroleum and remain in a reservoir until produced. In general, a reservoir is both the reservoir rock and its fluid content [1]. The life of a reservoir can be classified into primary or secondary recovery phases. In the primary recovery phase of a reservoir, oil is obtained by natural drive mechanisms. In the secondary phase a recovery process can be initiated to maintain the pressure in a reservoir by injecting water or gas. The later type is the focus of this research. Actually, there can be recovery processes after the secondary process but the mechanism for these is beyond the scope of this research.

Reservoir simulation has a key role in the development and management of petroleum resources. One objective of this research is to provide better tools for the understanding of the complex physical fluid flow processes that occur around a wellbore in a reservoir. A better understanding of this process could lead to increased recovery and reduced expenses. Classical reservoir simulation deals with the fluids on a gross average basis and does not account adequately for the flow pattern variations in the reservoir and fluid changes in the wellbore caused by pressure and time [1]. While many advances have been made in reservoir engineering and well drilling, the modeling and technology have lagged. The development of an accurate simulation tool that can be used to study flow in the vicinity of the wellbore is needed.

The accurate computer simulation of the multiphase fluid flow processes continues to be a difficult endeavor. Such problems feature near-discontinuities in the solution which are not sufficiently resolved by standard domain discretization procedures without extensive grid refinement. Meanwhile, the problem domains are often large-scale, irregularly shaped, and exhibit heterogeneous characteristics. Methods which are developed to simulate these processes are often useful in a number of other disciplines in which the differential equations are analogous (e.g. the flow of heat). Such processes as groundwater flow and hazardous waste migrations are closely related to reservoir engineering problems, and improvements in the modeling of one type of fluid flow problem may be utilized in obtaining an improved solution to another.

The direction of flow and rate of flow depend on the physical features of the flowing medium such as its viscosity, phase behavior, and the nature of the reservoir rock in which it is flowing such as permeability, pore geometry, etc. Viscosity is a fluid property responsible for the frictional drag or shear resistance which develops when one layer of a fluid slides over another. Permeability is a property that measures the ability of the reservoir rock to transport fluids through itself. Permeability is independent of the nature of the fluid and is determined solely by the structure of the porous media. Porosity measures the reservoir rock's ability to store petroleum, which may be defined as one minus the fraction of the bulk volume of the rock comprised of solid matter. Viscosity, permeability, and porosity are expressed as percentages.

New technologies have changed the way we search for the petroleum. We study the ground beneath the surface using technology that gives us a three-dimensional view of what that ground is like. All of these high-tech tools help pinpoint where the oil and natural gas are-and where they are not-so we drill fewer wells.

Engineers are constantly in search of tools that could help in enhancing the recovery of petroleum. Reservoir simulation has a key role in the development and management of this activity. Reservoir simulation is a process for predicting the behavior of a real reservoir from the analysis of a model of that reservoir. The model could be a scaled physical model examined in a laboratory, or mathematical. The mathematical model developed in this research is a set of nonlinear partial differential equations that describe the activities occurring within the reservoir. These activities are the simultaneous flow of three phases (water, oil, and gas) along with the mass transfer between these phases.

The mathematical model accounts for various factors affecting the behavior of the fluids. It takes into account gravity, pressure, heterogeneity, and geometry. The mathematical model begins by combining Darcy's flow for each phase with a simple differential material balance for each phase. Henry Darcy originally designed a flow tube to determine the most efficient means of filtrating the municipal sewage water in Dijon, France, in 1865. He found that the rate of water flow through a porous bed of a "given nature" is proportional to the pressure and to the cross-sectional area normal to the direction of flow and inversely proportional to the length of the flow path. He also determined that the quantity of flow is related to the

nature of the porous medium. Darcy's Law was initially developed for one-dimensional flow through a porous media. However, combining this relationship with calculus, this law has been extended to flow in two or three dimensions .

1.1 Statement of the Problem

The purpose of this research is to develop a more accurate numerical model of three dimensional three-phase fluid flow in a porous media; specialized for the treatment of the wellbore vicinity where majority of fluid activity occurs. Most reservoir simulations models available today, obtain solutions to fluid flow equations that are usually nonlinear partial differential equ ations by replacing derivatives with finite-difference approximations [4,6]. The use of these approximations introduces an error known as truncation error. For many problems the error is small and the approximate solutions of the subsequent finite difference equations are sufficiently accurate. However, truncation errors can cause significant solution inaccuracies for certain types of problems in which viscous forces are much larger than capillary forces.

Available public domain reservoir simulators represent the behavior of fluid flow in the wellbore vicinity with Cartesian coordinates and fail to provide robust and correct answers for the large pressure and saturation changes in this vicinity. One reason is that these reservoir simulators often make simplifications of the equations that are not physically realistic. Another reason is that they use a finite difference method to represent the wellbore vicinity where high mobility and large changes in the saturation of the fluids and pressure occur.

An important step for this simulation of physical phenomena is the transformation of the underlying differential equations into a finite discritized space. In the considered domain, the resulting partial differential equations are approximated using numerical methods on finite discrete intervals.

In spite of the enormous commercial potential and interest in this field, an extensive literature survey has shown that commercial modeling of the "wellbore" region has been extremely limited. Commercial simulators are proprietary in order for the companies to protect their investment from competitors. Therefore, commercial tools are not available to us in a form that can be modified for use with this research.

In order to develop a reservoir simulation tool, understanding of the physical problem is required. Mistakenly, it is thought that petroleum is found underground in a pool from which production occurs. On the contrary, oil and natural gas are trapped inside tiny rock holes or pores of rock which complicates the task of simulating the process. In the secondary recovery phase, petroleum is forced out by injecting another fluid through another well (injection well), Figure 1.1.

To produce petroleum from a production well, engineers generally drill in an approximate range from 1,000 to 20,000 feet deep for vertical wells and the same depth plus a horizontal length for horizontal wells. The well radius usually ranges between 4 to 6 inches at the reservoir depth. There are other issues associate with drilling wells that affect reservoir simulation. However drilling engineering is beyond the scope of this research.

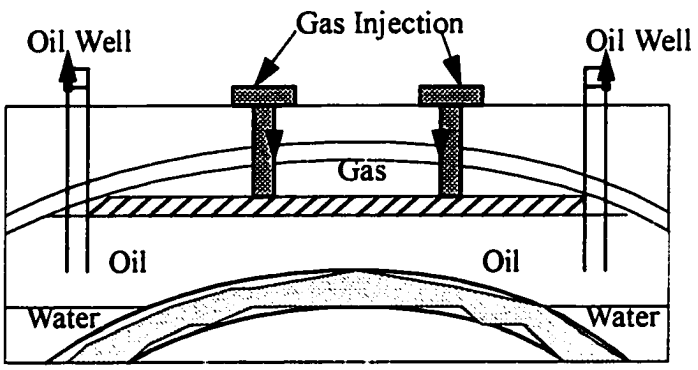


Figure 1.1: Oil Reservoir (not to scale)

A readily available public reservoir simulator BOAST II (Black Oil Applied Simulation Tool) from the United States Department of Energy could be modified to simulate the conditions encountered in the wellblock region of interest. Hence, our reservoir wellblock region model uses BOAST II as the basic building block because it was the best working simulator that supplied the source code to which our model could be interfaced.

BOAST II has several drawbacks, some of which are:

1. It is based on the finite difference model, which divides the entire region of interest into equally spaced blocks. This arrangement gives no special consideration to areas of high activity. For the production well-block, we use finite elements to create more points (sub-blocks) closer to the wellbore. Our results show that most activity takes place within the

first 20 feet or so from the well in a wellblock. Finite elements also better model radial flow in this region.

2. It uses one average pressure value for each reservoir block. To make modeling of this highly active block more accurate, we developed a finite element model to further examine this wellblock. Once we are finished producing a detailed map of pressure and saturation activities at various points in that block, all these values are averaged and inserted back into BOAST as an improved value for the block.

In building our simulator, we follow four basic major steps. First, a physical model of the flow process is developed incorporating as much physics as is deemed necessary to describe the essential phenomena. Second, a mathematical formulation of the physical model is developed, usually involving coupled systems of nonlinear partial differential equations. Third, once the properties of the mathematical model, such as existence, uniqueness, and regularity of the solution, are sufficiently well understood and the properties seem compatible with the physical model, discretized numerical approximations of the mathematical equations are produced [9]. Finally, a computer wellbore model is developed, executed, and results obtained which are compared with actual observations of this physical process to demonstrate its validity. The actual observations for testing where furnished by the United States Department of Energy - National Institute of Petroleum and Energy Research.

An important aspect of the wellbore research problem is that the geometry of flow is radial [10], which must be taken into account to accurately simulate

multiphase flow in the neighborhood of the wellbore. The simplest case is that of perfect radial flow of homogeneous fluid into a well. Such flow is obtained if the well completely penetrates the rock stratum and the distant fluid acts uniformly in all directions radiating from the axis of the well bore. From a practical point of view, this perfectly radial flow is too idealized because it implies an exactly uniform pressure imposed on a circular boundary.

It is anticipated that even cases with only a single well will, in general, have nonuniform pressure distribution over their external boundaries and the boundaries themselves over which the pressure distributions are pre-assigned and known will be other than circular in shape. In all cases, the flow into the wells will be unsymmetrical and the pressure distributions on the external boundary will be nonuniform.

If a localized region around the wellbore can be considered homogeneous, the flow is radial, and the fluid flow equations for a single phase can be formulated and solved analytically in cylindrical coordinates. In this case, it is known that a "radial" grid system provides much better results than the use of a rectangular grid system. As the distance from the wellbore increases, the flow becomes more linear. This means that a rectangular grid system can be applied at some distance from the wellbore with confidence to discretize the fluid flow equations and a Cartesian coordinate system would be accurate. Therefore it is necessary for reservoir simulators to properly represent the fluid flow in the regions of interest and then couple the various systems together.

Since the regions in the vicinity of the wellbore often have significant pressure and saturation changes, an implicit treatment of the transmissibility in this region is needed. On the other hand, in reservoir zones distant from the wellbore, the transmissibility may be treated explicitly. We employ a finite element method (FEM) for the wellbore region [93] and the finite difference method (FDM) from BOAST II for the other regions of the reservoir. The FEM is necessary for the wellbore region where large condition changes are exhibited. The FDM is adequate for other regions of the reservoir where more uniform conditions are likely to be found. The simulator then has to couple both regions together for the exchange of data. As part of this research a FEM/FDM grid interface is developed and employed as illustrated in Figure 4.3. Boundary conditions between these two methods are also developed. All fluids are treated as compressible and transient, unlike a common practice of assuming the opposite to reduce complexity.

To accomplish the objectives of this research, the following aspects of reservoir simulation were addresses:

- an improved representation for the wellbore vicinity;
- more accurate treatment of the wellbore/reservoir interaction;
- a coupling of the FEM wellbore region model with the FDM reservoir model;
- utilization of parallel computing.

Our wellbore model can be used with other reservoir simulators provided the interface is done at the source code level. The model works independently except

for the initial input of data. However, if the input data is faulty, the wellbore model will produce faulty results. The model can predict the wellbore behavior and can produce detailed history information about the fluid pressures and saturations at various locations within a wellblock; these were not available from BOAST II.

In addition to the output files that our wellbore model produces, we also developed and added a web-based multimedia visualization tool [90,91]. This tool reads the simulation output files, generates a web-based table showing the results, and then passes that data to a Java appellate for colored and animated visualization. This prototype model also suggested a bilingual graphical user interface and was presented at a visualization conference in Toronto, Canada [90].

CHAPTER 2. LITERATURE REVIEW

This chapter reviews petroleum engineering and computer science literature pertinent to this research. The topics covered include current simulations for the wellbore vicinity, examples of two types of current reservoir simulators, the development of domain decomposition in reservoir simulation, the coupling of FDM/FEM, some aspects of FEM, and parallel reservoir simulations.

2.1 Current Reservoir Simulations

Most of the current mathematical models for reservoir simulation are presented in [1,4,5,6,7,8,9]. There are many other sources and the number of publications about reservoir simulation indicates that this is a very mature field. However, very few of these models utilize the finite element method and even fewer couple the finite difference method and finite element method to produce a more accurate simulation. The finite difference method is often not very accurate in the vicinity of wellbores.

Results for a finite element method (FEM) simulation of a two-dimensional and two-phase (water and oil) reservoir simulation have been published [9]. This simulation was to model an injection and a production well. The injected fluid was water. Another FEM publication [10], simulated a two-phase two-dimensional coning problem. Both of these finite element simulation papers stated that the results were more accurate than the same simulation using finite difference methods. However, there was no published model using finite elements with 3-D and three-phase.

2.2 Well Models

Reservoir simulators use both numerical and analytical models to determine the flow within the wellbore vicinity. A well (wellbore) model should account for the "geometry" of flow as well as reservoir properties in the vicinity of the wellbore. Since the pressure calculated for a grid block that contains a well can differ from the actual wellbore bottom-hole flowing pressure, an auxiliary formula is required to resolve these differences in pressure. Assuming the flow around the wellbore is radial, single phase, and one-dimensional, an analytical solution exists for the differential equation in cylindrical coordinates for this problem [11,12]. These analytical solutions are often used as the formulas between the wellbore bottom-hole flowing pressure and the simulator calculated pressure.

2.2.1 Peaceman's Well Model

This section covers the more popular well models used in current reservoir simulations.

In the Peaceman well model [4], the pressure of the block containing the well is not necessarily equal to the average pressure of the block. In this model the pressure calculated for the well block is the same as the steady state flowing pressure at an equivalent radius, r_o. This equivalent radius can be used to relate the flowing bottom-hole wellbore pressure, P_{wf} to M_{cp}; the mass flow rate M_{cp} of component c in phase p, with the average wellblock pressure P_o, as shown in the following equation:

$$M_{cp} = \left(\frac{k_{rp}}{\mu_p}\right) x_{cp} \left[\frac{2\pi \bar{k} \Delta z}{\ln\left(\frac{r_o}{r_w}\right) + S}\right] (P_o - P_{wf}) \qquad (2.1)$$

where ρ_p is the density of phase p, r_o is the equivalent radius, r_w is the wellbore radius, x_{cp} is the mass fraction of component c in phase p, S is the skin factor, $\Delta z = z_{j+1} - z_{j-1}$ is the Cartesian coordinate difference, k_x and k_y are the permeabilities in the x and y directions respectively, $\bar{k} = \sqrt{k_x k_y}$, and k_{rp} is the absolute permeability in the radial direction of phase p.

Peaceman [4] showed how to calculate the approximate equivalent radius for the well using the equation for the pressure drop between injection and production wells. He also extended the interpretation of well block pressure to "rectangular" grid blocks. The equivalent radius was determined as a function of the aspect ratio $a = \frac{\Delta x}{\Delta y}$ of the grid block. A frequently used equation for the well block radius was:

$$r_o = 0.14\sqrt{\Delta x^2 + \Delta y^2} \qquad (2.1)$$

Babu et al. [13] developed a general analytical equation for calculating the equivalent well radius r_o. Their equation can be used for both vertical and horizontal wells and for any well location. For wells that are essentially centered inside the drainage area, they proposed the use of the following simplified equation:

$$r_o = 0.14(k_x k_y)^{1/4}\left[\frac{(Dx)^2}{k_x} + \frac{(Dy)^2}{k_y}\right]^{1/2} ABC \ , \qquad (2.2)$$

where:

$$ABC = \left[\frac{1 + \exp\left(2.215 - \frac{3.88 n_x n_y}{b}\right)}{1 + 0.533\left(\frac{b}{n_y}\right)}\right] , \qquad (2.3)$$

n_x and n_y are the number of grid blocks in the x and y directions respectively, Dx and Dy are mesh dimensions in feet, k_x and k_y are the permeability, and coefficient b is:

$$b = \frac{Dx}{Dy}\sqrt{\frac{k_y}{k_x}} \ . \qquad (2.4)$$

This equation is valid for uniform grids.

2.2.1 Transmissibility in Vicinity of Wellbore

Blair and Weinaug [14], MacDonald and Coats [15], and Behie [16] claim that pressure and saturation dependence on time of the first term of (2.2) and (2.3) must be consistent with the phase transmissibility coefficient of the grid blocks, otherwise convergence problems due to saturation oscillations can occur [17]. This is especially true for problems involving high capillary forces and/or small well blocks. Therefore, the well transmissibility coefficient should be treated implicitly. Others [18,19,20] state that the use of a fully implicit well can also be used with

IMPES models to obtain stable solutions with no increase in computing time per time step. In this research, the Implicit Pressure/ Explicit Saturation (IMPES) method for the block containing a well was used to obtain a solution.

2.3 Black-Oil Reservoir Model

A Black-Oil model treats water, oil and gas as three separate liquid phases with the gas having limited solubility in both the oil and water phases. The water and oil phases are assumed to be immiscible with no mass transfer between them. The oil phase at reservoir conditions is a mixture of stock tank oil and dissolved gas and the water phase is a mixture of stock tank water and dissolved gas. This model [1] ignores both reservoir temperature change and mass diffusion in fluid flow. Several previous studies applied finite elements to reservoir simulation, however, the black-oil and truly three-dimensional case has been avoided because of its complexity [21]. Our research uses the Black-Oil model because the available reservoir simulator BOAST II used this model. A Black-Oil reservoir simulator is most often used for primary recovery and waterflooding simulations.

There are other types of reservoir simulators that are used for enhanced oil recovery (EOR), that are discussed in details in the literature and are out of scope for this research [22,23]. These are compositional, thermal, and chemical flooding simulators. The wellbore vicinity model developed in this research will work in these types of simulators with the appropriate modifications. The techniques used in this research can serve as an approach to improve the models of the wellbore vicinity.

2.4 Domain Decomposition

The decomposition of domains has been used to numerically solve problems for over one hundred years. The main idea behind domain decomposition is based on a philosophy of divide-and-conquer. The domain of interest is divided into smaller subdomains, the problem is solved independently in the subdomains, then the solutions of the subdomains are combined to approximate the solution to the original problem. In general, the domain decomposition process has to be repeated until some convergence criterion using these subdomains is satisfied.

Bramble et al. [24] solved a system of equations resulting from the discretization of symmetric elliptic boundary value problems via the finite-element method. They developed a preconditioned algorithm for use with domain decomposition.

Apart from being able to solve these subproblems independently on different processors, these methods have other advantages. One advantage is to split a problem with complex geometry into a problem with regularly shaped subdomains. The ability to subdivide the original problem into smaller subdomains permits approximate solutions that could not be obtained otherwise for the highly structured flows in fluid dynamics, and resolves localized phenomena at fluid interfaces of multiphase flow in reservoir simulation [17,25].

There are several domain decomposition methods that have been devel-oped for solving large problems [17]. These methods differ through partitioning of original domains, solutions used with the resultant subdomains, and the approach used for interface problems. The partitioning can have overlapping, no overlapping,

strips, or boxes. The resultant subdomain equations can be approximated either linearly or non-linearly.

2.5 Coupling of FDM and FEM

Most reservoir simulators use either finite difference or finite element methods for the discretization of the governing partial differential equations. Each method has advantages. The finite difference method requires less computing time and storage compared to the other methods. However, the finite difference method's domain (reservoir) boundary representations often require special logic and are generally inaccurate. On the other hand, the finite element method is good at modeling the flow around arbitrarily shaped geometries. The finite element methods usually require more computing time and storage than the finite difference methods for the same problem. The finite difference method and the finite element method each have some properties that are better suited than the other for use in the simulation of fluid flow in complex geometries. There has been very little published [26,27,28] on the use of both FEM and FDM methods together in a model for compressible flow problems.

In recent years, there has been a renewed interest in finite element approximations in reservoir simulations, mainly due to their ability to more accurately approximate complex geometries, discontinuities, solutions for wells [29,30,31,32,33], and tracer injection studies [34]. However, to approach the performance of finite difference simulators requires that special attention be given to all

parts of the finite element solution technique. The most time consuming computation of the finite element method is in the solution of the resultant system of equations.

Nakahashi and Obayashi [28] presented a finite difference / finite element (FDM/FEM) zonal approach to analyze compressible flows in turbine cascades and compressor blade rows. In their approach the regions near turbine cascades or compressor blades used a boundary-fitted grid which is connected by the finite element mesh.

Ikegawa, et. al [27] presented a numerical technique to solve viscous incompressible flow problems which couples the finite element method and the finite difference method. Their computation was based on an overlapping girding system, where finite element meshes are generated around arbitrarily shaped bodies and a finite difference grid is partially superimposed on finite element meshes.

2.6 Parallel Computing in Reservoir Simulation

Architectural advances in the computer industry have resulted in a significant effort in the development of parallel software models for the simulation of reservoirs. For instance, for several years, the idea of using domain decomposition techniques was ignored until the interest in these methods was renewed to partly fulfill the need to develop simulations with parallel computers.

A great deal of work has been done on building efficient parallel reservoir simulators and efficient solvers which reduce both the required time and storage

[25,31,35,36,37,38]. Unfortunately, few of these parallel applications have found their way into a production environment [39]. Because numerous parallel solvers are available in the public domain that can be incorporated into reservoir simulators, we found it of little value to developing another parallel solver was considered of little value to investigate the development of another parallel solver. The need for parallel computation arises from the fact that reservoir simulations require the solution of very large systems of equations and this requires enormous amounts of computation.

CHAPTER 3. FINITE ELEMENT METHOD

The finite element method (FEM) is a computer-aided mathematical technique for obtaining approximate numerical solutions to the abstract equations of calculus that predict the response of physical systems subject to external influences. Such problems arise in many areas of engineering, science, and applied mathematics.

The FEM solves partial differential equations by first discretizing these equations in their space dimensions. The discretization is carried out locally over small regions of simple but arbitrary shape (the finite elements). This results in matrix equations relating the input at specified points in the elements (nodes) to the outputs at these same points. In order to solve equations over large regions, the matrix equations for the smaller sub-regions are usually summed node by node, resulting in global matrix equations. The finite element method is described in details in many textbooks and articles [41,42,43,44].

3.1 FEM Introduction

Before outlining the FEM problem solving procedure used in this research, the following concepts and related terminology are offered as an introduction to FEM.

The problem begins with the engineer or analyst who wants to describe or predict the response of a system that is subjected to external influences that change the state of the system. S/he is basically looking for a numerical solution to the

governing equations and loading conditions that characterize and determine the behavior of that system. The problem thus becomes mathematical.

Let's define: system, domain, governing equations, and loading conditions.

The system to be analyzed is typically, but not always, a physical object composed of various materials: solids, liquids, gases, plasmas, combinations of these, etc.

The domain of the problem is the region of space occupied by the system with known dimensions.

The governing equations may be differential equations expressing a conservation or balance of some physical property such as mass, momentum, or energy. They may also be integral equations expressing a variational principle, such as the minimization of potential energy for conservative mechanical systems. They may include constitutive equations, which describe particular types of material behavior; these contain experimentally determined physical properties of the materials that constitute the system.

Loading conditions are externally originating forces, temperatures, currents, fields, etc., that interact with the system, causing the state of the system to change. Loads acting in the interior of the domain appear as part of the governing equations. Loads acting on the boundary of the domain appear in separate equations called boundary conditions.

With the above concepts, a brief description of some of the terms used in the finite elements method follows:

- The domain of the problem is divided (partitioned) into smaller regions (subdomains) called elements. Adjacent elements touch without overlapping, and there are no gaps between the elements. The shapes of the elements are intentionally made as simple as possible, such as triangular and quadrilateral in two dimensional domains, and tetrahedral, pentahedral "pyramids", and hexahedral "bricks" in three dimensions. The entire mosaic-like pattern of elements is called a mesh (grid).
- Mesh generation, the process of partitioning a domain into a mesh of elements, was performed manually during the early years of the FEM. However, computer programs have automated this process.
- In each element the governing equations, usually in differential or integral form, are transformed into algebraic equations, called element equations, which are an approximation of the governing equations.
- The terms in the element equations are numerically evaluated for each element in the mesh. The resulting numbers are assembled (combined) into a much larger set of algebraic equations called the system equations. The later characterize the response of the entire system and usually comprise a very large number of equations, typically hundreds or thousands.
- At this point in an FEM the governing equations have been transformed and include the interior loads. The boundary conditions which contain the boundary load have not been dealt with. These are now imposed by modifying the system

equations. This involves adding values to existing terms and/or shifting terms from one side of the equations to the other.

- The resultant system equations are then solved with a computer using conventional numerical analysis techniques.
- The final operation, called postprocessing, displays the solution to the system equations in tabular, graphical, or pictorial form. Other physical meaningful quantities might be derived from the solution and also displayed.

3.2 Finite Element Discretization

The starting point of the finite element method is the subdivision of the domain into small subdomains called elements. An element is described by its vertices and other points on the edges. These points are called the nodes. The FEM mesh is its nodes and elements, Figure 3.1.

For every element, the sought solution is approximated by a polynomial. The approximation is determined at the nodes of the elements which is sufficient information to represent the approximation for the total element. The problem is to determine the value of the approximate solution at the nodes of the FEM mesh for the given PDE and boundary conditions. The weak formulation of the PDE (more general: the functional equation) is evaluated for specific shape functions N_i where:

- N_i is equal to one at the i^{th} node in the FEM mesh.
- N_i is equal to zero on all other nodes.
- N_i is a polynomial on every element.

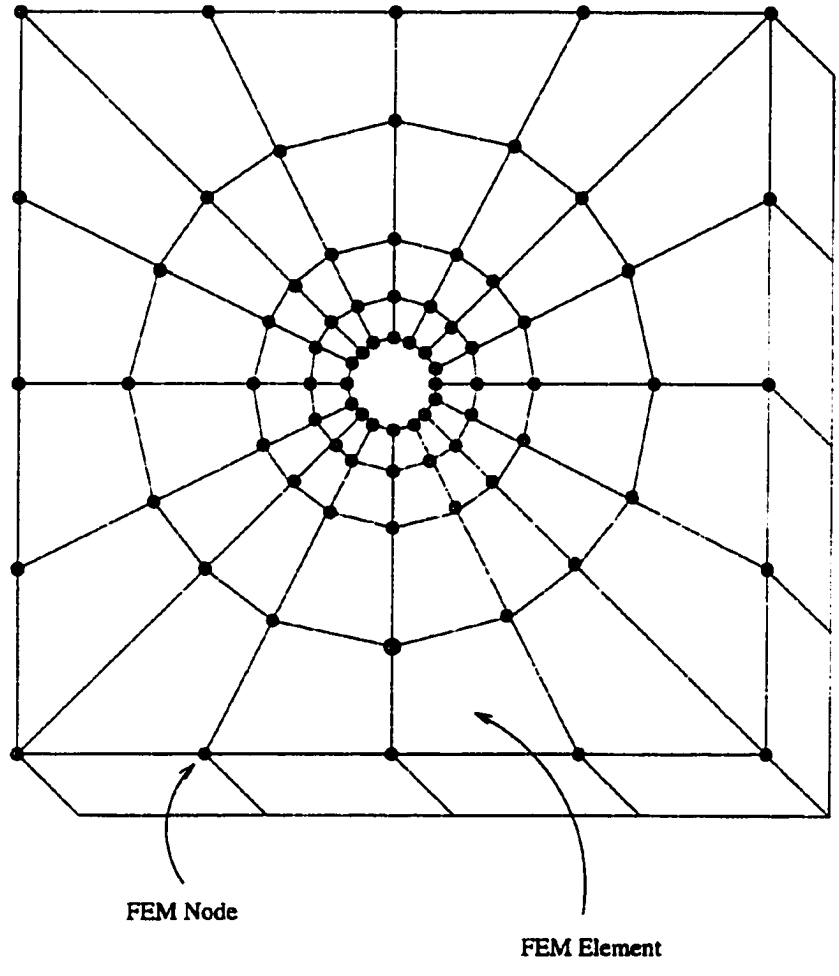


Figure 3.1: Example FEM Mesh

So one obtains a system of equations. The system has NDEG equations (n degrees of freedom), one equation for every node, and NDEG unknowns arising from the values of the sought approximate solution at the nodes. NDEG denotes the number of elements in the FEM mesh. In general, this system is nonlinear and therefore often requires an iterative method be used to calculate the solution of this discrete problem.

3.3 Convergence of the FEM

The mesh quality is important for the finite element method to ensure that a good approximation to the sought solution is calculated. The error e of the FEM approximation depends on the mesh size h . Where h is the maximum mesh size of elements in the mesh.

$$\| e \| = \| N_i - N_h \| \leq Ch^S \tag{3.1}$$

C is a real number which depends on the solution of the governing PDE and the mesh quality but is independent of the mesh size h . This estimation holds only under the assumption that the error from the numerical integration and from any stopping criterion can be neglected. The value for the convergence order s is determined by the smoothness of the sought solution. However, the actual values for C and s cannot be computed, thus the inequality cannot be used to estimate the error of the FEM approximation. The possible values that can be used are found in the published literature.

One can assume $s = 2$ if the solution is approximated by piecewise polynomials of order two. With $s = 2$, the bisection of the elements reduces the size of all elements in the mesh by one half which reduces the error by a factor of 0.25. If the solution approximation is improved by one digit, all elements have to be subdivided into four elements, thus the number of elements used in the problem grows by the factor $4x4x4 = 64$ for a three-dimensional problem. The maximal convergence order $s = 2$ holds only if the solution is smooth enough (the integrals of the square of the third spatial derivatives exist). Thus we used $s < 2$.

Typically the smoothness of the solution is destroyed by singularities. The behavior of the solution in the neighborhood of a singularity allows an approximation of the solution in an optimal manner. In other words, the solution is smooth as long as singularities and small neighborhoods of the singularities are not considered. To reduce the error from the FEM discretization significantly, local refinement of the finite element mesh is necessary in the neighborhood of the singularities. A-priori to a solution, the locations of the singularities are unknown. Likely singularity candidates are re-entrant corners in the domain, but this is not reliable. Therefore, the distribution of the a-posteriori error estimators is inspected to find the locations of large errors and thus refinement of the FEM mesh is performed in a suitable way. For this research all mesh refinement was done manually.

3.4 Possible FEM Benefits

The FEM is a very modular technique. The element equations can be used repeatedly, not only for all the elements in a particular mesh but also for other

problems and in other programs. New types of elements (and hence new sets of element equations) can be added to programs as the need arises, gradually building up an element library, that can be moved from program to program. Then, a user can select from the element library a mesh of different element types, much like a child using different blocks to build a structure. This has a definite impact on human resources, because when a person develops an FE computer program, it can be used to solve not just one specific problem but a whole class of problems that differ substantially in geometry, boundary conditions, and other properties.

CHAPTER 4. MESH SYSTEMS

To obtain approximate solutions to the nonlinear partial differential equations that describe multiphase flow in porous media, finite element discretization of space and finite difference in time are commonly used [45]. This requires a calculation of flow between elements using the pressure for each element. Consequently, this requires knowing the location of both spacial mesh nodes and well mesh nodes (grid points). The number and location of the spacial mesh nodes and the location of boundaries with respect to these element nodes can influence the accuracy of FEM/ FDM approximations.

To obtain the desired accuracy in a reservoir simulation, refinement of the mesh is often necessary in regions of significant change, for example the regions around wells. When a node is refined in some simulators, additional refinement occurs in regions that do not need to be refined. This problem can be avoided by the use of "Local Mesh Refinement" (LMR) techniques.

Some LMR techniques and examples of different meshs are presented in this chapter. Much of the discussion is for a one-dimensional problem, but this can be used to explain a similar mesh with more dimensions.

4.1 Block Centered

A block-centered mesh (grid) is a system used for the simulation of reservoirs by petroleum engineers [1]. In this method, element sizes are chosen: Δx_i,

$i = 1, 2, 3, \ldots, n$; and then the nodes x_i are determined so that they are at the center of these elements. The distance between adjacent nodes can be computed from:

$$\text{distance} = \frac{Dx_i + Dx_{i+1}}{2} \tag{4.1}$$

The nodes are located at the center of each element; there are no nodes located at the mesh boundaries. This is the type of mesh system used in BOAST II. BOAST II can support up to three dimensions with a block centered model.

4.2 Coarse Mesh

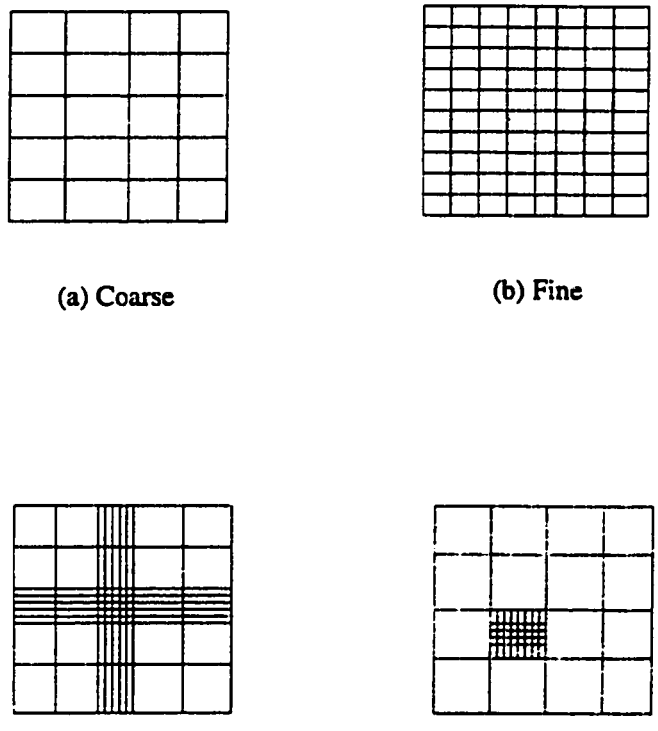
A coarse mesh for reservoir simulation is often used in a preliminary subdivision of the reservoir, Figure 4.1.a. The coarse mesh can be selectively refined by reducing the size of some of its elements. Depending on how elements in the base mesh are subdivided, a mesh can be classified as:

4.2.1 Fine Mesh

A fine mesh could be formed by further subdividing all elements in a coarse mesh, Figure 4.1.b. A fine mesh has smaller elements than a coarse mesh.

4.2.2 Conventionally Refined Mesh

In the conventional approach to mesh refinement, only the area of interest is refined, but the fine mesh lines are extended to the external boundary of the reservoir, Figure 4.1.c. This refinement introduces extra elements in areas far removed from the region of interest.



(a) Coarse

(b) Fine

(c) Conventionally Refined

(d) Locally Refined

Figure 4.1: Coarse Mesh and its Refinements

4.2.3 Locally Refined Mesh

A locally refined mesh could be considered a special case of a conventionally refined mesh. In this, the fine mesh lines are not extended to the external boundary of the reservoir, Figure 4.1.d, but are all within the region of interest. Consequently, the total number of elements used is less than the number used in either the fine mesh or the "conventionally refined mesh". Using a locally refined mesh achieves the desired accuracy of a fine mesh but with substantially fewer elements. This results in less computation time being needed

4.3 Cylindrical Mesh

The near-well flow in an isotropic porous media has radial streamlines and circumferential equipotential lines. A cylindrical mesh system should be used for the well region, as shown in Figure 4.2.

4.4 Treatment of Irregularly Shaped Elements

The elements on the boundary between the wellbore vicinity and reservoir require special treatment because the transmissibility at this boundary must be conserved. This boundary can be described as transitional, i.e., going from radial to rectangular. These surfaces represent radial flow (all the elements in the well block) and linear flow (the reservoir outside the well block).

The flow through a curvilinear surface of an element is radial for an isotropic porous medium in the vicinity of the well. To obtain the necessary consistency in the transmissibility at the boundary between the well block and the remainder of the

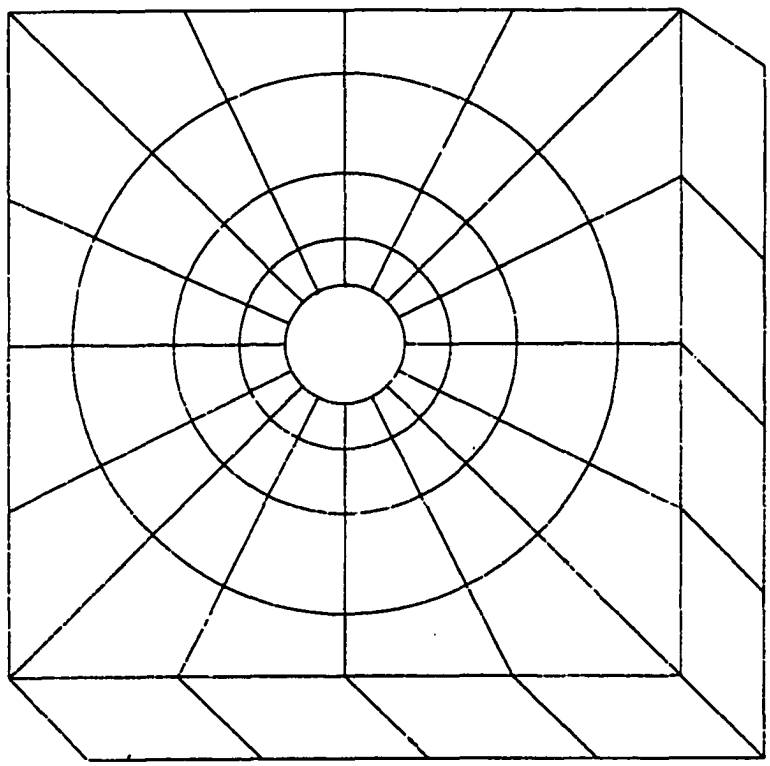


Figure 4.2: Cylindrical Mesh

reservoir, we used the transmissibility calculated by the original reservoir simulator (BOAST II) and distributed this to the nodes on the curvilinear surface with the physical properties of the well block. Our approach uses smaller elements that fit exactly into the space of the original well block which is "removed", Figure 4.3, with no overlap and no omissions.

4.5 Wellbore Vicinity Model

The accuracy of our representation of a well should be improved because the FEM hexahedral elements used in the vicinity of the well block, as shown on Figure 4.3, represents the physical flow process much better. The pressure and saturations within the vicinity of the well obtained during the simulation can be used to refine mesh accordingly. This is a viable well model for others to consider. To interface our new well block model requires the transmissibilities at the boundaries of the well block plus either the well production or pressure histories. These histories may be either actual or estimated.

The transmissibilities of the three phases (oil, water and gas) that are needed for the interface between the well block and its neighboring blocks are already computed by the reservoir simulator (BOAST II). These transmissibilities were used and coupled our new well model with BOAST II. This coupling assumes Darcy flow at all interfaces. Using this assumption, an average phase volumetric flow-rate can be computed from the transmissibility for each face of the well block. Additional details of this interfacing are presented in Chapter 6.

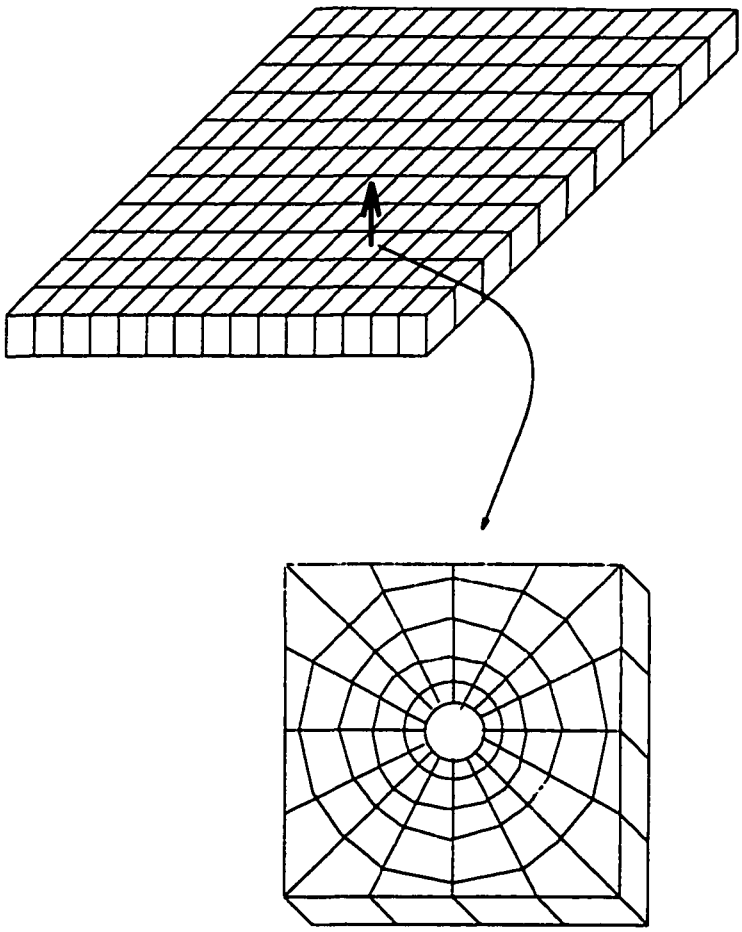


Figure 4.3: Wellbore Vicinity Model

CHAPTER 5. NUMERICAL MODEL

The approximation of the solution of the fundamental equations for the flow of fluid in a porous media; i.e., an oil reservoir, requires a numerical approach. The most popular numerical approach to the simulation of an oil reservoir is the finite difference method. The primary reason for this is that this method is exceptionally easy to understand and use. In addition, almost all of the early and many current oil reservoir simulations are performed using the finite difference method. Finite difference simulation of reservoirs started with Peaceman & Rachford [4].

The first use of the finite element method (FEM) for oil reservoir simulation used Galerkin's method and rectangular elements. With rectangular elements, the FEM has the potential to produce more accurate solutions than the finite difference method by using mesh refinement or higher degree polynomials as basis functions. FEM simulations of oil reservoirs began with Price [46]. Although FEM has the potential to overcome several known numerical deficiencies of the finite difference method, its growth in oil reservoir simulation was slow. Eventhough the FEM could be more accurate than finite differences, it was viewed as too computationally difficult to be practical. Much of the early FEM development has occurred in other fields of engineering [10], [47], and [48]. Today, the FEM is often used for oil reservoir simulations [29]. We chose to use the FEM and Galerkin's method because we wanted the potential for increased accuracy. We also used hexahedral elements (3-D) instead of rectangles (2-D).

In the finite difference method, instead of computing a continuous, sufficiently smooth function for the solution which satisfies the fundamental PDEs, average values are sought that approximate the average solution for a finite set of nodes inside the problem domain. In the FEM, one seeks a solution function which minimizes (Galerkin method) the integral of the residual of the fundamental equations among all functions that are sufficiently smooth and that satisfy the boundary conditions. One advantage of the FEM is the ability to handle arbitrary boundary geometry.

Another important feature of FEM is the ability to handle truly arbitrary boundary conditions and to include non-homogeneous materials. These features mean that we can more accurately simulate oil reservoirs of arbitrary shapes that are composed of many different material regions. These different material regions could have constant properties or properties that vary with spacial location. We also have more freedom in applying boundary conditions.

5.1 Discretization of the Flow Equations

The equations for a finite element method oil reservoir simulation will be developed. These equations can be used to approximate the triphasic flow of compressible-variably saturated fluids through a porous media. The governing equations for triphasic flow will be developed using the BOAST technical manual [5] as a model. The equations that govern the flow of variably saturated-immissible fluids through porous media are derived from combining the equation of continuity of

mass equation and Darcy's Law. The fundamental law of momentum conservation cannot be applied to describe the fluid flow in a porous media because it would require a precise knowledge of the system's geometry.

The geometry of the porous space in rock is very complex and irregular, and is not reasonably defined for oil reservoirs [46]. Therefore, instead of the momentum, Darcy's Law is substituted, which relates the fluid flow velocity components, and the flow potential gradient for each phase using the fluid pressure P and the porosity j of the porous medium.

Figure 5.1 represents flow of a fluid into and out of a single reservoir right parallelpiped (hexahedron):

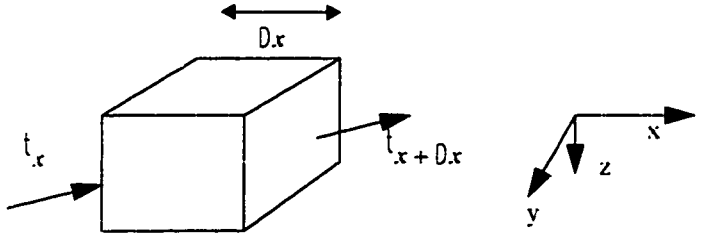


Figure 5.1: The Coordinate Convention Follows BOAST's

As shown by the arrows, the fluid that flows into this right parallelpiped (hexahedron) at x is t_x and the fluid that flows out of the right parallelpiped (hexahedron) at $x + Dx$ is t_{x+Dx}. t denotes the fluid flux and is defined as the rate of flow of mass per unit cross-sectional area normal to the direction of flow. Invoking various simplifying assumptions, the conservation of mass yields:

mass_entering - mass_exiting =

accumulation_of_mass_in_parallelpiped (5.1)

If the parallelpiped has length Δx, width Δy, and depth Δz, then the mass entering and the mass leaving are:

$$\text{mass in} = [(t_x)_x \Delta y \Delta z + (t_y)_y \Delta x \Delta z + (t_z)_z \Delta x \Delta y]\Delta t \tag{5.2}$$

$$\text{mass out} = (t_x)_{x+\Delta x} \Delta y \Delta z + (t_y)_{y+\Delta y} \Delta x \Delta z + (t_z)_{z+\Delta z} \Delta x \Delta y]\Delta t + q \Delta x \Delta y \Delta z \Delta t \tag{5.3}$$

where q is a source/sink term that represents the effective mass pumped into this parallelpiped (a source) or the mass withdraw from the parallelpiped (a sink). Both the sink and the source are "wells".

The accumulation of mass in this parallelpiped of each phase p is the change of concentration of that phase c_p in the time interval Δt. If the concentration c_p is defined as the total mass of phase p (oil, water, or gas) in this parallelpiped divided by its volume, then the accumulation term for phase p is:

$$[(c_p)_{t+\Delta t} - (c_p)_t]\Delta x \Delta y \Delta z \tag{5.4}$$

Mass In - Mass Out = Mass Accumulation (5.5)

$$
\begin{aligned}
&[(t_x)_x \Delta y \Delta z + (t_y)_y \Delta x \Delta z + (t_z)_z \Delta x \Delta y] \Delta t \\
&-[(t_x)_{x+\Delta x} \Delta y \Delta z + (t_y)_{y+\Delta y} \Delta x \Delta z \\
&+ (t_z)_{z+\Delta z} \Delta x \Delta y (\Delta t + -q \Delta x \Delta y \Delta z \Delta t) \\
&= [(c_p)_{t+\Delta t} - (c_p)_t] \Delta x \Delta y \Delta z \\
&-\left[\frac{(t_x)_{x+\Delta x} - (t_x)_x}{\Delta x}\right] - \left[\frac{(t_y)_{y+\Delta y} - (t_y)_y}{\Delta y}\right] \\
&-\left[\frac{(t_z)_{z+\Delta z} - (t_z)_z}{\Delta z}\right] - q \\
&= \left[\frac{(c_p)_{t+\Delta t} - (c_p)_t}{\Delta t}\right]
\end{aligned}
\tag{5.6}
$$

and in the limit as $\Delta x, \Delta y, \Delta z, \Delta t \to 0$ is:

$$
-\frac{\partial t_x}{\partial x} - \frac{\partial t_y}{\partial y} - \frac{\partial t_z}{\partial z} - q = \frac{\partial c_p}{\partial t} \tag{5.7}
$$

Each phase satisfies the mass conservation equation (5.6). In this formulation, we consider three fluid phases: oil, water, and gas. It is assumed that a one-way transfer occurs in the form of gas into either water or oil. The oil component refers to the residual liquid at atmospheric pressure left after a differential vaporization and the same for the water phase. The gas component refers to the remaining fluid.

The flux in a given direction can be written as the density of the fluid (ρ_p) multiplied by its velocity (V_p) in that given direction, where the subscripts used

were o - oil, w - water, and g - gas. To account for volume changes due to different pressures at the reservoir and the surface, a "*formation volume factor*" is introduced for each phase (B_p). B_p is the volume of phase p as a ratio of its volume measured at reservoir conditions to its volume measured at standard surface conditions (60° F and 14.7 psia). The dissolved volume of gas measured in cubic feet (SCF) at standard surface conditions per barrel of stock tank oil is given as the ratio R_{so} and R_{sw} is the volume of gas measured at standard conditions dissolved in a barrel of stock tank (STB) water. The units for R_{so} and R_{sw} are SCF/STB.

$$(\vec{i})_o = \frac{\rho_o \vec{V}_o}{B_o} \tag{5.8}$$

$$(\vec{i})_w = \frac{\rho_w \vec{V}_w}{B_w} \tag{5.9}$$

$$(\vec{i})_g = \frac{\rho_g \vec{V}_g}{B_g} + \frac{R_{so}\rho_g \vec{V}_o}{B_o} + \frac{R_{sw}\rho_g \vec{V}_w}{B_w} \tag{5.10}$$

Using Darcy velocities, the x-components for individual phases are:

$$(v_o)_x = -K_x \lambda_o \frac{1}{\rho_{o,x}}\left[P_o - \frac{\rho_o g z}{144 g_c}\right] \tag{5.11}$$

$$(v_w)_x = -K_x \lambda_w \frac{1}{\rho_{w,x}}\left[P_w - \frac{\rho_w g z}{144 g_c}\right] \tag{5.12}$$

$$(v_g)_x = -K_x \lambda_g \frac{1}{\mathfrak{f}_x}\left[P_g - \frac{r\phi_g g z}{144 g_c}\right] \tag{5.13}$$

Let's assume $g = g_c$. Similar expressions for the velocities can be written for the y and z components. Each phase mobility factor λ_p is defined as the ratio of the relative permeability to flow of the phase divided by its viscosity, $\lambda_p = \frac{K_{rp}}{\mu_p}$. The phase densities can be expressed as:

$$r\phi_o = \frac{1}{B_o}[r_o + R_{so} r_g] \tag{5.14}$$

$$r\phi_w = \frac{1}{B_w}[r_w + R_{sw} r_g] \tag{5.15}$$

$$r\phi_g = \frac{r_g}{B_g} \tag{5.16}$$

and the phase concentrations as:

$$c_o = \frac{\phi r_o S_o}{B_o} \tag{5.17}$$

$$c_w = \frac{\phi r_w S_w}{B_w} \tag{5.18}$$

$$c_g = \phi r_g\left[\frac{S_g}{B_g} + \frac{R_{so} S_o}{B_o} + \frac{R_{sw} S_w}{B_w}\right] \tag{5.19}$$

where ϕ is the porosity and S_p is the saturation of phase p. Combining these equations[1], (5.7) through (5.10) and (5.17) through (5.19), which perform volumetric balances with respect to surface conditions, gives the following conservation equations for the individual phase components:

Oil

$$-\left[\frac{1}{\Delta x}\left(\frac{\rho_o}{B_o}V_{xo}\right)_{x+\frac{\Delta x}{2}} + \frac{1}{\Delta y}\left(\frac{\rho_o}{B_o}V_{yo}\right)_{y+\frac{\Delta y}{2}} + \frac{1}{\Delta z}\left(\frac{\rho_o}{B_o}V_{zo}\right)_{z+\frac{\Delta z}{2}}\right] - q_o = \frac{1}{\Delta t}\left(\frac{\phi\rho_o S_o}{B_o}\right)_{t+\Delta t} \tag{5.20}$$

Water

$$-\left[\frac{1}{\Delta x}\left(\frac{\rho_w}{B_w}V_{xw}\right)_{x+\frac{\Delta x}{2}} + \frac{1}{\Delta y}\left(\frac{\rho_w}{B_w}V_{yw}\right)_{y+\frac{\Delta y}{2}} + \frac{1}{\Delta z}\left(\frac{\rho_w}{B_w}V_{zw}\right)_{z+\frac{\Delta z}{2}}\right] - q_w = \frac{1}{\Delta t}\left(\frac{\phi\rho_w S_w}{B_w}\right)_{t+\Delta t} \tag{5.21}$$

Gas

$$-\frac{1}{\Delta x}\left[\frac{\rho_g}{B_g}V_{xg} + \frac{R_{so}\rho_g}{B_o}V_{xo} + \frac{R_{sw}\rho_g}{B_w}V_{xw}\right] - \frac{1}{\Delta y}\left[\frac{\rho_g}{B_g}V_{yg} + \frac{R_{so}\rho_g}{B_o}V_{yo} + \frac{R_{sw}\rho_g}{B_w}V_{yw}\right]$$

$$-\frac{1}{\Delta z}\left[\frac{\rho_g}{B_g}V_{zg} + \frac{R_{so}\rho_g}{B_o}V_{zo} + \frac{R_{sw}\rho_g}{B_w}V_{zw}\right] - q_g =$$

$$-\frac{1}{\Delta t}\left[\phi\rho_g\left(\frac{S_g}{B_g} + \frac{R_{so}S_o}{B_o} + \frac{R_{sw}S_w}{B_w}\right)_{t+\Delta t}\right] \tag{5.22}$$

---

1. All densities are at standard surface conditions.

where q_o, q_w, and q_g are the production rates of oil, water, and gas per unit volume of the porous medium.

The densities at standard conditions are constants, and these equations can be reduced as:

Oil:

$$-\left[\frac{1}{1x}\left(\frac{1}{B_o}V_{xo}\right)_{\theta} + \frac{1}{1y}\left(\frac{1}{B_o}V_{yo}\right)_{\theta} + \frac{1}{1z}\left(\frac{1}{B_o}V_{zo}\right)_{\theta}\right] - \frac{q_o}{r_o} = \frac{1}{1t}\left(\frac{S_o}{B_o}\right)_{\theta} \tag{5.23}$$

Water:

$$-\left[\frac{1}{1x}\left(\frac{1}{B_w}V_{xw}\right)_{\theta} + \frac{1}{1y}\left(\frac{1}{B_w}V_{yw}\right)_{\theta} + \frac{1}{1z}\left(\frac{1}{B_w}V_{zw}\right)_{\theta}\right] - \frac{q_w}{r_w} = \frac{1}{1t}\left(\frac{S_w}{B_w}\right)_{\theta} \tag{5.24}$$

Gas:

$$-\frac{1}{1x}\left(\frac{V_{xg}}{B_g} + \frac{R_{so}V_{xo}}{B_o} + \frac{R_{sw}V_{xw}}{B_w}\right)_{\theta} - \frac{1}{1y}\left(\frac{V_{yg}}{B_g} + \frac{R_{so}V_{yo}}{B_o} + \frac{R_{sw}V_{yw}}{B_w}\right)_{\theta} -$$

$$\frac{1}{1z}\left(\frac{V_{zg}}{B_g} + \frac{R_{so}V_{zo}}{B_o} + \frac{R_{sw}V_{zw}}{B_w}\right)_{\theta} - \frac{q_g}{r_g} = \frac{1}{1t}\left[\left(\frac{S_g}{B_g} + \frac{R_{so}S_o}{B_o} + \frac{R_{sw}S_w}{B_w}\right)_{\theta}\right] \tag{5.25}$$

(5.23) through (5.25), can be written:

$$-\nabla \cdot \left(\frac{\vec{u}_o}{B_o}\right)_{\theta} - \frac{q_o}{r_o} = \frac{1}{1t}\left(\frac{S_o}{B_o}\right)_{\theta} \tag{5.26}$$

$$-\nabla\cdot\frac{\vec{u}_w}{B_w}-\frac{q_w}{\rho_w}=\frac{\partial}{\partial t}\left(\frac{\phi S_w}{B_w}\right) \tag{5.27}$$

$$-\nabla\cdot\left[\frac{\vec{u}_g}{B_g}+\frac{R_{so}\vec{u}_o}{B_o}+\frac{R_{sw}\vec{u}_w}{B_w}\right]-\frac{q_g}{\rho_g}=\frac{\partial}{\partial t}\left[\phi\left(\frac{S_g}{B_g}+\frac{R_{so}S_o}{B_o}+\frac{R_{sw}S_w}{B_w}\right)\right] \tag{5.28}$$

where the vector notation represents:

$$\nabla\cdot\vec{u}_p=\frac{\partial}{\partial x}u_{xp}+\frac{\partial}{\partial y}u_{yp}+\frac{\partial}{\partial z}u_{zp} \tag{5.29}$$

These continuity equations for each phase can also be found in [1], [6], or [48].

Define a phase p potential[1], Ψ_p as:

$$\Psi_p=P_p-\frac{\rho_p Z}{144} \tag{5.30}$$

Then we introduce a dyadic notation $\hat{M}_o$, M_w, and M_g to represent a tensor of rank two for the mobility of each phase.

$$\hat{M}_o=\frac{1}{\mu_o}(\vec{\vec{k}}_{ro}) \tag{5.31}$$

where $(\vec{\vec{k}}_{ro})$ is the permeability matrix whose terms are given individually by

---

1. The density is in $lb(mass)/ft^3$ and the division by 144 converts this into psi. A density of $1 gm/cm^3 = 62.4 lbs/ft^3$.

the product $k_{ijk}k_{ro}$, where k_{ijk} is a local absolute permeability and k_{ro} the relative permeability of oil, and μ_o the viscosity of oil. A similar dyadic can be developed for the water and gas phases:

$$\hat{M}_w = \frac{1}{\mu_w}(\vec{k}\vec{k}_{rw}) \tag{5.32}$$

$$\hat{M}_g = \frac{1}{\mu_g}(\vec{k}\vec{k}_{rg}) \ . \tag{5.33}$$

Then let:

$$\hat{\lambda}_o = \frac{\hat{M}_o}{B_o} = \frac{k_{ro}}{B_o \mu_o} \tag{5.34}$$

$$\hat{\lambda}_w = \frac{\hat{M}_w}{B_w} = \frac{k_{rw}}{B_w \mu_w} \tag{5.35}$$

$$\hat{\lambda}_g = \frac{M_g}{B_g} = \frac{k_{rg}}{B_g \mu_g} \tag{5.36}$$

and the Darcy velocities may be written as:

$$\vec{v}_o = -\hat{M}_o \cdot \nabla\Upsilon_o \tag{5.37}$$

$$\vec{v}_w = -\hat{M}_w \cdot \nabla\Upsilon_w \tag{5.38}$$

$$\vec{v}_g = -\hat{M}_g \cdot \nabla\Upsilon_g \tag{5.39}$$

Using this notation, (5.26) through (5.28) become:

$$\nabla \cdot (\hat{i}_o \cdot \nabla \Upsilon_o) - \frac{q_o}{r_o} = \frac{1}{1t}\frac{\partial}{\partial t}\left(\frac{S_o}{B_o}\right) \tag{5.40}$$

$$\nabla \cdot (\hat{i}_w \cdot \nabla \Upsilon_w) - \frac{q_w}{r_w} = \frac{1}{1t}\frac{\partial}{\partial t}\left(\frac{S_w}{B_w}\right) \tag{5.41}$$

$$\nabla \cdot [\hat{i}_g \cdot \nabla \Upsilon_g + R_{so}\hat{i}_o \cdot \nabla \Upsilon_o + R_{sw}\hat{i}_w \cdot \nabla \Upsilon_w] - \frac{q_g}{r_g} =$$

$$\frac{1}{1t}\left[\frac{\partial}{\partial t}\left(\frac{S_g}{B_g} + \frac{R_{so}S_o}{B_o} + \frac{R_{sw}S_w}{B_w}\right)\right] \tag{5.42}$$

Equations (5.40) through (5.42) form the required system of partial differential equations for triphasic flow in the "black oil" reservoir system. Examining these three equations reveals nine unknowns, namely, Υ_o, Υ_w, Υ_g, k_{ro}, k_{rw}, k_{rg}, S_o, S_w, and S_g. To solve these three equations will require six auxiliary equations to obtain a general solution. The first four of these auxiliary equations are:

$$k_{ro} = k_{ro}(S_o, S_w) \tag{5.43}$$

$$k_{rw} = k_{rw}(S_o, S_w) \tag{5.44}$$

$$k_{rg} = k_{rg}(S_o, S_w) \tag{5.45}$$

$$S_o + S_w + S_g = 1 \tag{5.46}$$

The remaining two auxiliary equations come from considering capillary pressure. "Capillary pressure exists because of the interfacial tensions between the liquids and the contact angles between the rock and the fluid phases. In general, a capillary pressure associated with two fluid phases is defined as the difference between pressures of the nonwetting and wetting phases. Because we have three fluids (oil, water, and gas) flowing together it is necessary to introduce two capillary pressures, ..." [48]. The capillary pressure of oil-to-water P_{cow} and the capillary pressure of gas-to-oil P_{cgo} are what we will use to express these capillary pressures.

Equations (5.40) through (5.42) have $\gamma_o, \gamma_w, \gamma_g$ representing the phase potentials and representing phase pressures P_o, P_w , and P_g these unknowns can be reduced with the use of capillary pressures:

$$P_{cow} = P_o - P_w \tag{5.47}$$

$$P_{cgo} = P_g - P_o \tag{5.48}$$

Experimentally P_{cow} (capillary pressure of oil-to-water) and P_{cgo} (capillary pressure of gas-to-oil) can be measured and are assumed to be functions of S_o, S_w, and S_g only. Using these equations and (5.30), the water and gas potentials are:

$$\gamma_w = P_o - P_{cow} - \frac{r'_w Z}{144} \tag{5.49}$$

$$\gamma_g = P_o - P_{cgo} - \frac{r'_g Z}{144} \tag{5.50}$$

$$\gamma_o = P_o - \frac{r'_o Z}{144} \tag{5.51}$$

Combining (5.49) and (5.51) with (5.40) through (5.42), we obtain the following oil, water, and gas equations:

Oil:

$$\nabla \cdot (\lambda_o \cdot \nabla P_o) - \nabla \cdot \left[\lambda_o \nabla \frac{r'_o Z}{144}\right] - \frac{q_o}{r_o} = \frac{1}{1t}\left[\frac{S_o}{B_o}\right] \tag{5.52}$$

Water:

$$\nabla \cdot (\lambda_w \cdot \nabla P_o) - \nabla \cdot \left[\lambda_w \nabla \frac{r'_w Z}{144} + P_{cow}\right] - \frac{q_w}{r_w} = \frac{1}{1t}\left[\frac{S_w}{B_w}\right] \tag{5.53}$$

Gas:

$$\nabla \cdot [\lambda_g \cdot \nabla P_o + P_{cgo} - \frac{r'_g Z}{144} + R_{so}\lambda_o \cdot \nabla P_o - \frac{r'_o Z}{144}$$

$$+ R_{sw}\lambda_w \cdot \nabla P_o - P_{cow} - \frac{r'_w Z}{144}] - \frac{q_g}{r_g} = \frac{1}{1t}\left[\frac{S_g}{B_g} + \frac{R_{so}S_o}{B_o} + \frac{R_{sw}S_w}{B}\right] \tag{5.54}$$

Equations (5.52) through (5.54) and (5.46) are the equations to be solved for P_o,

S_o, S_w, and S_g. All other physical properties are known either as functions of the unknowns, or from field and laboratory data. In the IMPES procedure used in our FEM-BOAST model, we first obtain an implicit solution of one of these equations for the pressure P_o. To do this we will combine the flow equations (5.52) through (5.54) and (5.46).

Since

$$S_o = 1 - S_w - S_g \tag{5.55}$$

we have:

$$\frac{dS_o}{dt} = -\frac{dS_w}{dt} - \frac{dS_g}{dt} \tag{5.56}$$

and

$$\frac{\partial}{\partial t}\left(\frac{S_o}{B_o}\right) = \frac{1}{B_o}\cdot\frac{\partial S_o}{\partial t} + \frac{S_o}{B_o}\cdot\frac{\partial}{\partial t} + S_o\left(-\frac{1}{B_o^2}\right)\cdot\frac{\partial B_o}{\partial t} \tag{5.57}$$

$$\frac{\partial}{\partial t} = \frac{d}{dP_o}\cdot\frac{\partial P_o}{\partial t} \tag{5.58}$$

$$\frac{\partial B_o}{\partial t} = \frac{dB_o}{dP_o}\cdot\frac{\partial P_o}{\partial t} \tag{5.59}$$

Therefore,

$$\frac{\partial}{\partial t}\left(\frac{S_o}{B_o}\right) = \frac{1}{B_o}\cdot\frac{\partial S_o}{\partial t} + \frac{S_o}{B_o}\left(\frac{d}{dP_o}\right)\frac{\partial P_o}{\partial t} - \frac{S_o}{B_o^2}\left(\frac{dB_o}{dP_o}\right)\frac{\partial P_o}{\partial t} \tag{5.60}$$

$$= \frac{1}{B_o}\left[\frac{\partial S_o}{\partial t} + S_o\left(\frac{1}{B_o}\frac{dB_o}{dP_o}\right)\frac{\partial P_o}{\partial t}\right] \tag{5.61}$$

Similarly, we have:

$$\frac{\partial}{\partial t}\left(\frac{S_w}{B_w}\right) = \frac{1}{B_w}\cdot\frac{\partial S_w}{\partial t} + \frac{S_w}{B_w}\frac{\partial}{\partial t} - \frac{S_w}{B_w^2}\cdot\frac{\partial B_w}{\partial t} \tag{5.62}$$

$$\frac{\partial}{\partial t} = \frac{d}{dP_o}\cdot\frac{\partial P_o}{\partial t} \tag{5.63}$$

$$\frac{\partial B_w}{\partial t} = \frac{dB_w}{dP_o}\cdot\frac{\partial P_o}{\partial t} \tag{5.64}$$

$$\frac{\partial}{\partial t}\left(\frac{S_w}{B_w}\right) = \frac{1}{B_w}\left[\frac{\partial S_w}{\partial t} + S_w\left(\frac{1}{B_w}\frac{dB_w}{dP_o}\right)\frac{\partial P_o}{\partial t}\right] \tag{5.65}$$

For the gas equation, we treat it as three parts.

$$\frac{\partial}{\partial t}\left[\frac{S_g}{B_g} + \frac{R_{so}S_o}{B_o} + \frac{R_{sw}S_w}{B_w}\right] = L_1 + L_2 + L_3 \tag{5.66}$$

where L_1, L_2, and L_3 are as follows:

$$L_1 = \frac{\partial}{\partial t}\left(\frac{S_g}{B_g}\right) = \frac{S_g}{B_g}\frac{\partial}{\partial t} + \frac{1}{B_g}\frac{\partial S_g}{\partial t} - \frac{S_g}{B_g^2}\frac{\partial B_g}{\partial t} \tag{5.67}$$

By using the following two equations:

$$\frac{\partial \rho}{\partial t} = \frac{d\rho}{dP_o}\frac{\partial P_o}{\partial t} \tag{5.68}$$

$$\frac{\partial B_g}{\partial t} = \frac{dB_g}{dP_o}\frac{\partial P_o}{\partial t} \tag{5.69}$$

we have:

$$L_1 = \frac{S_g}{B_g}\left[\frac{d\rho}{dP_o}\right]\frac{\partial P_o}{\partial t} + \frac{\rho}{B_g}\frac{\partial S_g}{\partial t} - \frac{\rho S_g}{B_g^2}\left[\frac{dB_g}{dP_o}\right]\frac{\partial P_o}{\partial t} \tag{5.70}$$

$$= \frac{\rho}{B_g}\frac{\partial S_g}{\partial t} + \frac{\rho S_g}{B_g}\left[\frac{1}{\rho}\frac{d\rho}{dP_o} - \frac{1}{B_g}\frac{dB_g}{dP_o}\right]\frac{\partial P_o}{\partial t} \tag{5.71}$$

$$L_2 = \frac{1}{\partial t}\left[\frac{\rho R_{so} S_o}{B_o}\right] \tag{5.72}$$

$$= \frac{R_{so} S_o}{B_o}\frac{\partial \rho}{\partial t} + \frac{\rho S_o}{B_o}\frac{\partial R_{so}}{\partial t} + \frac{\rho R_{so}}{B_o}\frac{\partial S_o}{\partial t} - \frac{\rho R_{so} S_o}{B_o^2}\frac{\partial B_o}{\partial t} \tag{5.73}$$

$$= \frac{\rho S_o}{B_o}\left[\frac{R_{so}}{\rho}\frac{d\rho}{dP_o} + \frac{dR_{so}}{dP_o} - \frac{R_{so}}{B_o}\frac{dB_o}{dP_o}\right]\frac{\partial P_o}{\partial t} + \frac{\rho R_{so}}{B_o}\frac{\partial S_o}{\partial t} \tag{5.74}$$

$$L_3 = \frac{1}{\partial t}\left[\frac{\rho R_{sw} S_w}{B_w}\right] \tag{5.75}$$

$$= \frac{R_{sw}S_w}{B_w}\frac{\partial}{\partial t} + \frac{S_w}{B_w}\frac{\partial R_{sw}}{\partial t} + \frac{R_{sw}}{B_w}\frac{\partial S_w}{\partial t} - \frac{R_{sw}S_w}{B^2_w}\frac{\partial B_w}{\partial t} \tag{5.76}$$

$$= \frac{S_w}{B_w}\left[\frac{R_{sw}}{}\frac{d}{dP_o} + \frac{dR_{sw}}{dP_o} - \frac{R_{sw}}{B_w}\frac{dB_w}{dP_o}\right]\frac{\partial P_o}{\partial t} + \frac{R_{sw}}{B_w}\frac{\partial S_w}{\partial t} \tag{5.77}$$

Combining L_1, L_2, and L_3 we have:

$$L_1 + L_2 + L_3 = \frac{}{B_g}\frac{\partial S_g}{\partial t} + \frac{R_{so}}{B_o}\frac{\partial S_o}{\partial t} + \frac{R_{sw}}{B_w}\frac{\partial S_w}{\partial t} +$$

$$\frac{}{B_g}\frac{S_g}{}\frac{d}{dP_o} - \frac{S_g}{B_g}\frac{dB_g}{dP_o} + \frac{R_{so}S_oB_g}{B_o}\frac{d}{dP_o} + \frac{S_oB_g}{B_o}\frac{dR_{so}}{dP_o}$$

$$-\frac{R_{so}S_oB_g}{B_o^2}\frac{1}{B_g}\frac{dB_o}{dP_o} + \frac{R_{sw}S_wB_g}{B_w}\frac{d}{dP_o} + \frac{S_wB_g}{B_w}\frac{dR_{sw}}{dP_o}$$

$$-\frac{B_gR_{sw}S_w}{B^2_w}\frac{dB_w}{dP_o}\frac{\partial P_o}{\partial t} \tag{5.78}$$

(5.46) is now used to eliminate $\frac{\partial S_g}{\partial t}$ from (5.78). Differentiation of (5 46) by t and rearranging gives:

$$\frac{\partial S_g}{\partial t} = -\frac{\partial S_o}{\partial t} - \frac{\partial S_w}{\partial t} \tag{5.79}$$

substituting (5.79) into (5.78) and simplifying:

$$L_1 + L_2 + L_3 = \left(\frac{R_{so}}{B_o} - \frac{1}{B_{g\emptyset}}\right)\frac{\Delta S_o}{\Delta t} + \left(\frac{R_{sw}}{B_w} - \frac{1}{B_{g\emptyset}}\right)\frac{\Delta S_w}{\Delta t}$$

$$+ \left(\frac{S_g}{B_g}\frac{\Delta}{\Delta P_o} - \frac{S_g}{B_g^2}\frac{B_g}{\Delta P_o} + \frac{R_{so}S_o}{B_o}\frac{\Delta}{\Delta P_o} + \frac{S_o}{B_o}\frac{\Delta R_{so}}{\Delta P_o} - \frac{S_o R_{so}}{B_o^2}\frac{\Delta B_o}{\Delta P_o}\right)$$

$$+ \frac{R_{sw}S_w}{B_w}\frac{\Delta}{\Delta P_o} + \frac{S_w}{B_w}\frac{\Delta R_{sw}}{\Delta P_o} - \frac{R_{sw}S_w}{B_w^2}\frac{\Delta B_w}{\Delta P_{o\emptyset}}\frac{\Delta P_o}{\Delta t} \qquad (5.80)$$

Equations (5.71), (5.77), and (5.80) are three equations in three unknowns P_o, S_o, and S_w. Multiplying (5.71) by $(B_o - R_{so}B_g)$, (5 77) by $(B_w - R_{sw}B_g)$, (5.80) by B_g, and adding:

$$(B_o - R_{so}B_g)L_1 + (B_w - R_{sw}B_g)L_2 + B_g L_3 =$$

$$(B_o - R_{so}B_g)\frac{1}{B_o}\frac{\Delta S_o}{\Delta t} + (B_w - R_{sw}B_g)\frac{1}{B_w}\frac{\Delta S_w}{\Delta t}$$

$$+ B_g\left(\frac{R_{so}}{B_o} - \frac{1}{B_{g\emptyset}}\right)\frac{\Delta S_o}{\Delta t} + B_g\left(\frac{R_{sw}}{B_w} - \frac{1}{B_{g\emptyset}}\right)\frac{\Delta S_w}{\Delta t}$$

$$+ (B_o - R_{so}B_g)\left(\frac{S_o}{B_o}\frac{\Delta}{\Delta P_o} - \frac{S_o}{B_o^2}\frac{\Delta B_o}{\Delta P_{o\emptyset}}\right)\frac{\Delta P_o}{\Delta t}$$

$$+ (B_w - R_{sw}B_g)\left(\frac{S_w}{B_w}\frac{\Delta}{\Delta P_o} - \frac{S_w}{B_w^2}\frac{\Delta B_w}{\Delta P_{o\emptyset}}\right)\frac{\Delta P_o}{\Delta t}$$

$$+ \left(\frac{S_g}{\Delta P_o} - \frac{S_g}{B_g}\frac{B_g}{\Delta P_o} + \frac{B_g R_{so}S_o}{B_o}\frac{\Delta}{\Delta P_o} + \frac{B_g S_o}{B_o}\frac{\Delta R_{so}}{\Delta P_o}\right)$$

$$-\frac{S_o R_{so}}{B_o^2}\frac{\partial B_o}{\partial P_o}+\frac{B_g R_{sw} S_w}{B_w}\frac{\partial}{\partial P_o}+\frac{B_g S_w}{B_w}\frac{\partial R_{sw}}{\partial P_o}$$

$$-\frac{B_g R_{sw} S_w}{B_w^2}\frac{\partial B_w}{\partial P_o}\frac{\partial P_o}{\partial t} \qquad (5.81)$$

After simplification:

$$(B_o - R_{so}B_g)L_1 + (B_w - R_{sw}B_g)L_2 + B_g L_3 =$$

$$\left[(S_o + S_w + S_g)\frac{\partial}{\partial P_o} - \frac{S_g}{B_g}\frac{\partial B_g}{\partial P_o} + S_o\left(\frac{B_g}{B_o}\frac{\partial R_{so}}{\partial P_o}\right.\right.$$

$$\left.\left.-\frac{1}{B_o}\frac{\partial B_o}{\partial P_o}\right) + S_w\left(\frac{B_g}{B_w}\frac{\partial R_{sw}}{\partial P_o}-\frac{1}{B_w}\frac{\partial B_w}{\partial P_o}\right)\right]\frac{\partial P_o}{\partial t} \qquad (5.82)$$

Compressibilities of oil C_o, water C_w, gas C_g, and rock C_r:

$$C_o = -\frac{1}{B_o}\frac{\partial B_o}{\partial P_o}+\frac{B_g}{B_o}\frac{\partial R_{so}}{\partial P_o} \qquad (5.83)$$

$$C_w = -\frac{1}{B_w}\frac{\partial B_w}{\partial P_o}+\frac{B_g}{B_w}\frac{\partial R_{sw}}{\partial P_o} \qquad (5.84)$$

$$C_g = -\frac{1}{B_g}\frac{\partial B_g}{\partial P_o} \qquad (5.85)$$

$$C_r = \frac{1}{\phi}\frac{\partial \phi}{\partial P_o} \qquad (5.86)$$

and the total compressibility:

$$C_t = C_r + S_o C_o + S_w C_w + S_g C_g \tag{5.87}$$

Employing these definitions in (5.52) through (5.54) and (5.46) in (5.82)

$$(B_o - R_{so}B_g)\left[\nabla \cdot \vec{k}\,\frac{\lambda_o}{B_o}\nabla P_o + CG_o - \frac{q_o}{r_{osc}}\right]$$

$$+ (B_w - R_{sw}B_g)\left[\nabla \cdot \vec{k}\,\frac{\lambda_w}{B_w}\nabla P_o + CG_w - \frac{q_w}{r_{wsc}}\right]$$

$$+ B_g\left[\nabla \cdot \vec{k}\left(\frac{\lambda_g}{B_g} + \frac{R_{so}\lambda_o}{B_o} + \frac{R_{sw}\lambda_w}{B_w}\right)\nabla P_o + CG_g - \frac{q_g}{r_{gsc}}\right]$$

$$= \phi C_t \frac{\partial P_o}{\partial t} \tag{5.88}$$

where

$$CG_o = -\nabla \cdot \left(\hat{\lambda}_o \cdot \nabla \frac{r'_o Z_o}{144}\right) \tag{5.89}$$

$$CG_w = -\nabla \cdot \left(\hat{\lambda}_w \cdot \nabla \frac{r'_w Z}{144} + P_{cow}\right) \tag{5.90}$$

and

$$CG_g = \nabla \cdot \left(\hat{\lambda}_g \cdot \nabla P_{cgo} - \frac{r'_g Z_o}{144}\right) - R_{so}\hat{\lambda}_o \nabla \frac{r'_o Z_o}{144}$$

$$-R_{sw}\hat{\lambda}_w \nabla P_{cow} + \frac{r'_w Z_o}{144} \tag{5.91}$$

We assume that the capillary pressure for oil-to-water P_{cow} is a function of the water saturation only, and the capillary pressure for gas-to-oil P_{cgo} is a function of the gas saturation only. Then:

$$\nabla P_{cow} = \hat{i}\frac{\partial P_{cow}}{\partial x} + \hat{j}\frac{\partial P_{cow}}{\partial y} + \hat{k}\frac{\partial P_{cow}}{\partial z} \tag{5.92}$$

$$= \hat{i}\frac{\partial S_w}{\partial x} + \hat{j}\frac{\partial S_w}{\partial y} + \hat{k}\frac{\partial S_w}{\partial z}\frac{dP_{cow}}{dS_w} \tag{5.93}$$

or

$$\nabla P_{cow} = \frac{dP_{cow}}{dS_w}\nabla S_w \tag{5.94}$$

and

$$\nabla P_{cgo} = \hat{i}\frac{\partial P_{cgo}}{\partial x} + \hat{j}\frac{\partial P_{cgo}}{\partial y} + \hat{k}\frac{\partial P_{cgo}}{\partial z} \tag{5.95}$$

$$= \hat{i}\frac{\partial S_g}{\partial x} + \hat{j}\frac{\partial S_g}{\partial y} + \hat{k}\frac{\partial S_g}{\partial z}\frac{dP_{cgo}}{dS_g} \tag{5.96}$$

or

$$\nabla P_{cgo} = \frac{dP_{cgo}}{dS_g}\nabla S_g \tag{5.97}$$

Equation (5.88) is the pressure equation. Our approach solves three-dimensional triphasic flow by implicitly solving equation (5.88) for P_o, then explicitly solving equations (5.61), (5.65), and (5.46) for the phase saturations at each time step. The Fortran source code of BOAST II and the computational implementation of this IMPES solution is presented in Appendix A.

5.2 Finite Element Formulation

In the finite element method, trial functions composed of a linear combination of shape functions are used to approximate a solution for the problem domain's pressure and saturation. The shape functions $N_j^{(e)}$ are initially localized over single elements and then these trial functions are joined together with trial functions of adjacent elements in a process called assembly. In the approximation of pressure for each element $P_o^{(e)}$ (5.92), of saturation for each element $S_o^{(e)}$ (5.93), and $S_w^{(e)}$ (5.94) a linear combination of shape functions are used but all of the independent variables are not in the shape functions. Using the classical separation of variables technique, the shape functions are constructed to be dependent upon the spacial variables and their coefficients to be dependent on time. The approximations for $P_o^{(e)}$, $S_o^{(e)}$ and $S_w^{(e)}$ follow.

The element trial solution for the oil pressure:

$$P_o^{(e)} \gg \sum_{j=1}^{n} P_j^{(e)}(t) N_j^{(e)}(x, y, z) = \sum_{j=1}^{n} P_j N_j \tag{5.98}$$

The element trial solution for the oil saturation:

$$S_o^{(e)} \approx \sum_{j=1}^{n} S_{oj}^{(e)}(t) N_j^{(e)}(x, y, z) = \sum_{j=1}^{n} S_{oj} N_j \tag{5.99}$$

The element trial solution for the water saturation:

$$S_w^{(e)} \approx \sum_{j=1}^{n} S_{wj}^{(e)}(t) N_j^{(e)}(x, y, z) = \sum_{j=1}^{n} S_{wj} N_j \tag{5.100}$$

The element trial solution for the gas saturation:

$$S_g^{(e)} \approx \sum_{j=1}^{n} S_{gj}^{(e)}(t) N_j^{(e)}(x, y, z) = \sum_{j=1}^{n} S_g N_j \tag{5.101}$$

The same trial functions are used for the fluid density but these have coefficients which depend upon the depth (z). The approximation for ρ_o is:

$$\rho_o \approx \sum_{j=1}^{n} (\rho_o z)_j N_j^{(e)}(x, y, z) \tag{5.102}$$

where the $N_j^e(x, y, z)$ are the shape functions and n is the number of degrees of freedom (DOF) in each element (hexahedral elements are used).

It should be noted that our pressure and saturation equations are highly non-linear and are coupled. Each solution of these equations will result in a distribution

of the pressure P_o, oil saturation S_o, and water saturation S_w for the problem's domain at a given time. Then the problem is solved step by step in time space.

The discretization of a problem in the finite element method begins with an element formulation for each equation as follows.

Conductance Term:

$$\int_{v}^{(e)} (\nabla \cdot (\lambda_o \cdot \nabla P_o)) N_i dV \tag{5.103}$$

where

$$\nabla = \vec{i}\frac{\partial}{\partial x} + \vec{j}\frac{\partial}{\partial y} + \vec{k}\frac{\partial}{\partial z} \tag{5.104}$$

$$\nabla P_o = \sum_{j=1}^{n} P_j^{(e)} \nabla N_j \tag{5.105}$$

Using Green's Theorem:

$$\int_{v}\int\int (f\nabla^2 g + \nabla f \cdot \nabla g) dV = \int_{s}\int f \frac{\partial g}{\partial n} dA \tag{5.106}$$

$$\int_{v}\int\int (f\nabla^2 g) dV = -\int_{v}\int\int (\nabla f \cdot \nabla g) dV + \int_{s}\int f \frac{\partial g}{\partial n} dA \tag{5.107}$$

Green's Theorem applied to the conductance term:

$$\int_{v}^{(e)} \nabla \cdot (\lambda_o \cdot \nabla P_o) N_i dV = -\int_{v}^{(e)} (\lambda_o \cdot \nabla P_o) \cdot \nabla N_i dV$$

$$+ \int_{s}^{(e)} (\hat{l}_o \cdot \hat{\nabla} P_o) \cdot \hat{n} N_i ds \tag{5.108}$$

where $\vec{n}$ is the unit outer normal vector to the surface S of an element. Observe that $\hat{l}_o \cdot \hat{\nabla} P_o$ is the oil flux, and we represent this oil flux normal to the surface S as $t_o^{(e)}$:

$$t_o^{(e)} = (\hat{l}_o \cdot \hat{\nabla} P_o) \cdot \vec{n} \tag{5.109}$$

$$\int_{s}^{(e)} t_o^{(e)} N_i ds = t_o^{(e)} \int_{s}^{(e)} N_i ds \tag{5.110}$$

For sufficiently small time steps and elements, we assume that the flux on each face in an element e is constant.

For a quadrilateral face of a hexahedral:

$$t_o^{(e)} \int_{s}^{(e)} N_i ds = \frac{t_o^{(e)_A}}{4} \tag{5.111}$$

That is 1/4 of the flux is assigned to each node of the quadrilateral face of the hexahedral.

Then assuming a medium with no mixed permeability terms $(k_x, k_y, and\ k_z)$ produces:

$$-\int_{v}^{(e)} (\hat{l}_o \cdot \hat{\nabla} P_o) \cdot \hat{\nabla} N_i dV = -\int_{v}^{(e)} \sum_{j=1}^{n} (\hat{l}_o P_j \cdot \hat{\nabla} N_j) \cdot \hat{\nabla} N_i dV \tag{5.112}$$

$$= -\sum_{j=1}^{n} \left[\int_{v}^{(e)} [(\hat{l}_o \cdot \hat{\nabla} N_j) \cdot \hat{\nabla} N_i] dV \right] P_j \tag{5.113}$$

$$= -\frac{1}{B_o \mu_o} \sum_{j=1}^{n} \int_{v}^{(e)} \left[(K_{ro})_x \frac{\partial N_j}{\partial x} \hat{i} + (K_{ro})_y \frac{\partial N_j}{\partial y} \hat{j} \right.$$

$$\left. + (K_{ro})_z \frac{\partial N_j}{\partial z} \hat{k} \right] \cdot \left[\hat{i} \frac{\partial N_i}{\partial x} + \hat{j} \frac{\partial N_i}{\partial y} + \hat{k} \frac{\partial N_i}{\partial z} \right] dV P_j$$

$$-\int_{v}^{(e)} (\hat{l}_o \cdot \hat{\nabla} P_o) \cdot \hat{\nabla} N_i dV \tag{5.114}$$

$$= -\frac{1}{B_o \mu_o} \sum_{j=1}^{n} P_j \int_{v}^{(e)} \left[(K_{ro})_x \frac{\partial N_j}{\partial x} \frac{\partial N_i}{\partial x} \right.$$

$$\left. + (K_{ro})_y \frac{\partial N_j}{\partial y} \frac{\partial N_i}{\partial y} + (K_{ro})_z \frac{\partial N_j}{\partial z} \frac{\partial N_i}{\partial z} \right] dV \tag{5.115}$$

let,

$$\boxed{A_{ij}^{(e)} = \int_{v}^{(e)} -\frac{1}{B_o \mu_o} \left[(K_{ro})_x \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + (K_{ro})_y \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} + (K_{ro})_z \frac{\partial N_i}{\partial z} \frac{\partial N_j}{\partial z} \right] dV} \tag{5.116}$$

Gravity Term:

$$-\int_{v}^{(e)}\left[\vec{N}\cdot\left.\vec{\nabla}\right|_{o}\cdot\vec{N}\sum_{j=1}^{n}\frac{(r'_{o}z_{j})N_{j}}{144}\right]N_{i}dV=-\sum_{j=i}^{n}\frac{A_{ij}^{(e)}[r'_{o}z]_{j}}{144} \tag{5.117}$$

let:

$$D_{oi}=-\frac{r'_{o}}{144}\frac{(K_{ro})_{z}}{B_{o}\mu_{o}}\int_{v}^{(e)}\frac{\partial N_{i}}{\partial z}dV \tag{5.118}$$

Injection/Production term:

$$Q_{oi}^{(e)}=-\int_{v}^{(e)}\frac{q_{o}}{r_{o}}N_{i}dV \tag{5.119}$$

We can either handle all injection/production with the flux terms or choose to use (5.119) as shown.

Saturation Capacity Term:

$$\int_{v}^{(e)}\frac{1}{B_{o}}\frac{\partial S_{w}}{\partial t}N_{i}dV,\frac{1}{B_{o}}\int_{v}^{(e)}\sum_{j=1}^{n}\frac{\partial S_{wj}}{\partial t}N_{j}N_{i}dV,\frac{1}{B_{o}}\sum_{j=1}^{n}\left[\frac{\partial S_{wj}}{\partial t}\int_{v}^{(e)}N_{j}N_{i}dV\right] \tag{5.120}$$

$$\int_{v}^{(e)}\frac{1}{B_{o}}\frac{\partial S_{g}}{\partial t}N_{i}dV,\frac{1}{B_{o}}\sum_{j=1}^{n}\left[\frac{\partial S_{gj}}{\partial t}\int_{v}^{(e)}N_{j}N_{i}dV\right] \tag{5.121}$$

$$\boxed{C_{ij}^{o},\frac{1}{B_{o}}\int_{v}^{(e)}N_{i}N_{j}dV} \tag{5.122}$$

Compressibility Term:

$$\sum_{v}^{(e)} -\frac{1}{B_o}\left[C_r - \frac{1}{B_o}\frac{{}^{1}B_o}{{}^{1}P_o}\right]\left[\frac{{}^{1}P_o}{{}^{1}t} - S_w\frac{{}^{1}P_o}{{}^{1}t} - S_g\frac{{}^{1}P_o}{{}^{1}t}\right]N_i dV \tag{5.123}$$

let:

$$a_o = \left[C_r - \frac{1}{B_o}\frac{{}^{1}B_o}{{}^{1}P_o}\right] \tag{5.124}$$

The following saturation trial functions are to be solved at the k^{th} timestep or the previous timestep represented by the k superscript:

$$S_w^{(e)} = \sum_{l=1}^{n} (S_w^k)_l N_l \tag{5.125}$$

$$S_g^{(e)} = \sum_{l=1}^{n} (S_g^k)_l N_l \tag{5.126}$$

$$\frac{{}^{1}P_o}{{}^{1}t} = \sum_{j=1}^{n} \left(\frac{{}^{1}\bar{P}_o}{{}^{1}t\phi_j}\right) N_j \tag{5.127}$$

$$-\frac{a_o}{B_o}\sum_{v}^{(e)}\left[\frac{{}^{1}P_o}{{}^{1}t} - S_w\frac{{}^{1}P_o}{{}^{1}t} - S_g\frac{{}^{1}P_o}{{}^{1}t}\right]N_i dV =$$

$$-\frac{a_o}{B_o}\left[\sum_{v}^{(e)}\sum_{j=1}^{n}\left(\frac{{}^{1}\bar{P}_o}{{}^{1}t\phi_j}\right)N_j N_i dV\right]$$

$$-\left[\sum_{v}^{(e)}\sum_{l=1}^{n}(S^k{}_g)_l N_l \sum_{j=1}^{n}\left(\frac{{}^{1}\bar{P}_o}{{}^{1}t\phi_j}\right)N_j N_i dV\right]$$

$$-\left[\int_{v}\sum_{l=1}^{n}(S_w^k)_l N_l\right]\left\{\sum_{j=1}^{n}\frac{\partial P_o}{\partial t}\frac{\partial}{\partial \theta_j}N_j\right\}N_i dV$$

$$-\sum_{j=1}^{n}\frac{\partial P_o}{\partial t}\frac{\partial}{\partial \theta_j}\left\{\sum_{l=1}^{n}\left[(S^k{}_g)_l\int_{v}N_i N_j N_l dV\right]\right\}\theta \tag{5.128}$$

$$+\frac{\alpha_o}{B_o}\sum_{j=1}^{n}\frac{\partial P_o}{\partial t}\frac{\partial}{\partial \theta_j}\int_{v}N_i N_j dV$$

$$-\sum_{j=1}^{n}\frac{\partial P_o}{\partial t}\frac{\partial}{\partial \theta_j}\left\{\sum_{l=1}^{n}\left[(s^k{}_w)_l\int_{v}N_i N_j N_l dV\right]\right\}\theta \tag{5.129}$$

Define:

$$f_2(i,j)=\int_{v}N_i N_j dV \tag{5.130}$$

and

$$f_3(i,j,l)=\int_{v}N_i N_j N_l dV \tag{5.131}$$

Then:

$$-\frac{\alpha_o}{B_o}\int_{v}\left[\frac{\partial P_o}{\partial t}-S_w\frac{\partial P_o}{\partial t}-S_g\frac{\partial P_o}{\partial t}\right]N_i^{(e)}dV$$

$$+\frac{\alpha_o}{B_o}\sum_{j=1}^{n}\frac{\partial P_o}{\partial t}\frac{\partial}{\partial \theta_j}\left[\sum_{l=1}^{n}\left[(S_w^k)_l+(S_g^k)_l\right]f_3(i,j,l)\right]$$

$$-\sum_{j=1}^{n}\frac{\partial P_o}{\partial t}\frac{\partial}{\partial \theta_j}f_2(i,j)$$

$$= \frac{\rho_o}{B_o} \sum_{j=1}^{n} \left[\sum_{l=1}^{n} \left[(S_w^k)_l + (S_g^k)_l \right] f_3(i,j,l) \frac{\partial}{\partial t} - f_2(i,j) \right] \frac{\partial P}{\partial t} \theta_j \tag{5.132}$$

$$CO_{ij}^{(e)} \cdot \frac{\rho_o}{B_o} \left[\sum_{l=1}^{n} \left[(S_w^k)_l + (S_g^k)_l \right] f_3(i,j,l) \frac{\partial}{\partial t} - f_2(i,j) \right] \tag{5.133}$$

The element stiffness matrix components for the oil phase using a dot to represent the partial derivative with respect to time:

$$\boxed{A_{ij}^{(e)} P_j^{k+1} + CO_{ij}^{(e)} \dot{P}_j + C_{ij}^{\cdot} (\dot{S}_w + \dot{S}_g)_j + Q_{oi}^{(e)} + D_{oi}^{(e)} = 0} \tag{5.134}$$

The element formulation for the water phase begins as follows:

Conductance Term:

$$\int_{v}^{(e)} \nabla \cdot (\hat{l}_w \cdot \nabla P_o) N_i dV =$$

$$- \int_{v}^{(e)} (\hat{l}_w \cdot \nabla P_o) \cdot \nabla N_i dV + \int_{s}^{(e)} (\hat{l}_w \cdot \nabla P_o) \cdot \vec{n} N_i ds \tag{5.135}$$

$$t_w^{(e)} = (\hat{l}_w \cdot \nabla P_o) \cdot \vec{n} \tag{5.136}$$

$$\int_{s}^{(e)} t_w^{(e)} N_i ds \, , \, t_w^{(e)} \int_{s}^{(e)} N_i ds \, , \, \frac{t_w^{(e)} A}{4} \tag{5.137}$$

$$-\dot{o}(\hat{I}_w \cdot \tilde{N}P_o) \cdot \tilde{N}N_i dV \,,$$

$$-\frac{1}{B_w m_w} \overset{n}{\underset{j=1}{\dot{o}}} P_j \overset{(e)}{\underset{v}{\dot{o}}} \left[(K_{rw})_x \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} \right.$$

$$\left. + (K_{rw})_y \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + (K_{rw})_z \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right] dV \qquad (5.138)$$

$$\boxed{H_{ij}^{(e)} = -\frac{1}{B_w m_w} \overset{(e)}{\underset{v}{\dot{o}}} \left[(K_{rw})_x \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + (K_{rw})_y \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + (K_{rw})_z \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right] dV} \qquad (5.139)$$

Gravity Term:

$$-\dot{o}\tilde{N} \cdot \overset{\text{æ}}{\underset{\text{è}}{\text{ç}}}\hat{I}_w \cdot \tilde{N} \overset{\text{æ}}{\underset{\text{è}}{\text{ç}}} \frac{r_w Z}{144} \overset{\text{ö}}{\underset{\text{ø}}{\text{÷}}} \overset{\text{ö}}{\underset{\text{ø}}{\text{÷}}} N_i dV = \overset{n}{\underset{j=1}{\dot{o}}} H_{ij}^{(e)} \left[\frac{r_w Z}{144} \right]_j \qquad (5.140)$$

$$-\dot{o}\tilde{N} \cdot \overset{\text{æ}}{\underset{\text{è}}{\text{ç}}}\hat{I}_w \cdot \tilde{N} \overset{\text{æ}}{\underset{\text{è}}{\text{ç}}} \frac{r_w Z}{144} \overset{\text{ö}}{\underset{\text{ø}}{\text{÷}}} \overset{\text{ö}}{\underset{\text{ø}}{\text{÷}}} N_i dV = D_{wi}^{(e)} \qquad (5.141)$$

Injection/Production Term:

$$\boxed{-\overset{(e)}{\underset{v}{\dot{o}}} \frac{q_w}{r_w} N_i dV = Q_{wi}^{(e)}} \qquad (5.142)$$

The oil saturation equation is:

$$\frac{B_o \Delta t}{i}\left[[A_{ij}](P_o)_j^{m+1} + D_{oj} + Q_{oj} \right] + S_{oj}^{m} =$$

$$S_{oj}^{m+1}\left[1+\left(C_r-\frac{1}{B_o}\frac{dB_o}{dP_o}\right)\left(P_o\right)_j^{m+1}-\left(P_o\right)_j^{m}\right] \tag{5.143}$$

Similarly, the water saturation equation is:

$$\frac{B_w \Delta t}{}\left[\left[H_{ij}\right]\left(P_o\right)_j^{m+1}+D_{wj}+Q_{wj}\right]+S_{wj}^{m}=$$

$$S_{wj}^{m+1}\left[1+\left(C_r-\frac{1}{B_w}\frac{dB_w}{dP_o}\right)\left(P_o\right)^{m+1}-\left(P_o\right)_j^{m}\right] \tag{5.144}$$

Using (5.54); the element formulation for the gas phase begins as follows:

Conductance Term is:

$$\int_v^{(e)} \nabla\cdot\left(\lambda_g\cdot\nabla P_o\right)N_i dV=-\int_v^{(e)}\left(\lambda_g\cdot\nabla P_o\right)\cdot\nabla N_i dV + \int_s^{(e)}\left(\lambda_g\cdot\nabla P_o\right)\cdot\vec{n}N_i dS \tag{5.145}$$

$$\lambda_g^{(e)}=\left(\lambda_g\cdot\nabla P_o\right)\cdot\vec{n} \tag{5.146}$$

$$\lambda_g^{(e)}\int_s^{(e)} N_i dS=\frac{\lambda_g^{(e)}A}{4} \tag{5.147}$$

$$-\int_v^{(e)}\left(\lambda_g\cdot\nabla P_o\right)\cdot\nabla N_i dV=\sum_{j=1}^{n}\int_v^{(e)}\left[\left(\lambda_g\cdot\nabla N_j\right)\cdot\nabla N_i\right]dV\,P_j \tag{5.148}$$

$$=-\frac{1}{B_g\mu_g}\sum_{j=1}^{n}P_j\int_v^{(e)}\left(K_{rg}\right)_x\frac{\partial N_j}{\partial x}\frac{\partial N_i}{\partial x}$$

$$+(K_{rg})_y \frac{\partial N_j}{\partial y}\frac{\partial N_i}{\partial y} + (K_{rg})_z \frac{\partial N_j}{\partial z}\frac{\partial N_i}{\partial z}\Big] dv \tag{5.149}$$

let

$$\boxed{E_{ij}^{(e)} = \int_v^{(e)} \left[-\frac{1}{B_g \mu_g} \left((K_{rg})_x \frac{\partial N_j}{\partial x}\frac{\partial N_i}{\partial x} + (K_{rg})_y \frac{\partial N_j}{\partial y}\frac{\partial N_i}{\partial y} + (K_{rg})_z \frac{\partial N_j}{\partial z}\frac{\partial N_i}{\partial z} \right) \right] dV} \tag{5.150}$$

Capillary Term:

$$\int_v^{(e)} \frac{dP_{cgo}}{dS_g} [\nabla \cdot (\vec{i}_g \cdot \nabla S_g)] N_i dV = \frac{dP_{cgo}}{dS_g} \int_v^{(e)} (\nabla \cdot (\vec{i}_g \cdot \nabla S_g)) N_i dV \tag{5.151}$$

$$= -\frac{dP_{cgo}}{dS_g} \int_v^{(e)} (\vec{i}_g \cdot \nabla S_g) \cdot \nabla N_i dV + \frac{dP_{cgo}}{dS_g} \int_s^{(e)} (\vec{i}_g \cdot \nabla S_g) \vec{n} dS \tag{5.152}$$

as before,

$$-\frac{dP_{cgo}}{dS_g} \int_v^{(e)} (\vec{i}_g \cdot \nabla S_g) \cdot \nabla N_i dV$$

$$= -\frac{dP_{cgo}}{dS_g} \sum_{j=1}^{n} \frac{(S_g)_j}{B_g \mu_g} \int_v^{(e)} \left[(K_{rg})_x \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} \right.$$

$$\left. + (K_{rg})_y \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + (K_{rg})_z \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right] dV \tag{5.153}$$

$$\boxed{G_{ij}^{(e)} = \int_v^{(e)} -\frac{dP_{cgo}}{dS_g B_g \mu_g} \left[(K_{rg})_x \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + (K_{rg})_y \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + (K_{rg})_z \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z} \right] dV} \tag{5.154}$$

Saturation Capacity Term is:

$$-\int_{v}^{(e)} \frac{1}{B_g}\frac{\partial S_g}{\partial t} N_i dV = -\sum_{j=1}^{n} \int_{v}^{(e)} \frac{1}{B_g}\frac{\partial S_g}{\partial t} N_j N_i dV \tag{5.155}$$

Let,

$$f_g = -\int_{v}^{(e)} \frac{1}{B_g} N_j N_i dV \tag{5.156}$$

Gravity Term is:

$$-\int_{v}^{(e)} \nabla \cdot \frac{k_g}{144}\rho_g \cdot \nabla \frac{k_g}{144} N_i dV = -\sum_{j=1}^{n} E_{ij}^{(e)} \left[\frac{k_g Z}{144}\right]_j = D_{gi}^{(e)} \tag{5.157}$$

Injection/Production Term is:

$$\boxed{-\int_{v}^{(e)} \frac{q_{fg}}{\rho_g} N_i dV = Q_{gi}^{(e)}} \tag{5.158}$$

Finally, using the finite element equations that we have developed for three-dimensional three-phase fluid flow, the IMPES pressure equation is:

$$(B_o - R_{so}B_g)\left[\nabla \cdot \vec{k} \cdot \frac{k_o}{B_o}\nabla P_o + CG_o - \frac{q_o}{\rho_{osc}}\right]$$

$$+ (B_w - R_{sw}B_g)\left[\nabla \cdot \vec{k} \cdot \frac{k_w}{B_w}\nabla P_o + CG_w - \frac{q_w}{\rho_{wsc}}\right]$$

$$+B_g\left[\nabla\cdot\vec{k}\,\frac{k_{rg}}{B_g}+\frac{R_{so}k_{ro}}{B_o}+\frac{R_{sw}k_{rw}}{B_w\,\phi}\nabla P_o+CG_g-\frac{q_g}{r_{gsc}}\right]$$

$$=jC_t\frac{\partial P_o}{\partial t} \tag{5.159}$$

After substituting and reducing terms, (5.160) becomes:

$$(B_o-R_{so}B_g)\left[A_{ij}^{(e)}P_j^{n+1}+D_{oi}^{(e)}+Q_{oi}^{(e)}\right]$$

$$+(B_w-R_{sw}B_g)\left[H_{ij}^{(e)}P_j^{n+1}+D_{wi}^{(e)}+Q_{wi}^{(e)}\right]$$

$$+B_g\left[E_{ij}^{(e)}P_j^{n+1}+R_{so}A_{ij}^{(e)}P_j^{n+1}+R_{sw}H_{ij}^{(e)}\,P_j^{n+1}\right]$$

$$+B_gD_{gi}+B_gQ_{gi}=jC_t\left[\frac{P_j^{n+1}-P_j^{n}}{\Delta t}\right] \tag{5.160}$$

Collect terms,

$$j\Delta t\left[(B_o-R_{so}B_g)A_{ij}^{(e)}+(B_w-R_{sw}B_g)H_{ij}^{(e)}\right.$$

$$\left.+B_gE_{ij}^{(e)}+B_gR_{so}A_{ij}^{(e)}+R_{sw}H_{ij}^{(e)}\right]-jC_t\delta_{ij}P_j^{n+1}$$

$$=\Delta t(B_o-R_{so}B_g)\left[D_{oi}^{(e)}+Q_{oi}^{(e)}\right]$$

$$+(B_w-R_{sw}B_g)[\Delta t]\left(D_{wi}^{(e)}+Q_{wi}^{(e)}\right)$$

$$+ B_g \Delta t [D_{gi}^{(e)} + Q_{gi}^{(e)}] - \sum C_t P_j^n \tag{5.161}$$

After simplification; the IMPES pressure equation becomes:

$$\Delta t \sum_i B_o A_{ij} + B_w H_{ij} + B_g E_{ij}^{(e)} - \sum_D C_t P_j^{n+1} = fl_1 - \sum C_t P_j^n \tag{5.162}$$

where,

$$fl_1 = \Delta t (B_o - R_{so} B_g)[D_{oi}^{(e)} + Q_{oi}^{(e)}]$$

$$+ (B_w - R_{sw} B_g)[\Delta t](D_{wi}^{(e)} + Q_{wi}^{(e)}) + B_g \Delta t [D_{gi}^{(e)} + Q_{gi}^{(e)}] \tag{5.163}$$

To write the stiffness matrix in a compact form.

$$[X] = \sum_i B_o A_{ij} + B_w H_{ij} + B_g E_{ij}^{(e)} - \sum_D C_t \Delta t \tag{5.164}$$

and

$$[fl^{(e)}] = fl_1 - \sum C_t P_j^n \tag{5.165}$$

then

$$[X][P^{(e)}] = [fl^{(e)}] \tag{5.166}$$

or

$$[p^{(e)}] = [X]^{-1}[fl^{(e)}] \tag{5.167}$$

The size of the stiffness matrix $[X]$ depends upon the number of nodal points in the mesh, which has a major impact on the accuracy of the solution. This is the pressure equation that we solve for implicitly.

CHAPTER 6. COMPUTATIONAL MODEL

An available public reservoir simulation model, BOAST II (Black Oil Applied Simulation Tool), was modified to better simulate the conditions in the vicinity of the wellbore. The U.S. Department of Energy released the original black oil model named BOAST in 1982. BOAST II was released in 1987 to provide more flexibility and to overcome some limitations of the original model. Our research uses BOAST II as the reservoir simulator to demonstrate our results. Often we refer to BOAST II simply as "BOAST." We recognize that this reservoir simulator has several drawbacks, some of which are:

- BOAST II is based on the finite difference model, which divides the entire "reservoir" of interest into blocks. Its mechanisms to simulate areas of high activity are not accurate. For production in the vicinity of the wellbore, we added our finite element code, creating more nodes in this region where most activity occurs, especially within the first 20 to 30 feet of the wellbore.
- BOAST II calculates one average pressure for each mesh block in the reservoir. To model the highly changing region in the vicinity of the wellbore more accurately, the finite element equations from Chapter 5 where interfaced to demonstrate and produce better results for this region. Then for each timestep, in the simulation after the FEM pressures and saturations were computed at nodes in the well block, the FEM pressures and saturations obtained are averaged and inserted into BOAST II to improve the average values for the well block.

- A major problem with BOAST II is that it uses IMPES without any iteration to solve for pressures and saturations at each timestep. The BOAST II results for some simulations are well known to be in error and the problem has been identified as IMPES. That is BOAST II answers have been compared to other reservoir simulators that use fully implicit solutions and this identified the problem.

We are interested in better simulation of the activity in the wellbore vicinity, thus our FEM model of this activity spends its computational time accurately simulating the flow within this region. BOAST II does not support the simulation of the wellbore vicinity region and our FEM routines where integrated into BOAST II to allow the study of the wellbore vicinity region. Earlier work by our research group had developed fully-implicit finite element (FEM) equations for two phases and two dimensions. This research simulates three phases (oil, water and gas) for three dimensions using an implicit pressure / explicit saturation (IMPES) solution for our FEM equations.

The IMPES method implicitly solves for the pressure distribution at each time step, then uses this pressure distribution to explicitly solve for the saturations at the same time step. IMPES requires less computation per time step; hence it is faster than a fully implicit solution for the same problem. IMPES also requires less storage than the fully implicit formulation. However, the solution obtained by IMPES is not as stable as the fully implicit method for many simulations. This is specially true in simulations in which there are rapid changes in saturations that

result from high flux rates. Our model was sensitive to this problem but its effect was reduced by either adaption of the mesh or a reduction in the size of the timestep.

6.1 System Requirements

Our FEM well vicinity model can be interfaced or adapted into other simulators for which we have the source code. The FEM well vicinity model is written in Fortran 77. The well vicinity model was designed to be modular so that important routines, e.g., the mesh generation routine MESHGEN, can be easily replaced with one that includes dynamic adaptive meshing. No external libraries are required, i.e., it is self-contained. The input datafile for BOAST II is used and the formats shown in Appendix B and Appendix C are examples of this.

Our FEM well vicinity model was designed to use the BOAST II data without any additional input data except adding the location of the well block to be modeled.

6.2 Simulation Process

Reservoir simulators are engineering tools that require some training before use. Reservoir simulators provide many useful estimates for reservoirs and well models. These estimates, together with economic evaluations, are used to make improved management and field decisions.

The major steps in a simulation are:

- Input data gathering
- Simulation runs

- History matching
- Reservoir predictions

6.3 Input Data Requirements

All the necessary input data for one execution of the BOAST II simulator are contained in a single file. A complete description of all possible input data is contained in the BOAST II manual [5].

6.4 Coupling of Well Vicinity and Reservoir Simulators

After BOAST II establishes the reservoir being simulated (initializes its tables, arrays, etc.) and completes the simulation of the first time step, our FEM well vicinity model begins its execution as shown on Figure 6.1.

BOAST II calculates one average pressure and one average saturation for each phase for the "center" of each block in the reservoir for each stratum. The wellbore vicinity model uses these average values from BOAST II as input and further processing. The first time our model is called after BOAST II completes the simulation of the first time-step, it generates the finite element mesh to be used, initializes each nodal pressure and each nodal saturation for all phases in this finite element mesh.

All the initial values for our model are obtained from BOAST II, e.g., reservoir properties (permeability, porosity, etc.), reservoir pressure, saturations, etc. These are used as the initial value to estimate behavior within the well vicinity. After each FEM computation (each timestep), the FEM model computes an average

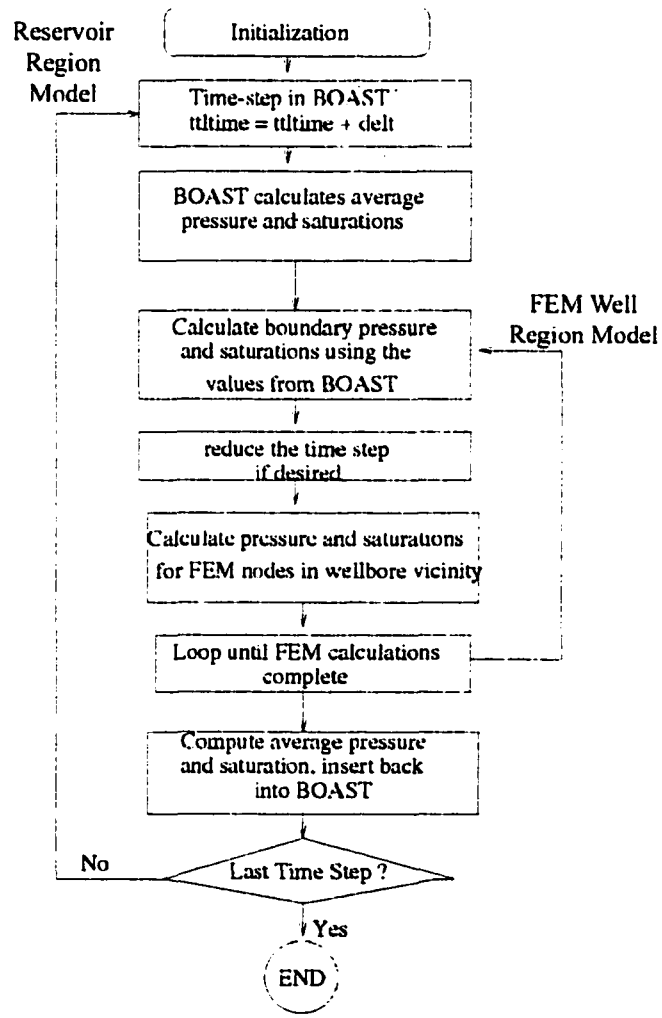


Figure 6.1: Interfacing our FEM model with BOAST II

for the FEM nodal values of pressure and each saturation, and inserts into BOAST II a single value for each of these per stratum for use in the next BOAST II time step in the simulation.

The pressure and saturations at each node in the wellbore vicinity are initialized to the same values for the BOAST II wellblock. After this initialization, all nodal pressures and saturations computed with the wellbore vicinity model are stored and kept for further use when the FEM wellbore vicinity model is called again.

To interface our wellbore vicinity model the following transitional mesh was used at the boundary of the wellblock for the three layered problem:

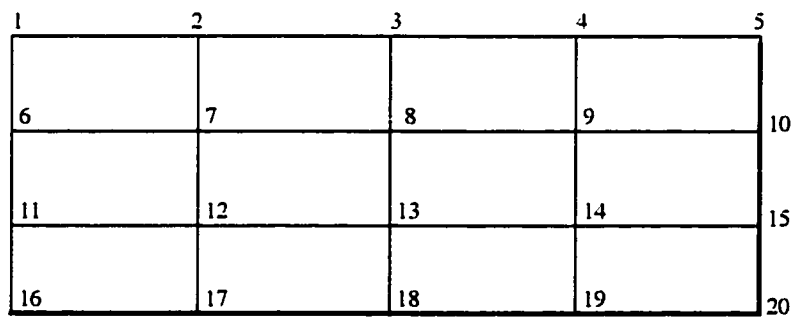


Figure 6.2: Wellbore Boundary Interfacing with BOAST II

Figures 6.2 and 6.3 show the FEM nodes per layer at each of the vertical four sides of the boundary of the wellbore vicinity model. The flow rate per boundary node at

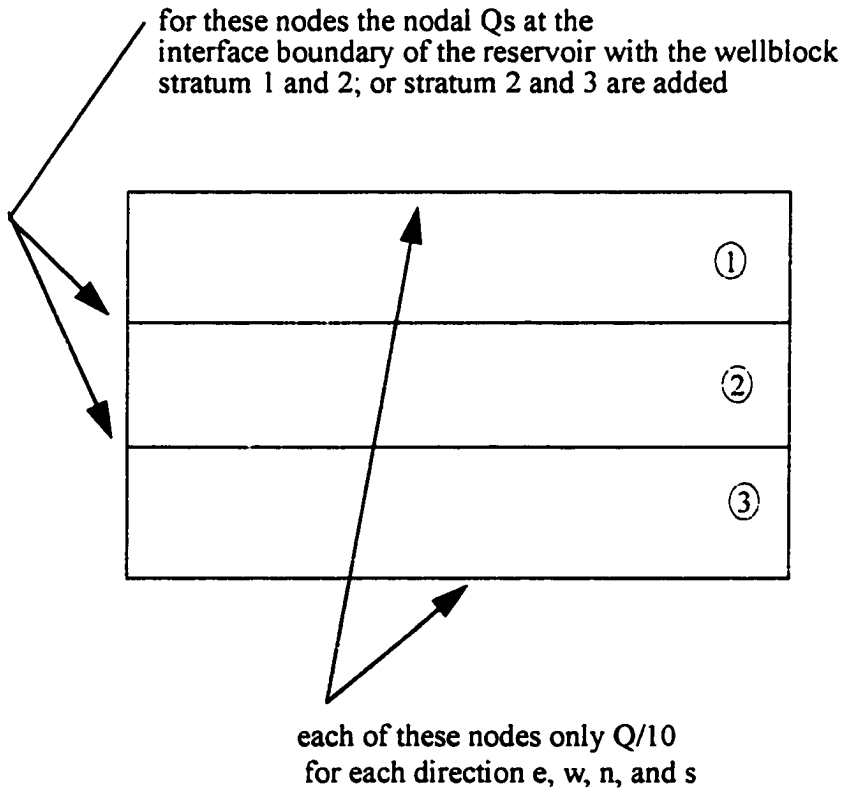


Figure 6.3: Determination of Nodal Fluid Distribution at Wellblock Interface

the interface was computed by using values from BOAST II and distributing these values as:

Q(layer#1 from BOAST II) / 10 =

Q_1 per node at wellbore interface for layer#1

Q(layer#2 from BOAST II) / 10 =

Q_2 per node at wellbore interface for layer#2

Q(layer#3 from BOAST II) / 10 =

Q_3 per node at wellbore interface for layer#3

The nodal values for the interface flow rate are determined in our FEM model as follows:

q (node 1,2,3,4, or 5) = Q_1

q (node 6,7,8,9, or 10) = $Q_1 + Q_2$

q (node 11,12,13,14, or 15) = $Q_2 + Q_3$

q (node 16,17,18,19, or 20) = Q_3

for each vertical interface surface.

6.5 Wellblock Nodal Pressure Distribution

This 2-D areal view for one stratum (k) of the wellblock using BOAST II indices (i,j,k) is shown on Figure 6.4. In BOAST II a wellblock is identified by selecting one value for the indices i and j and allowing k to range over all its values (one value of k for each stratum). In BOAST II the block containing the well shown in Figure 6.4 is (i,j,k) and the FEM interface nodes that were used are shown as

nodes 1 through 8. Nodes 1, 3, 6, and 8 are at the corners of the (i,j,k) block and nodes 2, 4, 5, and 7 are at its mid points also shown on Figure 6.4.

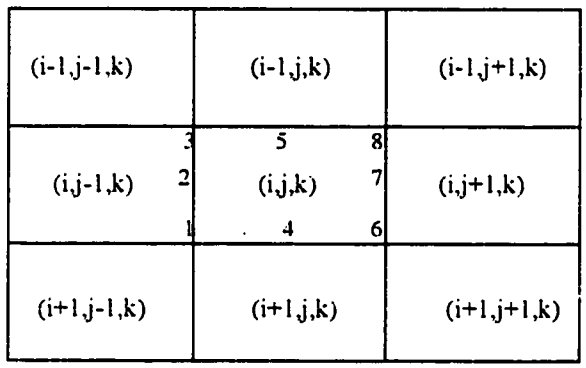


Figure 6.4: Wellblock with Indices (i,j,k)

For Case 1 with the well located in reservoir block, i=6, j=5, k=1, or block (6,5,1), the BOAST II-FEM interface pressure to be used for node 6 for example, at each timestep, is calculated as follows:

$$R_1^2 = \frac{[dx(i,j,k)]^2}{4} + \frac{[dy(i,j,k)]^2}{4} \tag{6.1}$$

$$R_2^2 = \frac{[[dx(i,j,k)] + [dx(i+1,j+1,k)]]^2}{4} \tag{6.2}$$

$$+ \frac{[[dy(i,j,k)] + [dy(i+1,j+1,k)]]^2}{4} \tag{6.3}$$

Then,

$$\frac{P_6 - P(i,j,k)}{\log R_1} = \frac{P(i+1,j+1,k) - P(i,j,k)}{\log R_2} \tag{6.4}$$

simplifying,

$$P_6 = [P(i+1,j+1,k) - P(i,j,k)]\left[\frac{\log R_1}{\log R_2}\right] + P(i,j,k) \tag{6.5}$$

where the "*P(i,j,k)*'s" are the pressures, the "*dx(i,j,k)*'s" and "*dy(i,j,k)*'s" are the lengths of "*x* and *y*" for each block calculated by BOAST II at each timestep. The nodal interface pressures at other nodes shown on Figure 6.4 are calculated in an analogous manner. Any additional nodal interface pressures are obtained by a linear interpolation (average) between a corner (such as P_6) and the appropriate (closest) mid-point pressure. The FEM-BOAST II interface saturations for each phase (oil, water, and gas) for all eight nodes, are calculated analogous to the pressure using phase saturations between the appropriate blocks in BOAST II.

For Case 2 the wellblock is located on a corner of the reservoir as shown on Figure 6.5. The pressure for FEM nodes 2, 3, and 5 are obtained as in Case 1 without any modification. But equations 6.2, 6.3, and 6.4 must be modified for use with nodes 1 and 8. This modification is required because BOAST II pressure *P(i+1,j+1,k)* does not exist for Case 2 because node *(i+1,j+1,k)* falls outside the boundary of the reservoir.

The pressures for nodes 1 and 8 are obtained by "reflecting" the appropriate BOAST II pressure so that the existing formulas can be adapted for use in Case 2.

"Reflecting" means $P(i-1, j, k)$ is also used for $P(i-1, j+1, k)$ to determine P_8 and $P(i, j-1, k)$ for $P(i+1, j-1, k)$ to obtain P_1 in Case 2.

To interface our FEM model to BOAST II requires equations 6.2, 6.3, and 6.4 to be modified for node 8 as:

$$R_2^2 = \frac{[dx(i, j, k)]^2 + [\text{'}dx(i-1, j+1, k)\text{'}]^2}{4} \tag{6.6}$$

$$+ \frac{[dy(i, j, k)]^2 + [\text{'}dy(i-1, j+1, k)\text{'}]^2}{4} \tag{6.7}$$

where "*dx*" and "*dy*" represent reflected lengths that are consistent with the reflected pressures. Then,

$$\frac{P_8 - P(i, j, k)}{\log R_1} = \frac{\text{'}P(i-1, j+1, k)\text{'} - P(i, j, k)}{\log R_2} \tag{6.8}$$

$$P_8 = [\text{'}P(i-1, j+1, k)\text{'} - P(i, j, k)]\left[\frac{\log R_1}{\log R_2}\right] + P(i, j, k) \tag{6.9}$$

where $P(i, j, k)$ and $\text{'}P(i-1, j+1, k)\text{'}$ are calculated at each timestep by BOAST II.

The pressure for all FEM interface nodes at the actual FEM-BOAST II interface, nodes 2, 3, and 5, are calculated as in Case 1. The remainder of the wellblock boundary nodes for Case 2 do not actually require interfacing with BOAST II because the boundary nodes 4, 6, and 7 are no longer on an interface with the simulated reservoir. Nodes 4,6, and 7 in Case 2 are actually located on the reservoir

boundary and are considered internal boundary nodes for the FEM wellblock vicinity model by specifying a zero flux for each of these three nodes.

The pressures for the remaining boundary nodes are obtained by using the distance x for the node from the corner (node 3) of the wellblock with following interpolation formula:

$$P_{node} = \frac{(P_l - P_3)}{\Delta x} x + P_3 \qquad (6.10)$$

where, l= 1 or 8, Δx is the length of the wellblock side and x is the distance of the node from corner (node 3) of the wellblock along this side, Figure 6.5.

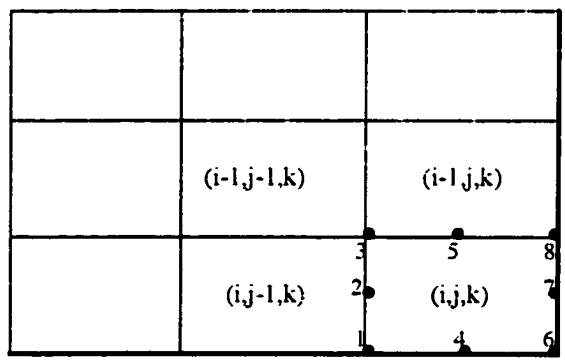


Figure 6.5: Wellblock (10,10,3) with Indices

6.6 Transmissibilities

The BOAST II calculated transmissibilities, saturations and pressures are used for both the initialization and then for the interfaces at each timestep between

BOAST II and the wellbore vicinity model. After transmissibilities are calculated, by BOAST II, they are subsequently used to determine the volumetric flow for each phase at each interface of the wellblock. The volumetric flow for each phase is calculated in the BOAST II subroutine SOLMAT and stored in arrays for this interfacing. These values are read once at the start of the simulation.

The transmissibilities for the gas phase calculated in the BOAST II subroutine SOLMAT were not saved but are used here to explain the BOAST notation. Because the transmissibilities are required for interfacing, six arrays were added to store each phase transmissibility for each stratum at the wellblock boundary. For example the gas transmissibility is:

GW (i,j,k) = AGW	gas west-face
GE (i,j,k) = AGE	gas east-face
GS (i,j,k) = AGS	gas south-face
GN (i,j,k) = AGN	gas north-face
GT (i,j,k) = AGT	gas top-face
GB (i,j,k) = AGB	gas bottom-face

Similar arrays were established for the water and oil phases using analogous notations of *w* and *o* for *g*.

Next, an average volumetric flow-rate for each phase is calculated for the "BOAST II side" of the wellblock boundary. These volumetric flow-rates will be distributed for the "FEM side" of this boundary as previously described in section

6.4. That is, for wellblock indices i and j fixed, the flow rate for each layer (k) into the wellblock is calculated. The arrays used to store these flow-rates are:

Water Phase

QWW (k) = DP1 * WW(i,j,k)

QWE (k) = DP2 * WE(i,j,k)

QWS (k) = DP3 * WS(i,j,k)

QWN (k) = DP4 * WN(i,j,k)

QWT (k) = DP5 * WT(i,j,k)

QWB (k) = DP6 * WB(i,j,k)

Oil Phase

QOW (k) = DP1 * OW(i,j,k)

QOE (k) = DP2 * OE(i,j,k)

QOS (k) = DP3 * OS(i,j,k)

QON (k) = DP4 * ON(i,j,k)

QOT (k) = DP5 * OT(i,j,k)

QOB (k) = DP6 * OB(i,j,k)

Gas Phase

QGW (k) = DP1 * GW(i,j,k)

QGE (k) = DP2 * GE(i,j,k)

QGS (k) = DP3 * GS(i,j,k)

QGN (k) = DP4 * GN(i,j,k)

QGT (k) = DP5 * GT(i,j,k)

QGB (k) = DP6 * GB(i,j,k)

Where the "DP" is the BOAST II calculated pressure drop across the interface. The right hand side of each equation is already BOAST II calculated and the left hand side is used for the interfacing of our FEM model. The new array naming convention used for the volumetric phase flow-rate is: Q, phase, face (subscript).

6.7 Node Numbering

An important feature of the finite element method is its accuracy for irregular boundaries. The wellblock boundary is regular hexahedral but the flow within the wellblock changes from "linear" to "radial". FEM is well suited to approximate this change.

The mesh for the wellblock is divided using elements as shown on Figure 6.6. There are eighty elements per stratum in Figure 6.6 and each of the hexahedral elements has eight local nodes. If one considers a surface of a stratum, there are ninety-six global nodes to be enumerated on Figure 6.6. It helps in the discussion to introduce a grouping of the nodes located the same distance from the center of the wellbore plus the nodes on the interface between the reservoir and the wellblock calling each group a band. Then, there are six bands shown on Figure 6.6. The "wellbore" band is composed of global nodes one through sixteen. The "radial" distance between these bands is not uniform.

Since fluid flow is usually of interest within approximately 20 to 30 feet of the wellbore, additional bands were placed in this vicinity. Our research, with this

data, has shown little change in results when we used more than seven bands for this calculation on these test cases.

To number the nodes, we started at the top of the first layer with the inner-most band starting with 1 and continued counter-clockwise until all the nodes in the first band are numbered as shown in Figure 6.6. Then continue numbering the nodes in the next band sequentially until all these nodes have been numbered and repeat this method for the remaining bands on the top of the first layer. Once all nodes on the top are numbered, continue with the bottom of the first layer. If we have more than one layer, only the bottom of each additional layer requires node numbers because its top layer is shared and already numbered. The bottom of layer one has the same nodes as the top of layer two, etc., as shown in Table 6.1.

Table 6.1: Wellblock Node Numbering

	Top of Layer	**Bottom of Layer**
Layer #1	1 - 96	97 - 192
Layer #2	97 - 192	193 - 288
Layer #3	193 - 288	289 - 384

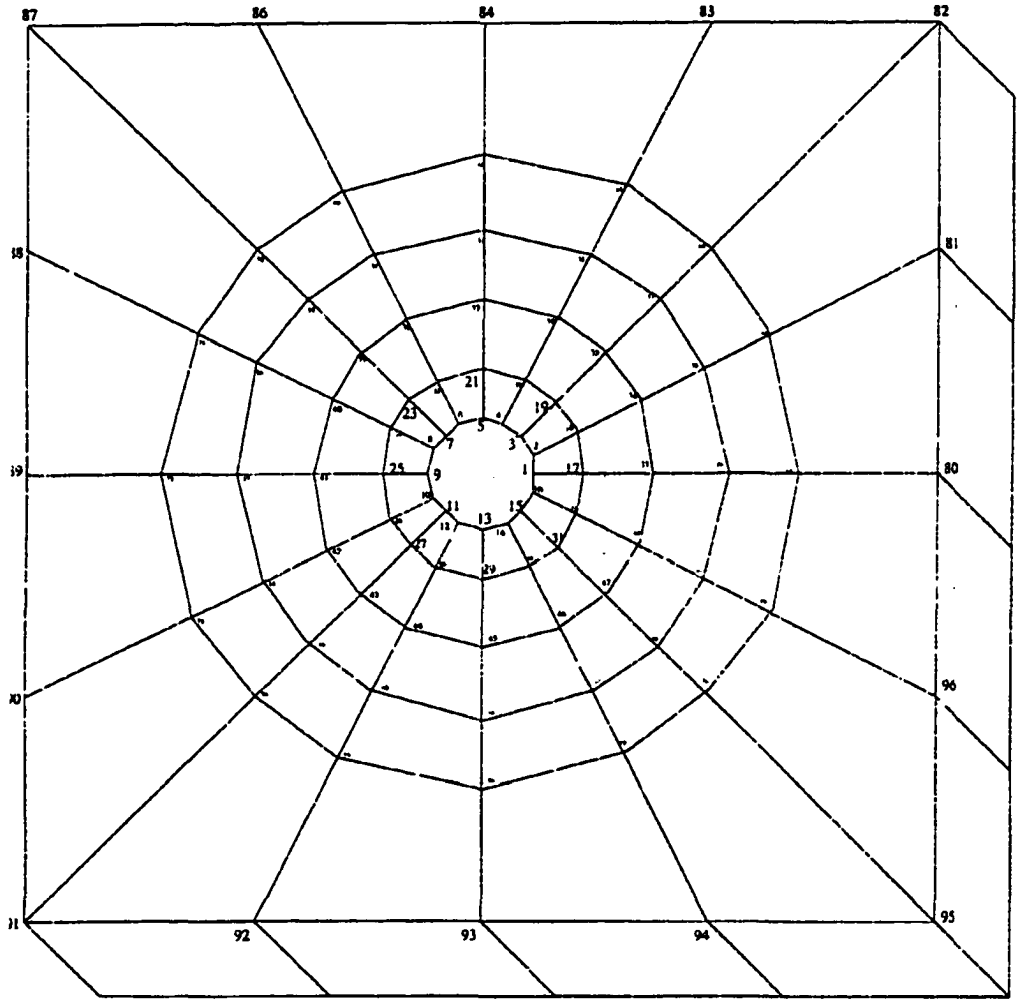


Figure 6.6: FEM Refined Wellbore Vicinity Region

CHAPTER 7. PARALLEL MODEL

For approximately 40 years, high performance computing has had an influence on the evolution of numerical predictive methods for fluid flow in a porous media. Improved fluid flow approximations for reservoirs have evolved. The complexity of reservoir simulation results, with respect to computational requirements, have helped drive the development of faster computers.

Parallel computing was introduced to use architectural features of computers to increase the computing speed, thereby making the numerical solution of even larger scientific and engineering problems possible. One focus of this research centers on how to cost-effectively introduce strongly coupled parallelism into reservoir simulation.

Despite the numerous approaches [31,35,36,49,50,51] that have already been developed and implemented on different parallel computers, several problems remain before parallel computing becomes commonplace as a tool for applications. Important areas of concern for reservoir simulation are: load balancing, data structures, linear equation solvers, etc. Recently proposed parallel numerical algorithms are based on, or closely related to, the principle of domain decomposition used.

One issue that slows the utilization of distributed memory parallel computers is the conversion of existing programs [31] which continues to be difficult. Some progress has been made with shared memory systems with parallelization through the use of language dependent parallel directives. This approach was

chosen to parallelize the FEM wellbore vicinity code. Existing tools were used to determine which portion of our sequential code was best suited to convert to parallel. The code selected could have been done on either a shared memory or a distributed memory parallel platform.

7.1 Wellbore Vicinity Parallelization

One focus of current Computer Science research is to develop parallel algorithms; and this wellbore vicinity model is a good choice because of the computer time required. To develop the parallel FEM algorithm, we:

- Used single subscripts where possible.
- Used multi-nested DO-loops.
- Studied the effects of unfolding each DO-loop to improve performance.
- Used subroutines outside of the Do-loop where possible.
- Reduced the number of subroutine arguments passed by utilizing common blocks.
- Used I/O statements only at the end of the computations.

The wellbore vicinity code was initially compiled on an IBM SP/2 at the Cornell Theory Center using IBM Data Explorer and parallel high performance Fortran (pHPF) to profile and visualize the code. This profile identified the solver and the assembly as arrays that would benefit most from parallelization. The parallel code that we developed used these Cornell obtained results.

The parallel wellbore vicinity code was executed on an SGI Origin2000 with four nodes and shared memory. The parallel code was compiled with SGI's MIPSpro Fortran. Then it was executed using one processor, two processors, three processors, and four processors shown in Figure 7.1.

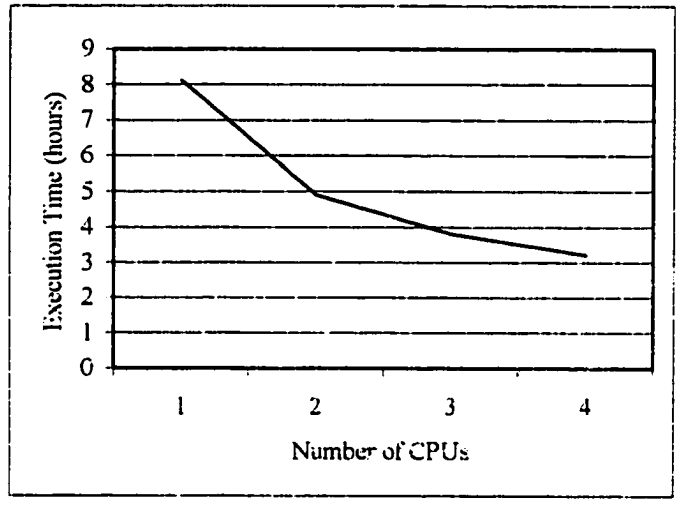


Figure 7.1: Timing of Parallel Execution

The code showed reasonable speedup as more processors were used. However, a natural extension of this work would be the application of parallel linear equation solvers to the FEM reservoir simulation model.

For parallel execution the following steps were used:

- The number of threads to be used when executing the program was defined as: setenv MP_SET_NUMTHREADS 4.
- The -mp option used allowed access to the Fortran multiprocessing library.

- The wellbore vicinity Fortran code parallelized was stored in a file named "Well-BoreVicinity.f".
- To execute: f77 -mp WellBoreVicinity.f .

With these steps, the SGI Origin2000 system created executable code to execute with its four processors. The Unix "ps -Af" command was used to verify that four processes were running in interactive mode. The "-mp" argument created another copy of "WellBore Vicinity.f" as "WellBoreVicinity.f.f" which contained the parallel directives: DO-loops were unfolded, arrays were distributed, variable names were resolved, etc.

The user flexibility for controlling the parallel code was substantial, e.g., some of the "Data Distribution Directives" applied to the code to obtain the desired parallelism is shown in Figure 7.2 and 7.3. The parallel implementation of the code in Figure 7.2 was found to execute better with a distribution for the two-dimensional arrays SOSTIFF(BLOCK,*) and SWSTIFF(BLOCK,*).

```
c$doacross

do i = 1, num_nodes*4
        do j = 1, num_nodes*4
                sofl(i) = sofl(i) + sostiff(i,j)*pfem(j)
                swfl(i) = swfl(i)+swstiff(i,j)*pfem(j)
        enddo
enddo
```

Figure 7.2: Parallel Example #1

The parallel implementation of the code that executes better with SOSTIFF(*,BLOCK) and SWSTIFF(*,BLOCK) is presented in Figure 7.3.

```
c$doacross
 do i = 1, num_nodes*4
       sosum=0.0
       swsum=0.0
       do j = 1, num_nodes*4
         sosum=sosum+sostiff(i,j)
         swsum=swsum+swstiff(i,j)
       enddo
         sofl(i)=sofl(i)+sosum*pfem(i)
         swfl(i)=swfl(i)+swsum*pfem(i)
 enddo
```

Figure 7.3· Parallel Example #2

Although nested parallelism was not supported by SGI, parallelism was exploited across the "perfectly nested DO-loops" in Figures 7.2 and 7.3. The restriction "perfectly nested" means that no code was allowed between the do-i and do-j statements or the enddo-i and enddo-j statements.

Another example shown on Figure 7.4 was also tested. The "c$doacross nest (i,j)" directive specified that the entire set of iterations across the (i, j) loops could be executed concurrently. Another parallel directive used was ONTO. This directive specified the processor topology when an array of more than one dimension is used and one of the dimensions could use more processors than the other.

```
do i = 1, num_nodes*4
      sosum=0.0
      swsum=0.0
c$doacross
     do j = 1, num_nodes*4
        sosum=sosum+sostiff(i,j)
        swsum=swsum+swstiff(i,j)
      enddo
        sofl(i)=sofl(i)+sosum*pfem(i)
        swfl(i)=swfl(i)+swsum*pfem(i)
enddo
```

Figure 7.4: Parallel Example #3

For instance, in the nested DOACROSS shown in Figure 7.5, the ONTO directive divided the available processors among the multiple parallel loops

```
c$doacross  nest  (i,j) onto (2, *)
do i = 1, num_nodes*4
        do j = 1, num_nodes*4
                stiff(i,j) = 0.0
        enddo
enddo
```

Figure 7.5: Parallel Example #4

The ONTO directive assigned processors in the ratio 1:2 to the two dimensional array STIFF(num_nodes*4,num_nodes*4). Both i and j indices were block distributed and two processors were allocated to the outer DO-i loop, the remaining processors were assigned to the DO-j loop.

After converting the wellbore vicinity code to parallel, execution profiles were examined to determine the effectiveness of each conversion. The SGI tool chosen for this was TIMEX, which provided the information used to determine whether or not the parallelized version performed better than the equivalent serial version. The number of threads used for a parallel execution of the code required a matching number of profile data files; one file per thread. Each data file was examined.

7.2 Other Parallel Implementations

Load balancing addresses the assignment of parallel tasks to processors to keep each processor doing useful work. The best parallel performance for our FEM code favors large granularity. That is, when subdivided into parallel tasks, these tasks should be as large as possible to reduce the overhead from interprocessor communication. Load balancing usually requires that each of the parallel tasks be approximately the same size (require the same execution time). Wheeler and Smith [52] addressed load imbalancing caused by irregularly shaped grids through a redistribution of the active cells among the processors achieving load balancing.

Emerging new parallel architectures may alleviate some problems associated with load balancing, message passing, and well management [36]. These new parallel computers have increased the interprocessor communication rates to speeds approaching direct memory access (about 200 Mb/s). One remaining question is whether distributed memory architectures can efficiently be utilized for simulation ?

The shared memory architectures used such parallel reservoir simulation are somewhat mature [25,36,50,51,53].

The solver for this research was a direct solver based on LU decomposition with pivoting. This solver was chosen because it is stable and fast for the small number of global nodes in our system. The time complexity for its factorization with pivoting and backsubstitution is estimated to be:

$$T_{LU}(M) = \frac{2}{3}M^3 + M^2 + M^2 \tag{7.1}$$

where an MxM system of equations is assumed for A in:

$$Ax = f. \tag{7.2}$$

This estimate is for a single processor system.

The finite element method develops an equation for each element and then assembles these elemental equations into a stiffness matrix which is solved. This is repeated for each timestep. An efficient parallelization of this would speed up the overall parallel performance. Our calculations showed that a better parallel direct solver is needed.

Existing sequential reservoir simulators do not achieve a desired level of parallel performance when converted to parallel, as was shown in Figure 7.1. Moreover, it is difficult to efficiently change sequential codes to parallel because the data structures employed are not specifically designed for parallel computers.

Methods such as domain decomposition at reservoir levels that allow local grid refinement around wells and localized fluid interfaces as well as parallel solvers should result in significant performance improvement [25].

Iterative solvers, such as GMRES, SOR, and bi-CG, have parallel implementations and would be an excellent improvement to BOAST II. Any improvement in the implementation of these solvers on shared-memory systems and distributed-memory systems would improve parallelism. The matrix-vector products used in these simulations are easily parallelized on shared-memory system by splitting the matrix into strips corresponding to the vector segment. Depending on bandwidth of the stiffness matrix, communication between processors may lead to communication bottlenecks.

CHAPTER 8. MULTIMEDIA VISUALIZATION

The information super-highway with other technologies, make it economical and possible to show and better utilize research results around the world. The reservoir simulation tools developed in this research can be used in reservoirs that exist in other countries.

Proper management of these reservoirs would be facilitated with the use of accurate simulations. The process of numerically modeling underground reservoirs is very difficult due to the complexity of fluid flow but is also very useful. For example, in petroleum reservoirs, the amount of oil and gas ultimately produced can be increased significantly with proper management which depends upon simulations.

A manager simulating a reservoir, could and probably should avoid both the complexity and the cost of developing a simulation program. One can avoid these costs by utilizing and customizing the existing simulation programs. With localized versions of operating systems, browsers, and database management systems, computer trained personnel can develop "front-end" applications in their language to visualize results of existing simulators. For example, with the aid of web browsers, one could translate outputs of simulators into the local language for presentation. This presentation can be text, audio, video, graphics, or all of these. A prototype was developed [91,92] that displays the simulator's results in a table format and dynamically animates these results using Java or VRML using the world wide web.

8.1 Wellbore Vicinity Prototype Model

The prototype model that we developed to visualize results of this fluid flow numerical model is based on the world wide web (the web) because it is platform independent and it supports various forms of media [90,91]. The webs's basic language is hypertext markup language (HTML). We used HTML specification 3.2 because of its support for tables to organize textual output of our simulation. The web also supports the new programming language Java. Java applets support interactive and dynamic applications that use audio, tex, sound, video, and images. Using JavaScripts, table entries can be modified by the end user, captured, and sent back to the executing simulation. With the Common Gateway Interface (CGI), the visualization server can deliver to the web browser information that was not readable by the browser, such as SQL database entries.

Traditional graphics tools provide only a limited visualization. To truly understand the underlying, physical phenomena of fluid flow in reservoirs, scientists need visualization tools that are in the language of the user, flexible, powerful, and easy to use.

The wellbore vicinity prototype system is a novel approach to create visualizations that are sufficiently powerful yet easy to use by scientists in their language, Figure 8.1. Computer scientists who implement this model should not have to be concerned with the details of the complex reservoir simulator. They only need to program the output files and understand any user feedback requirements if interaction with the simulator is desired.

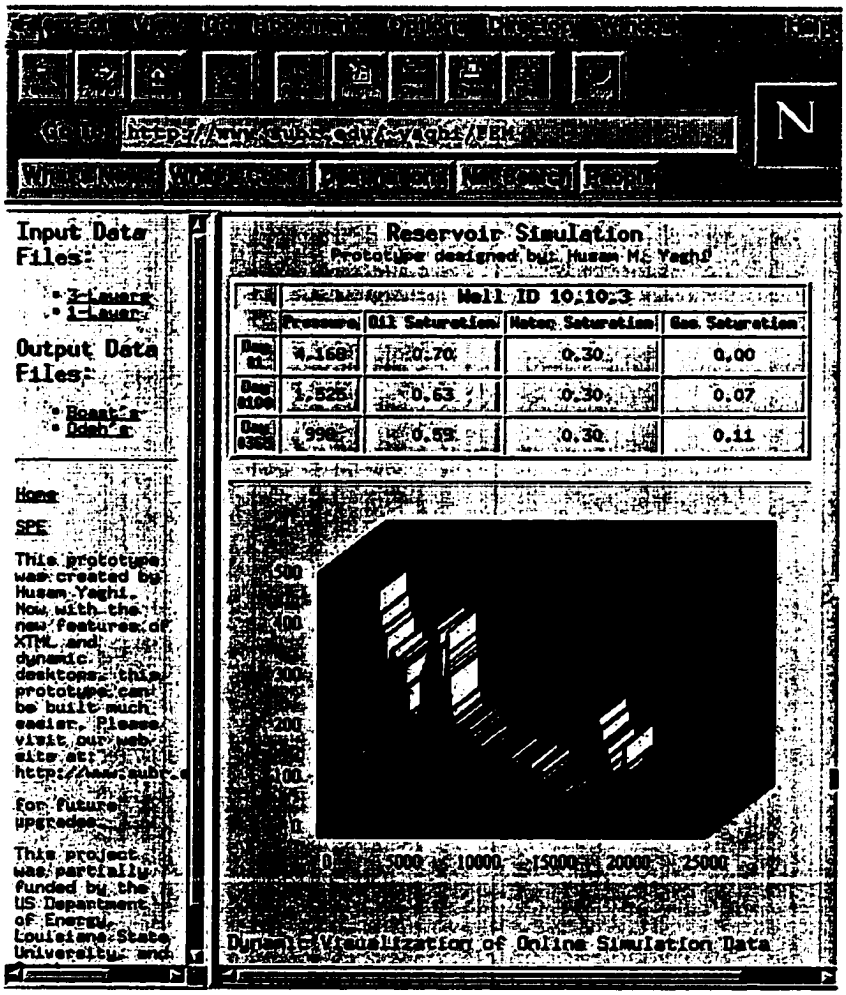


Figure 8.1: Wellbore Vicinity Visualization

In the last several years the need to visualize complicated dynamical processes in the geological sciences has been stimulated by the increasing availability of computers. The usual practice of visualizing the outputs of time-dependent simulations, such as hydrocarbon reservoirs, has been to postprocess the data after the simulation has finished. This approach has drawbacks when dealing with large data sets. It would involve a substantial financial investment in disk-space, exceeding tens of gigabytes, and a tremendous human effort to just retrieve and reload the data versus the less expensive visualization. Our approach here was the visualization of hydrocarbon reservoir simulations plus the concept of interactive visualization, making use of available systems.

8.2 Distributed Visualization

Recent research has been focused on interactive co-processing of visualization and flow field simulations using a cluster of available workstations. Parallel reservoir simulation can be monitored by transmitting a small subset of intermediate simulation results to a graphics workstation for each time step. This works for small problems but is prohibitive for other parts of reservoir simulation problems because these visualize the entire flow field. The amount of data transmitted in this could be enormous.

There exists a novel tool (COVISE) for the interactive visualization between distant sites which integrates visualization and simulation tasks across heterogeneous hardware platforms in a seamless manner.

For our simulation, we utilized the world wide web coupled with the emerging Java and VRML. These tools can be used with Microsoft Access or other database management systems to retrieve data and produce visualization that interacts with the running simulator.

Our wellbore vicinity prototype system is distributed and platform independent. Using the Hyper Text Transport Protocol (HTTP), the system modules, each residing on a different server located any where in the world, Figure 8.2, exchange information and collaborate to produce a unified simulation output to the end user.

For multilingual reservoir simulation, the simulation program and its outputs are all in English. However, with support from the web tools discussed, the end user's browser would determine the language used for presentation of the simulation results. The end user does not need to know about the details and complexity of the underlying simulation programs, rather s/he is delivered a browser interface module. This module translates communications between the visualization modules and the end user's "client browser". The client browser can use any linguistic form.

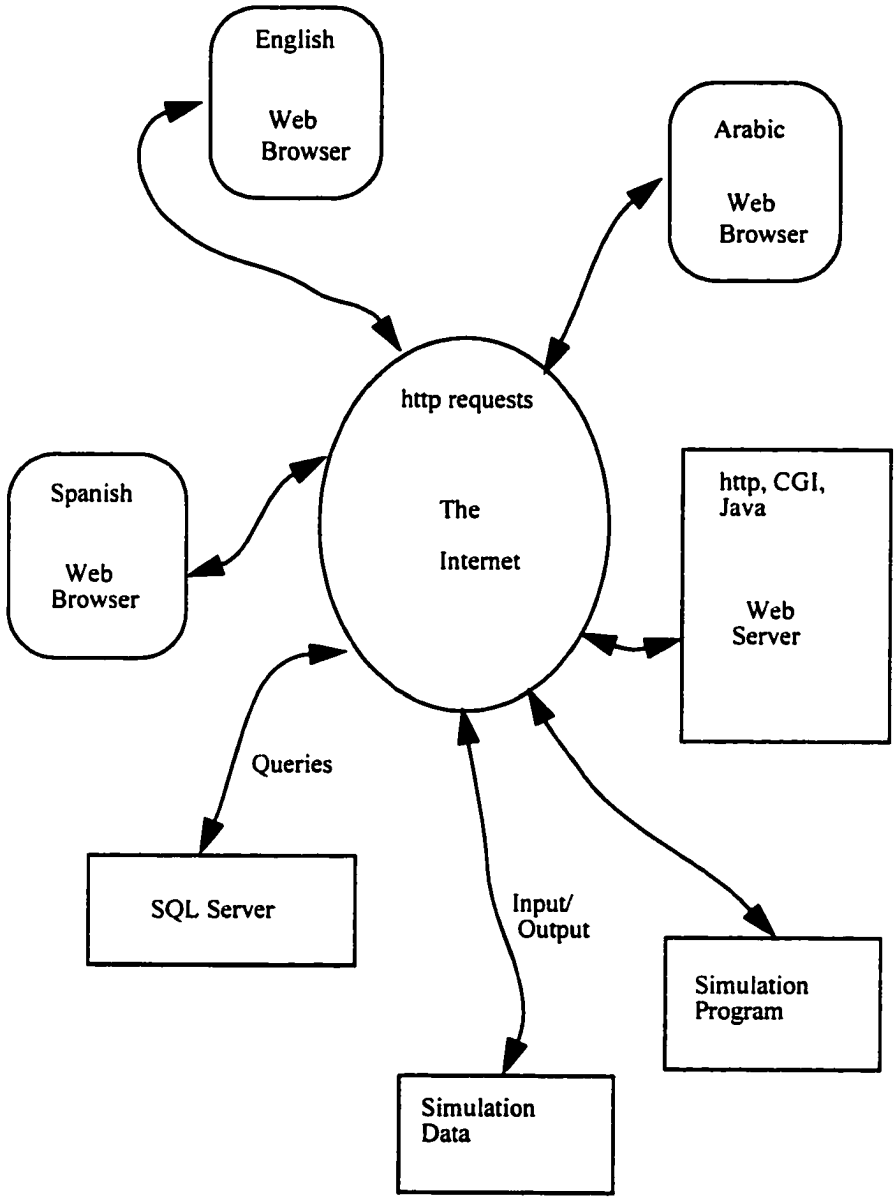


Figure 8.2: Distributed Visualization System

CHAPTER 9. RESULTS

A 3D triphasic finite element-finite difference hybrid black oil reservoir prototype simulator has been developed to improve and provide simulations in the vicinity of the wellbore. The finite element simulation of the wellbore was interfaced with BOAST II. The source code listing for the FEM wellbore vicinity is presented in Appendix A. This prototype has improved accuracy for heterogeneous wellbore regions composed of horizontal layers including high permeability. This prototype also supports the study of flow behavior in the wellbore vicinity.

To refine the solutions around the wells, the finite element method was used. To demonstrate the proposed use of this new wellbore vicinity simulation, solutions are obtained for a one-layer reservoir and three-layer reservoir.

The two SPE simulation problems selected are for a single production well. The production well had a constant rate and was located in the center of a square reservoir for the one-layer reservoir; Case 1. The production well was located at a "corner" in the three-layer reservoir with a varying rate and gas being injected to maintain the reservoir pressure; Case 2. In the gas injection demonstration problem (Case 2), the production well was located at the corner of the reservoir with zero flux at two of the boundaries in this wellblock.

All test data was furnished by Dr. Ming-Ming Chang, senior research engineer, U.S. Department of Energy's National Institute for Petroleum and Energy Research (NIPER).

Our improved results are compared with published results for these same simulations [3, 12, 35, 36, 69, 70, 72, 80, 81].

9.1 Case 1

The simulated reservoir in Case 1 is divided into 99 blocks (Figure 9.2) with only one production well located in block (i=6,j=5,k=1). Wellblock (6,5,1) has a flux at all boundaries; i.e., it has flow at the side boundaries of the block. The indices used for each block in the reservoir are given in Figure 9.2. The well vicinity for this is 120 feet long, 120 feet wide, and 50 feet in depth with the wellbore at its center, Figure 9.1.

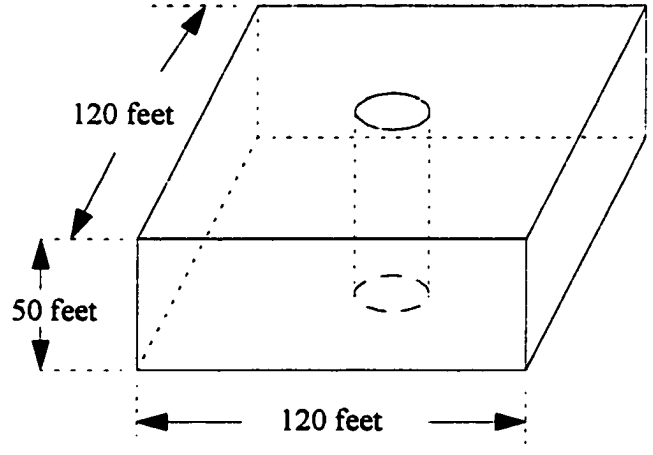


Figure 9.1: Well Vicinity (6,5,1)

Only the block with the production well is refined with the FEM mesh shown in Figure 9.3. Reservoir properties, pertinent data, and parameters for this problem are given in Appendix B.

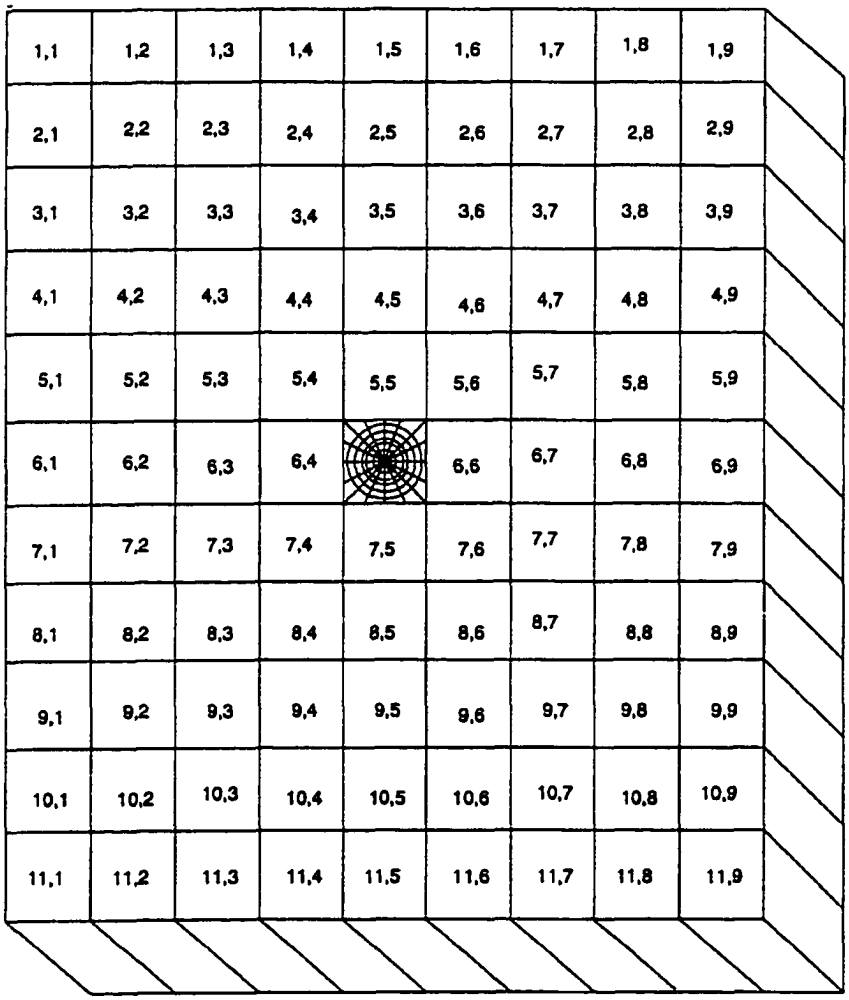


Figure 9.2: Case Study 1

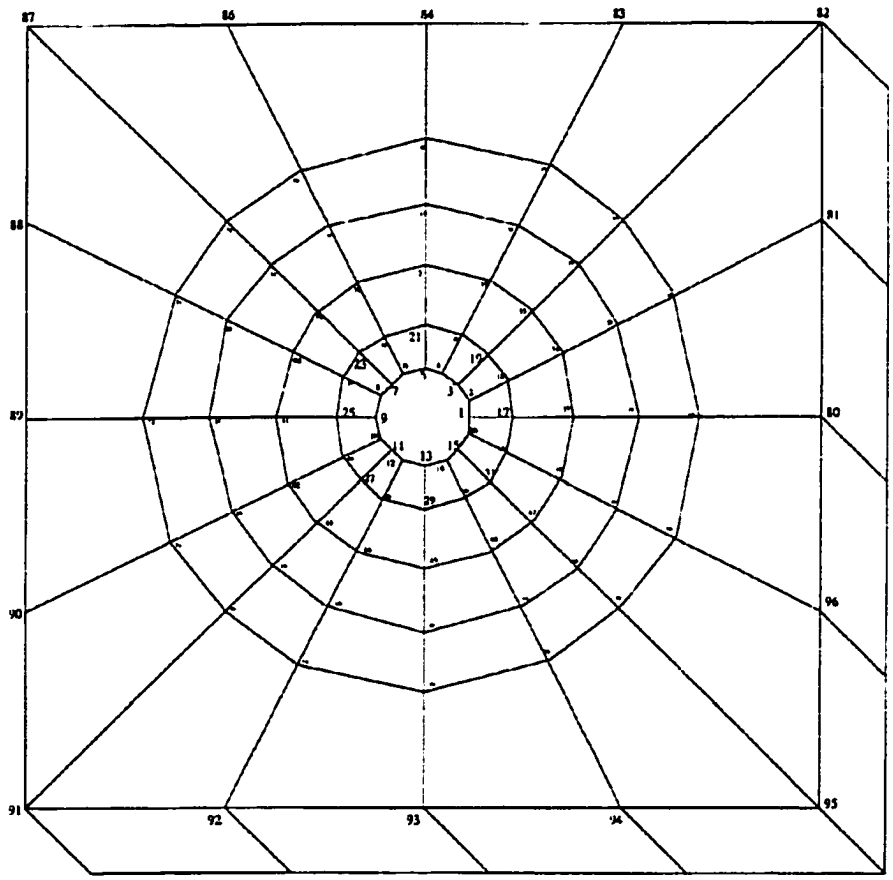


Figure 9.3: FEM Refined Well Region (6,5,1)

Initially, there were two liquid phases (oil and water) present in the reservoir. All gas was dissolved in the oil or water phases. The water phase was immobile because its concentration always remained below the critical saturation of water needed to cause mobility for this simulation. As the oil pressure decreases (oil is being produced) the gas comes out of the oil phase (the bubble point) or from the water phase producing free gas in the reservoir. The oil pressure decreases during this simulation as expected and free gas saturation appears in regions that are below the bubble point pressure.

We found no simulation results for this simple test case and cannot include a comparison to known results. Figure 9.4 presents a history of the average oil pressure in the wellblock for Case 1. The pressure decreased sharply at the beginning of the simulation, followed by a long time period of decrease at a steady rate. The FEM wellbore vicinity model interfaced with BOAST II was stable for the entire simulation time of 1500 days. The results were consistent with those obtained from BOAST II without the wellbore vicinity model; Figure 9.5. Figure 9.6 presents a history of the average saturation for the three phases.

Figures 9.7, 9.8, and 9.9 present results that are not possible with BOAST II. These results show that this FEM wellbore vicinity model can determine the behavior of fluids at any nodal point within the wellblock. Figures 9.7, 9.8, and 9.9 present oil pressure versus distance from the wellbore at simulation times of 10, 1000, and 1500 days respectively. Figure 9.10 demonstrates that pressure drops over time and increases when moving away from the wellbore.

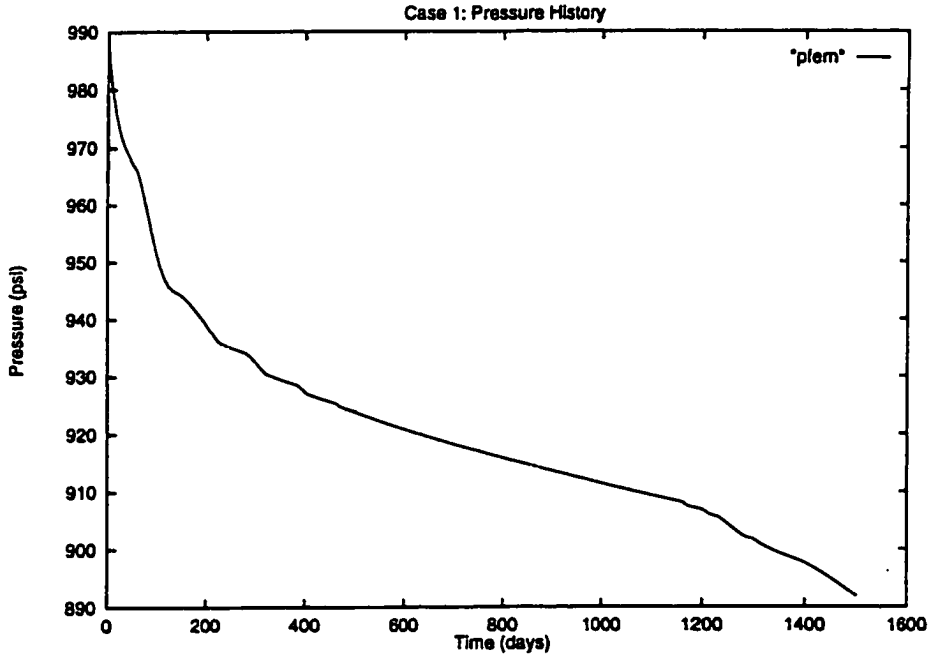


Figure 9.4: Case 1: Average Pressure History

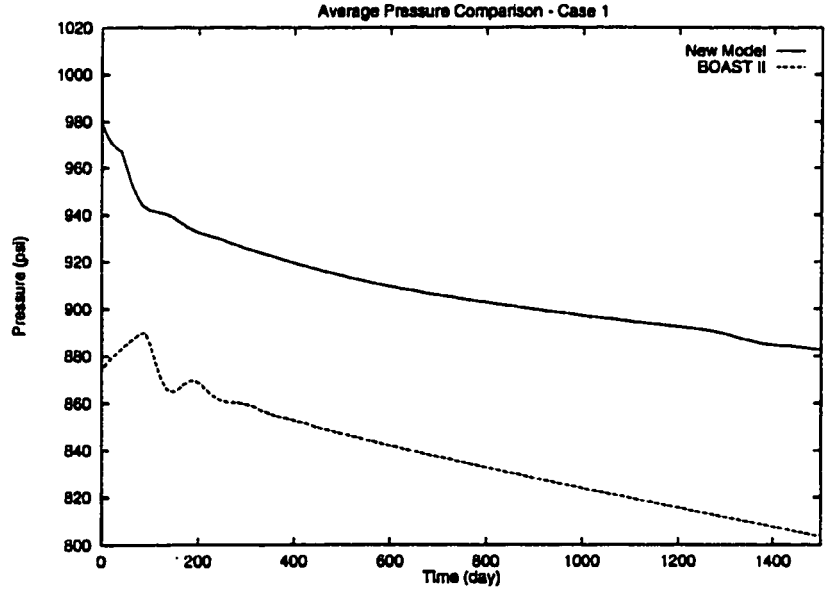


Figure 9.5: Case 1: Average Pressure Comparison

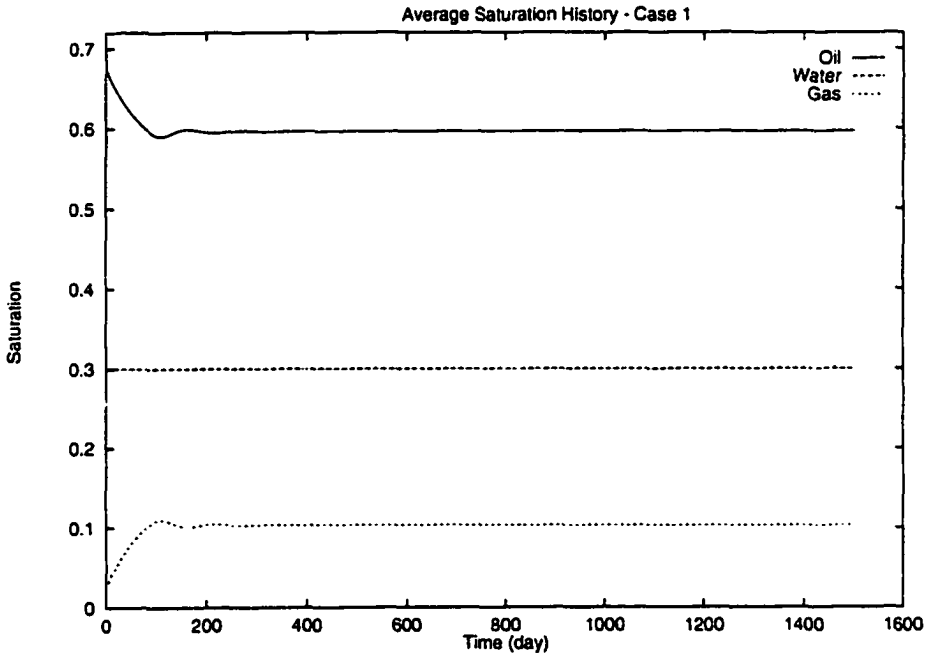


Figure 9.6: Case 1: Average Saturation History

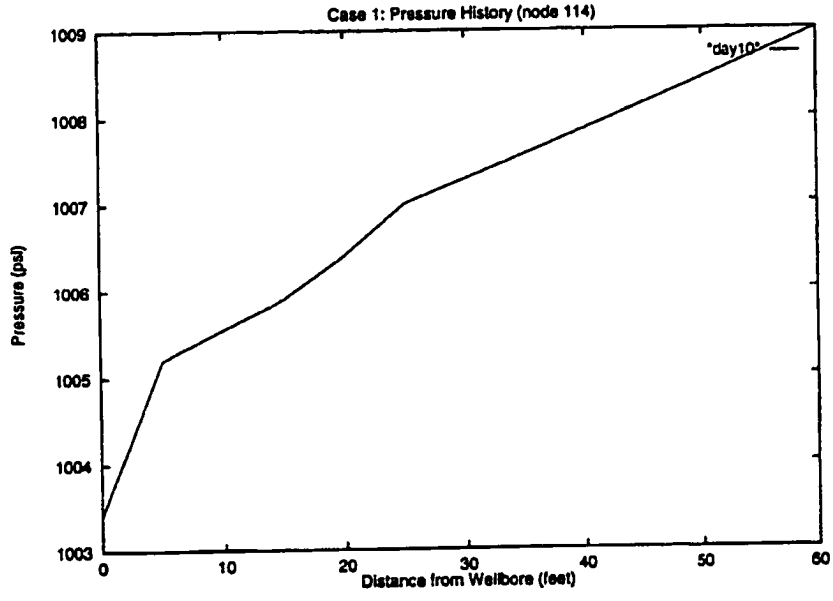


Figure 9.7: Case 1: Pressure After 10 Days

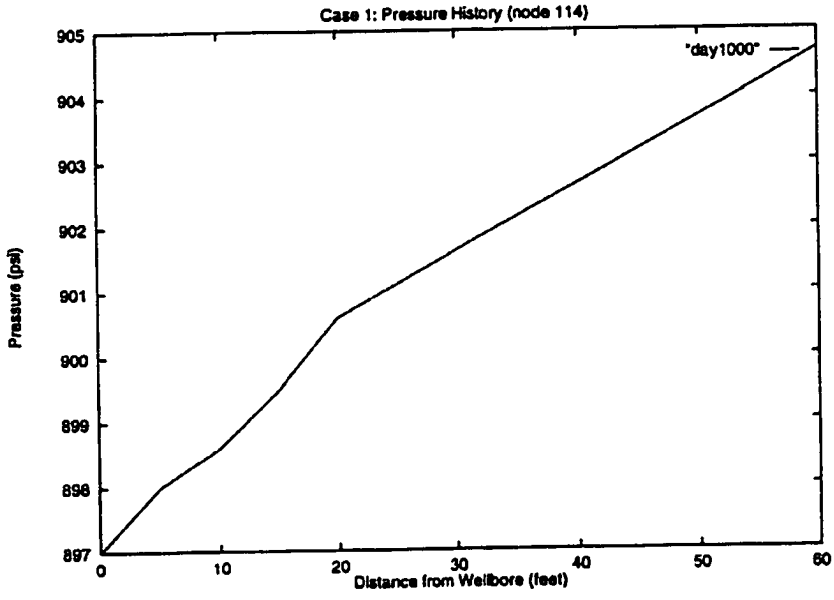


Figure 9.8: Case 1: Pressure After 1000 Days

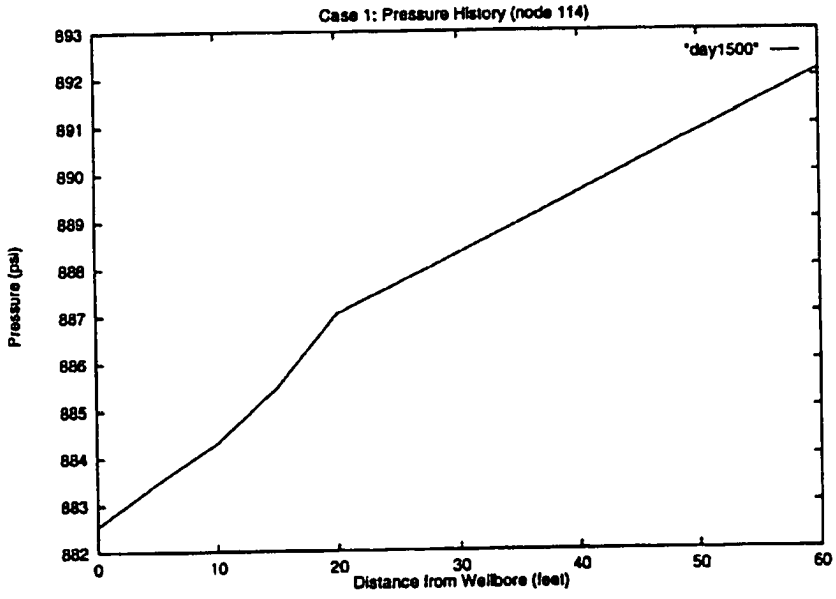


Figure 9.9: Case 1: Pressure After 1500 Days

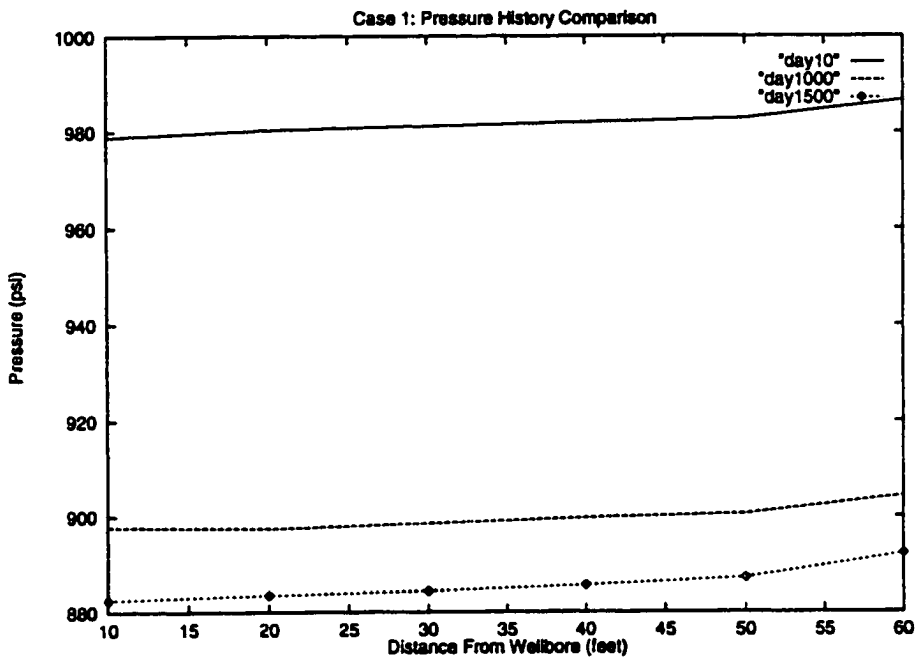


Figure 9.10: Case 1: Pressure History Comparison

9.2 Case 2

The reservoir simulation used for Case 2 shown on Figure 9.11 has two wells, an injection well and a production well; also known in the literature [12] as SPE Problem 1, Case 1. The reservoir is subdivided into the 300 blocks shown on Figure 9.12. The injection well is located in block (1,1,1) and the production well is located on the corner of the reservoir in block (10,10,3). The simulated reservoir is 1,000 feet long, 1,000 feet wide and 100 feet in depth. The top layer, in which the injection well is located, has a thickness of 20 feet, the middle layer has a thickness of 30 feet, and bottom layer, in which the production is located, has a thickness of 50 feet. The production well actually produces from layer 3 only or block (10,10,3) presented on Figure 9.12. The FEM wellblock vicinity for this simulation consists of the three blocks described with the original BOAST II indices as (10,10,1), (10,10,2), and (10,10,3) and shown on Figure 9.13.

The wellbore vicinity is shown on Figure 9.14 with nodal numbering representing the bottom layer (10,10,3) of the well vicinity from which the production occurs. Figure 9.14 is used as a reference for various results.

The FEM mesh is constructed radially out from the wellbore using 6 bands as described in section 6.7. The reservoir properties, pertinent data, and parameters used for this simulation problem are given in Appendix C.

Figure 9.15 presents a history of the average wellbore pressure from the FEM wellbore vicinity model simulation. Figure 9.16 presents the average saturation computed with the FEM wellbore vicinity model.

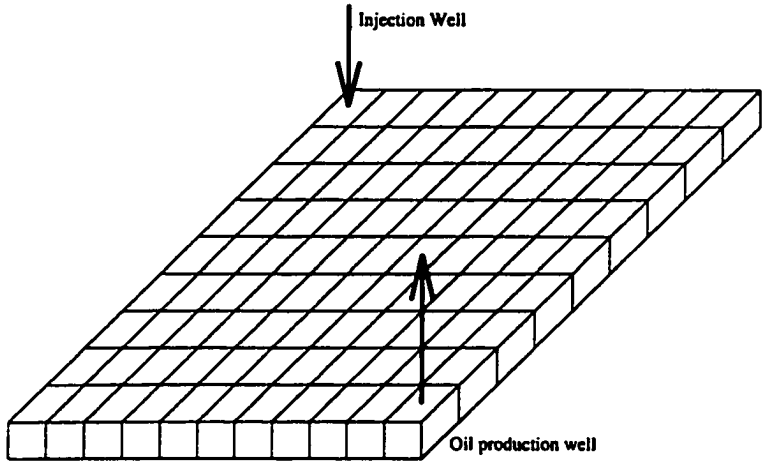


Figure 9.11: Case 2 (Injection & Production Wells)

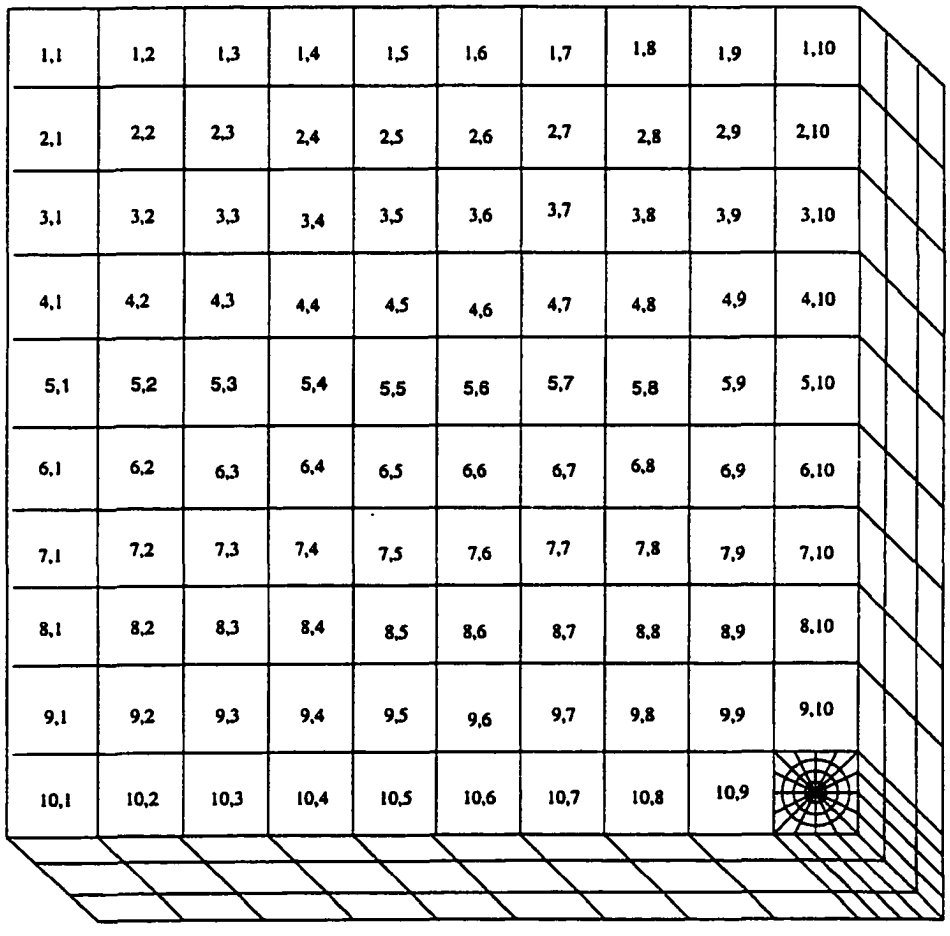


Figure 9.12: Case 2 - The Reservoir

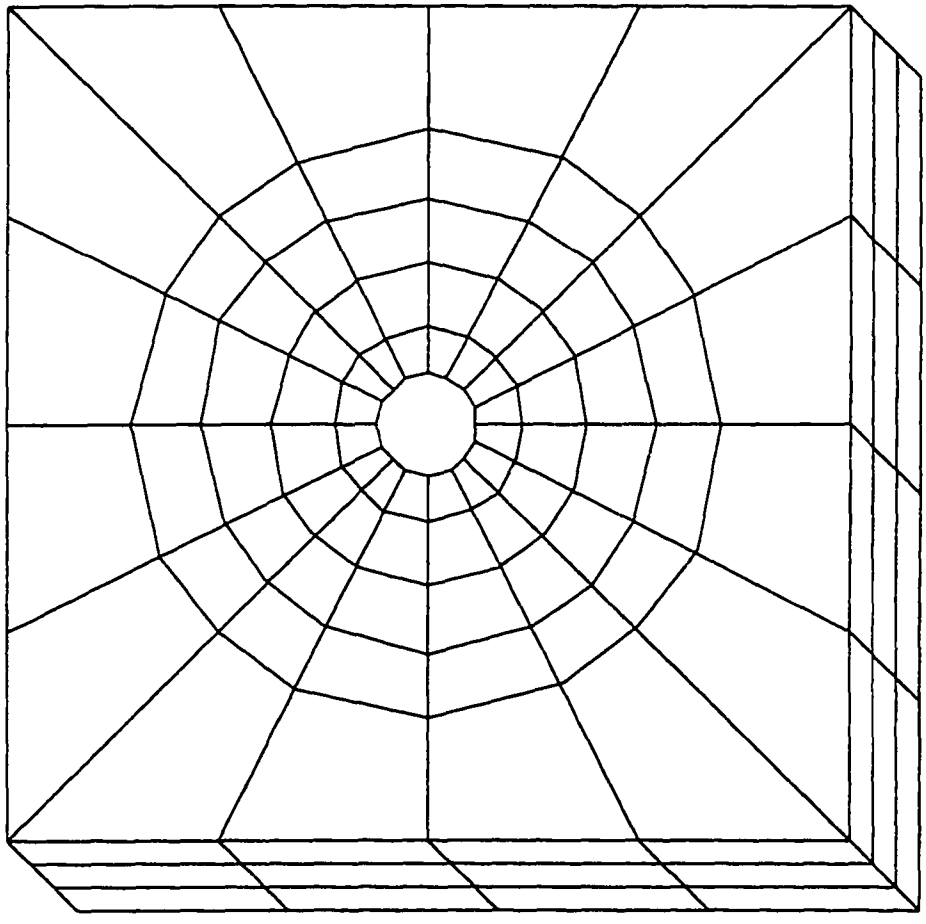


Figure 9.13: Case 2 - FEM Refined Well Region

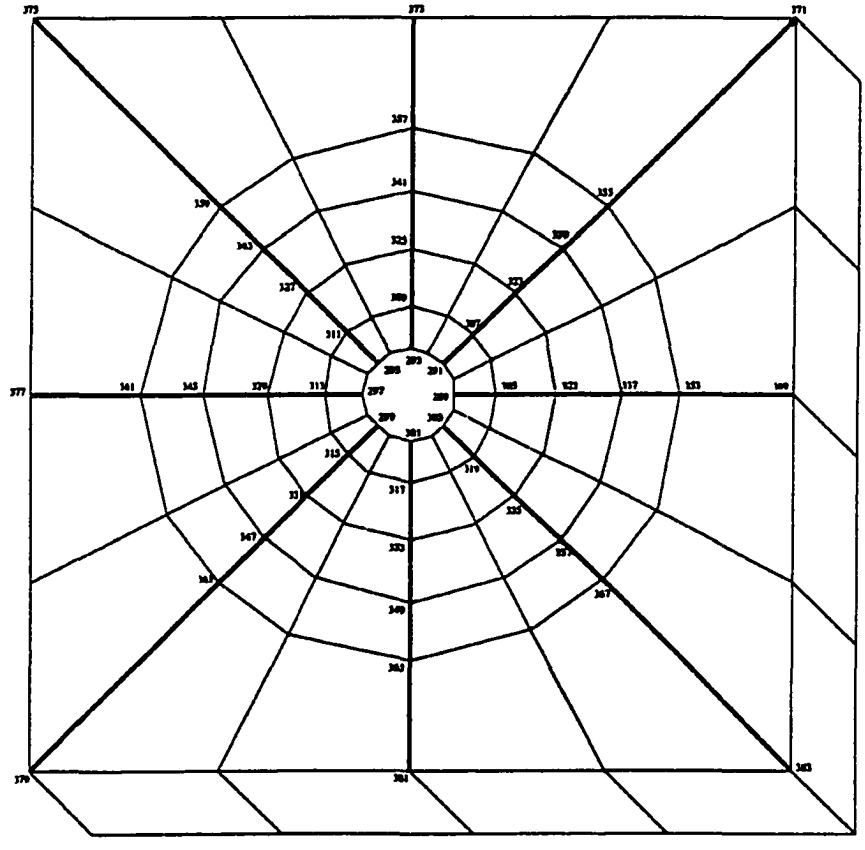


Figure 9.14: Node Numbering (at bottom of layer #3)

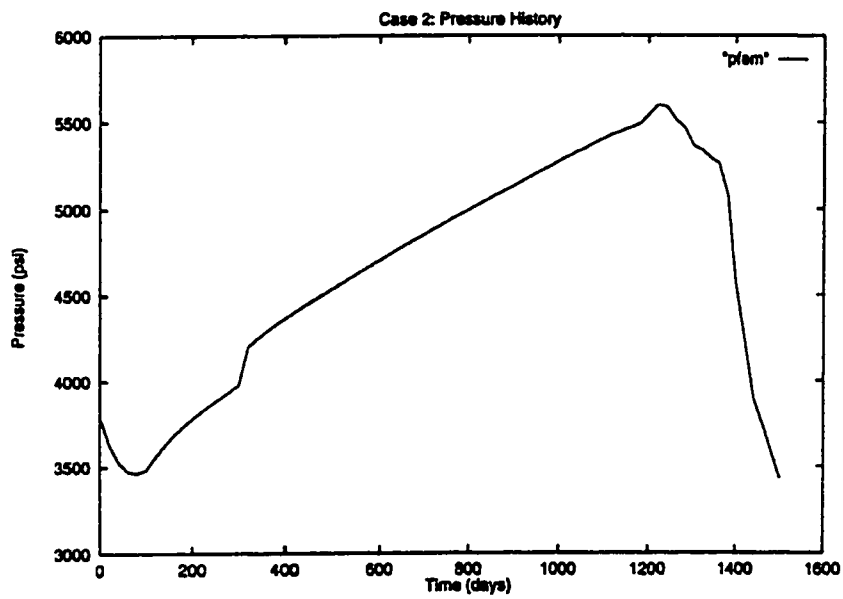


Figure 9.15: Case 2: Average Pressure History

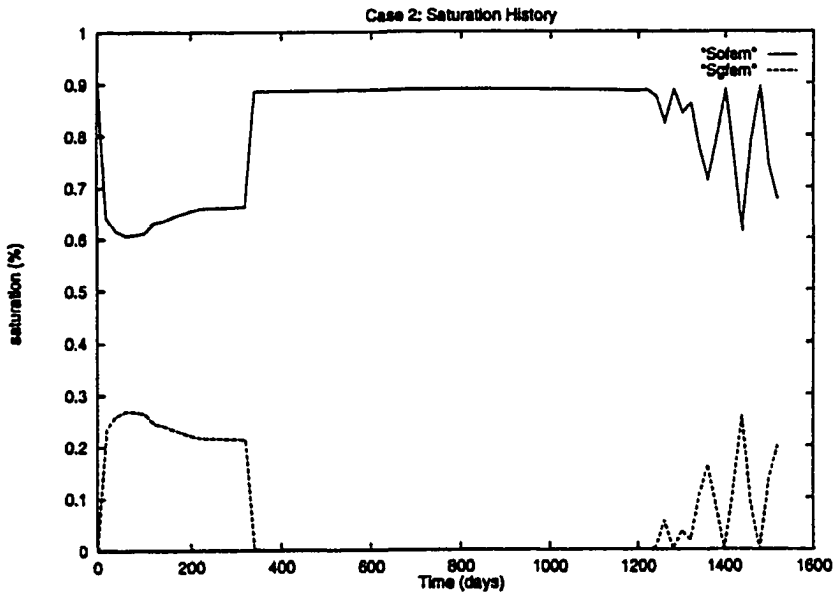


Figure 9.16: Case 2 - Average Saturation History

The FEM wellbore vicinity model was stable for the approximately four years. An explanation for this stability problem will be discussed later.

The results presented on Figures 9.17 - 9.24 are not possible with BOAST II. The figures demonstrate that the FEM wellbore vicinity model can model the behavior of fluids at any nodal point within the wellblock. Figures 9.17, 9.18, and 9.19 present oil pressure versus distance from the wellbore at simulation times of 10, 1000, and 1500 days respectively. Figure 9.20 demonstrates that the pressure drops over time and increases as we move away from the wellbore.

Figures 9.21, 9.22, and 9.23 present various nodal pressures at different locations within the wellblock. The results shown are consistent with the dynamics of fluid flow.

Figure 9.24 presents a comparison of pressure history at various nodes on the outer boundary of the wellblock; that has zero flux. The figure shows that the pressure at 50 feet above the bottom of the wellblock is higher than the pressure at 25 feet above or even at the bottom of the wellblock; represented by nodes 371, 369, and 383 respectively.

Table 1 presents a portion of output data that cannot be obtained by BOAST II. The data presents oil pressure versus distance from the wellbore for all three layers. The pressure at layer #1 is slightly higher than the pressure at layer #2. The pressure at layer #2 is slightly higher than the pressure at layer #3.

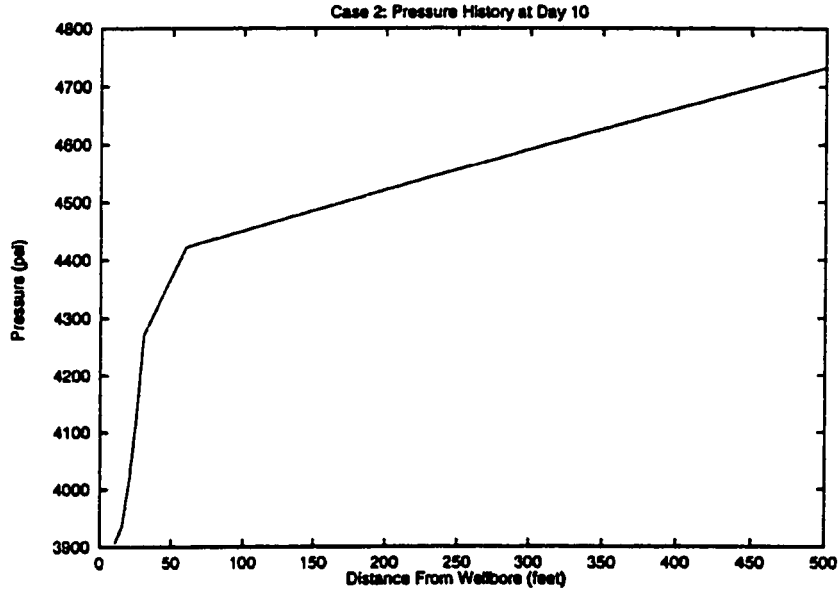


Figure 9.17: Case 2 - Pressure History at Day 10

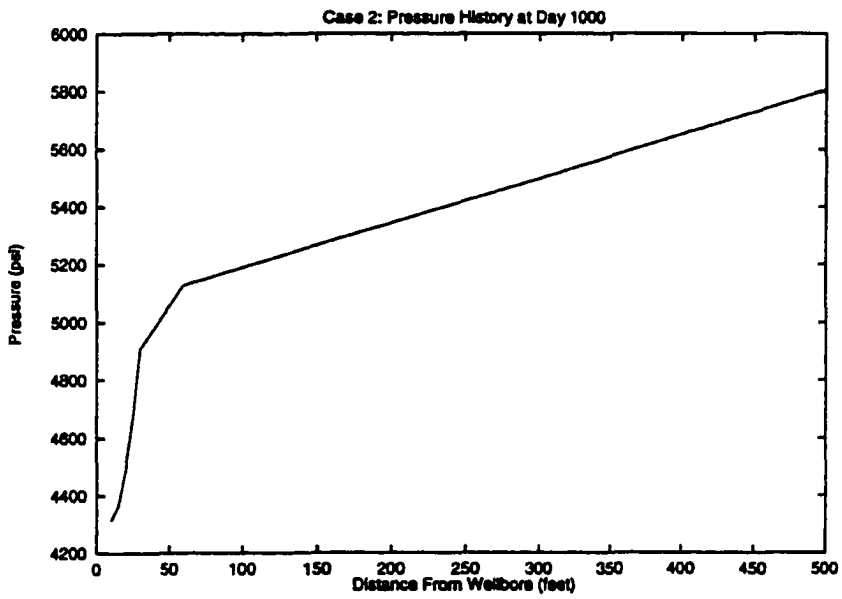


Figure 9.18: Case 2 - Pressure History at Day 1000

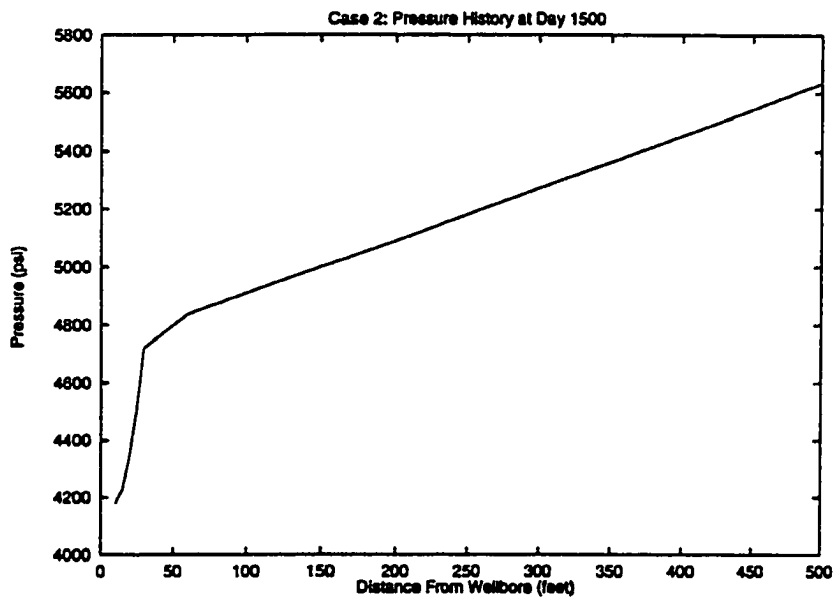


Figure 9.19: Case 2 - Pressure History at Day 1500

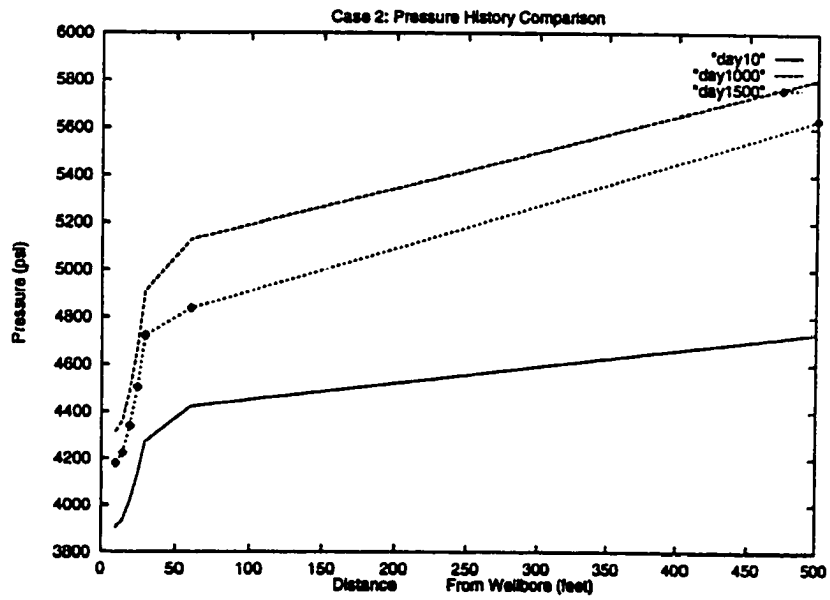


Figure 9.20: Case 2 - Pressure History Comparison

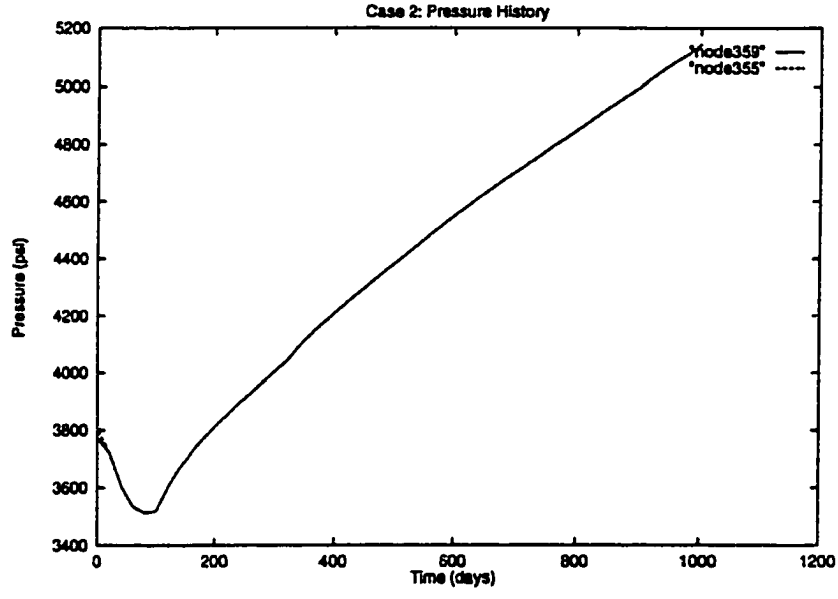


Figure 9.21: Case 2 - Nodal Pressure History

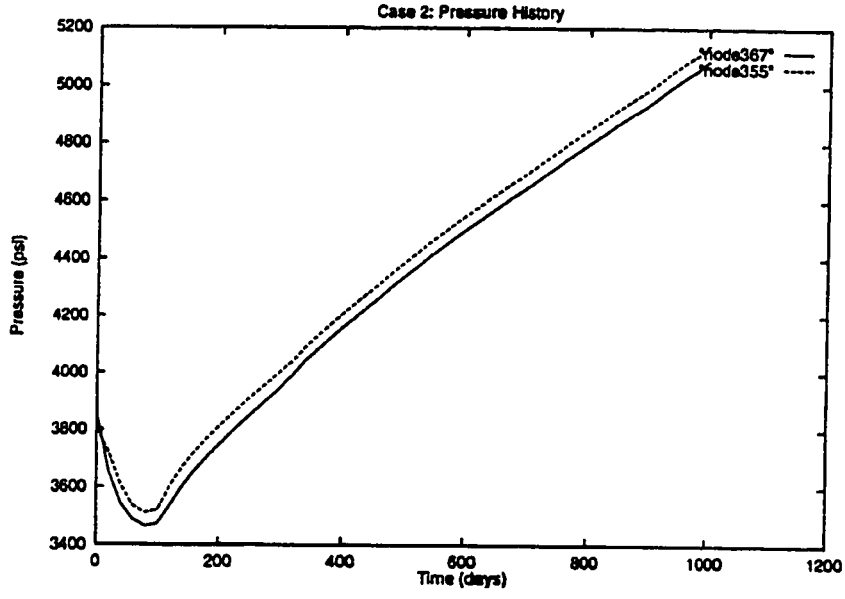


Figure 9.22: Case 2 - Nodal Pressure History

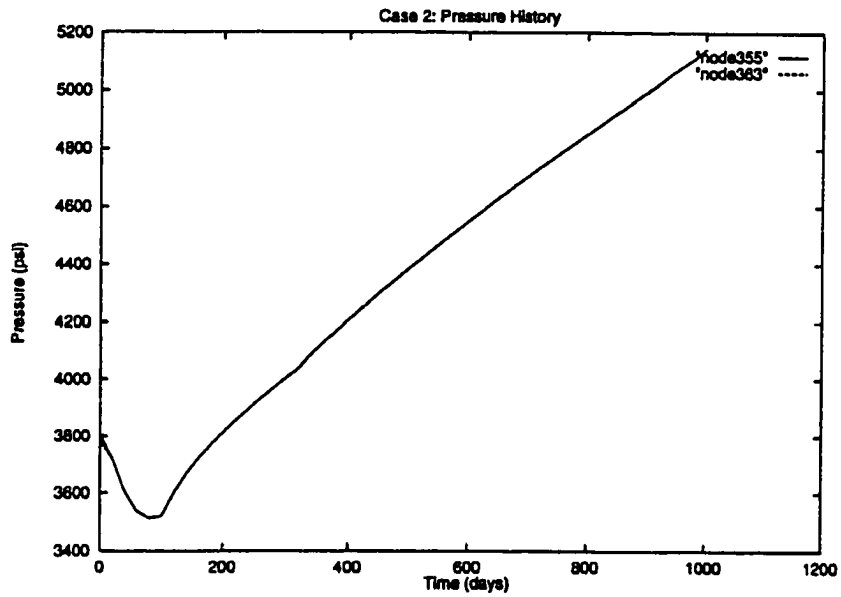


Figure 9.23: Case 2 - Nodal Pressure History

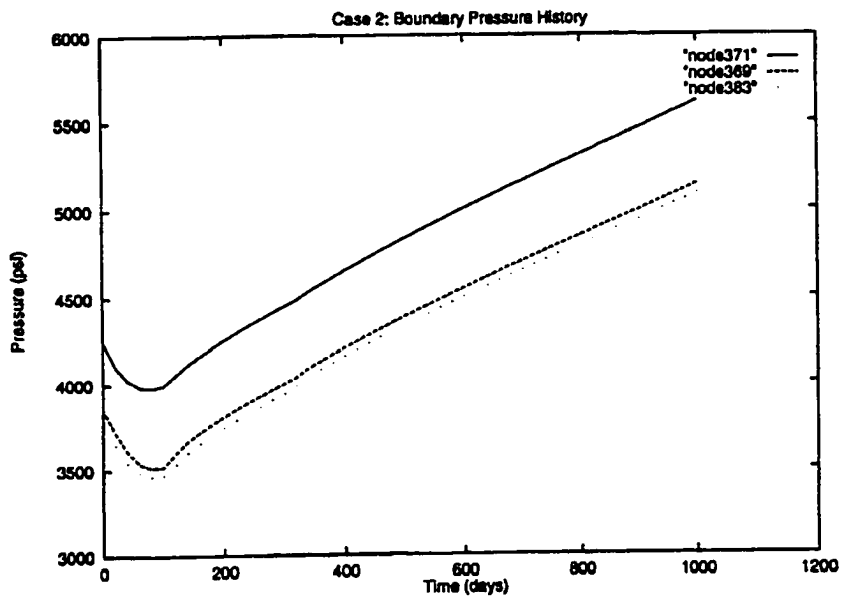


Figure 9.24: Case 2 - Boundary Pressure History

Table 1: Case 2: Pressure History versus Distance from Wellbore

Pressure at a node and its corresponding nodes in all three layers at a distance from the wellbore as shown at each of the three layers.

After 1 Day

Distance	Layer #1	Layer #2	Layer #3
1	3878.2342	3875.2068	3869.5934
2	3910.5131	3907.4878	3901.8208
4	3992.5740	3989.5880	3983.9515
8	4109.1347	4105.9637	4100.2119
16	4254.3415	4252.5788	4247.3401
60	4420.8992	4416.0482	4409.1121
500	4724.2622	4732.6831	4732.6831

After 6 Days

Distance	Layer #1	Layer #2	Layer #3
1	3623.5436	3620.8545	3615.8356
2	3653.7350	3651.0575	3645.9907
4	3730.6630	3727.9913	3722.9453
8	3838.7230	3835.9639	3830.8349
16	3977.9347	3976.2181	3971.4865
60	4113.0247	4109.2080	4103.1495
500	4523.3082	4531.7360	4531.7360

After 11 Days

Distance	Layer #1	Layer #2	Layer #3
1	1769.0662	1766.8512	1762.1003
2	1855.5820	1853.4285	1848.6067
4	2082.7827	2080.8578	2076.0639
8	2421.9774	2420.1215	2415.2566
16	2874.4097	2875.0803	2870.7191
60	3426.6763	3425.7105	3420.2911
500	4377.5210	4385.9325	4385.9325

Table 1: (continued)

After 826 Days

Distance	Layer #1	Layer #2	Layer #3
1	4122.3628	4119.1034	4112.9068
2	4167.1958	4163.9652	4157.7038
4	4281.8874	4278.6458	4272.4090
8	4443.1156	4439.8731	4433.5388
16	4653.4938	4651.3216	4645.4784
60	4852.3839	4849.0082	4841.5945
500	5479.1729	5487.5796	5487.5796

After 1001 Days

Distance	Layer #1	Layer #2	Layer #3
1	4256.5676	4253.2096	4246.7599
2	4307.3931	4304.0666	4297.5453
4	4437.5030	4434.1703	4427.6758
8	4620.9674	4617.6406	4611.0421
16	4859.5100	4857.3546	4851.2824
60	5091.9480	5088.6955	5080.9808
500	5768.6663	5777.1054	5777.1054

After 1316 Days

Distance	Layer #1	Layer #2	Layer #3
1	4176.2668	4172.9809	4166.5065
2	4238.8628	4235.5847	4229.0090
4	4399.6801	4396.4046	4389.8653
8	4628.1980	4625.0155	4618.3643
16	4925.5671	4923.4000	4917.3103
60	5223.6141	5221.7139	5213.9534
500	6043.7746	6049.7830	6049.7830

9.3 Comparison of Results

In January of 1981, the Society of Petroleum Engineers published an article by Aziz Odeh [12]. Then in March of 1986, the Journal of Petroleum Technology published another similar comparison by Weistein, Chappelear, and Nolen [80].

Odeh reported results of the pressure computed using several different simulators for case 2; Figures 9.25. The commercial simulators tested were stable for about 10 years of simulation. Our FEM wellbore vicinity pressure does not match the pressures reported by Odeh, however it improved on the pressure from the BOAST II simulator. These new pressure results are compared to the same calculations without our FEM wellbore vicinity code; Figures 9.25 and 9.26. The reason for the differences from the published is that our FEM wellbore vicinity model uses as input the already inaccurate results calculated by BOAST II. Since the input data to our model are the inaccurate values supplied by BOAST II, our simulation results are inaccurate and cannot be expected to match Odeh's. Some of the reasons for BOAST II's inaccurate results are:

- The IMPES method is used without any iteration.
- One average pressure is used for the center of each block, to represent pressure for the entire wellblock.
- Finite difference method is used for the entire reservoir.
- Does not do a fully implicit calculation.

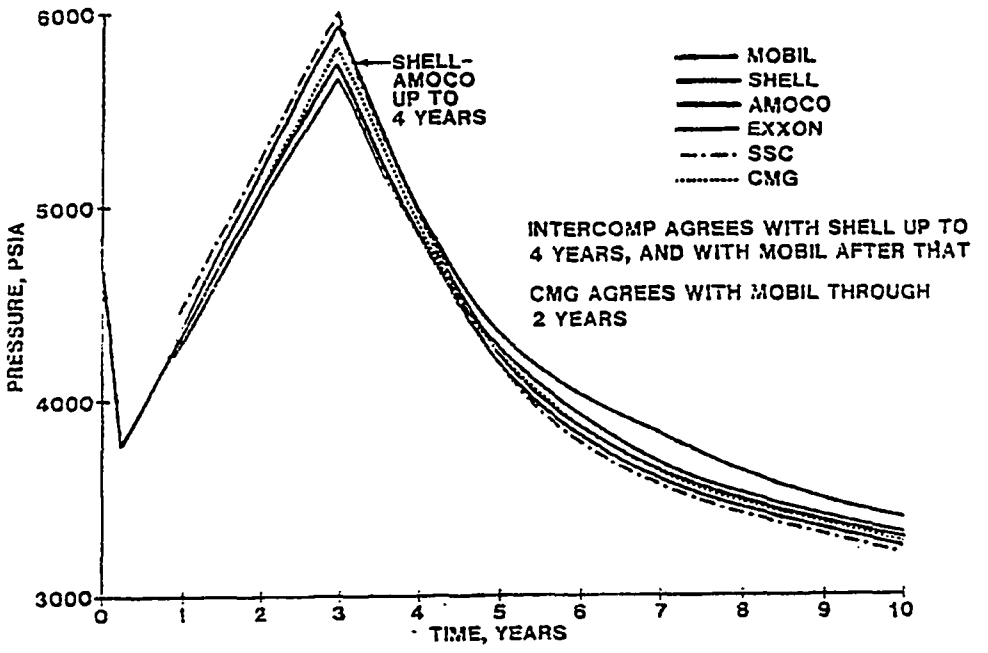


Figure 9.25: Odeh's Published Model

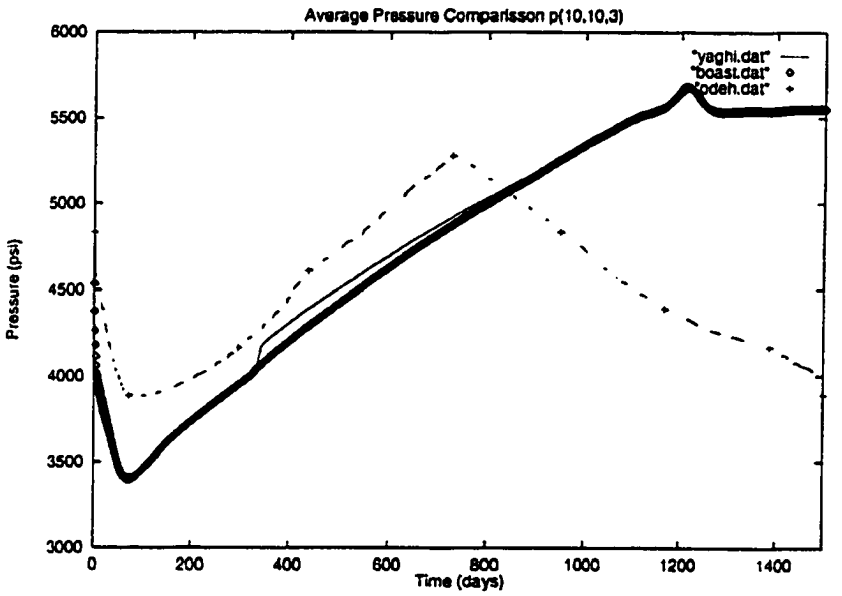


Figure 9.26: New Model versus Published Models

Comparing our FEM well vicinity pressure values with those from BOAST II, we actually improved the simulation, Figures 9.26. Based on flow principles, we conclude that our results are sound. We believe that with accurate input to our FEM model the answers would be more accurate.

Our triphasic flow system of equations is classified as a hyperbolic system because its eigenvalues are all negative. Peaceman [8] reviewed hyperbolic equations, solved sample problems, and concluded that this type of system is expected to exhibit either numerical dispersion or some oscillatory behavior. The influence of nonlinearity in the system was not explained by Peaceman. Though he suggested making a trade off between numerical dispersion and oscillatory behavior by adjusting the weighting factors. Both cannot disappear.

CHAPTER 10. SUMMARY

The finite element wellblock vicinity model was developed and interfaced with BOAST II. Our method of interfacing routines with BOAST II is unique and cost effective because it is not necessary for our model to return to BOAST II for any static data once the initial reservoir description was obtained. The data required was in the original BOAST II Fortran COMMON statements.

The simulation results obtained are consistent with reservoir engineering principles. To conduct the simulation, we used two data files furnished by NIPER. After simulation of approximately 1300 days of fluid flow, our model exhibited oscillations and the system became unstable. An improvement resulted from an improvement in the numerical representation of the activity and the use of improved boundary conditions. The specification of boundary conditions at a well had always been an active research problem.

The Case 1 dataset from NIPER was used to test our one layer 3-D model with two phases (oil and gas). We modified the equations to describe a three layer 3-D model, Case 2, with 3-phase flow (oil, gas, and water) in the wellblock. The Case 2 dataset, also from NIPER, had a gas injection well and a production well that produced from the deepest layer only (layer #3). Complicating matters, the production well has zero flux boundary conditions on two sides (east and south).

CHAPTER 11. CONCLUSION AND RECOMMENDATIONS

A numerical 3D-FEM solution for the simulation of multiphase fluid flow in porous media was developed. Such problems continue to be a difficult endeavor. We implemented this numerical solution into an existing simulator to model the vicinity of the well.

We implemented our wellbore model solution in a manner, which can be interfaced, in a similar way, into other reservoir simulators. For this research, our wellbore model was successfully interfaced with the Department of Energy's black oil simulator BOAST II.

Our wellbore model was tested with two (standard) data sets supplied by the United States Department of Energy - National Institute for Petroleum and Energy. For the first case, there were two liquid phases (oil and water) in the reservoir, which is often true for the primary recovery stage. Gas was dissolved in the oil and water phases. Figures 9.4 and 9.5 present the reservoir pressure versus time computed from our model. These reservoir pressures are comparable to what would be expected from this rate of production. Figure 9.6 shows that the computed saturation of oil and the computed saturation of gas are mirror images of one another. To further interpret these figures, after approximately 180 days gas appears from the oil phase and the bubble point pressure is reached. There is free gas in the wellbore even though the average reservoir pressure for the wellblock still more than the bubble point. The water phase on Figure 9.6 was constant (immobile) at 30% and

remained below the critical saturation for the whole simulation. Figures 9.7, 9.8, and 9.9 are the simulation results for node 114 located in the first quadrant on the second band and also shown on Figure 9.3. These nodal results are produced for every node shown on Figure 9.3. The reservoir pressure increases as the distance from the wellbore increases.

For test case 2 there were two wells in the reservoir, a gas injection well and a production well. This reservoir has three distinct strata. The production well is located at the corner of the reservoir with two sides of the wellblock having zero flux in all three strata. The injection well injects gas into the uppermost stratum and the production well produces from the lowest stratum. Figure 9.15 presents a history of the reservoir pressure at the production well. This pressure is initially drops and gas is below the bubble point before the pressure begins to increase due to the injected gas from the injection well to the wellblock. Figure 9.16 presents a history of the saturations at node 306 in the first quadrant second band shown on Figure 9.14. Both of these figures show oscillatory behavior after 1200 days of simulation. This instability is due partially to the inaccurate interface data passed at the interface at each BOAST II time step and used by our model.

To demonstrate the additional applications of our model Figures 9.18 - 9.24 were produced. These results are not possible with BOAST II. These figures are the simulation results for node 305 in the first quadrant second band. These nodal results are produced for every node shown on Figure 9.3. Figure 9.18 is the pressure versus distance from the wellbore at node 305 at the first day of simulation. Figures

9.19 and 9.20 also show the pressure history at node 305 after 1000 days and 1500 days of simulation respectively. The pressure decreases as the distance from the wellbore increases. Figure 9.20 compares the pressure history at node 305 after 1, 1000, and 1500 days of simulation. Figures 9.21 - 9.24 show results that cannot be provided by BOAST II either. These figures show a comparison of pressure at various nodes in the wellblock.

A comparison of our wellbore model using BOAST II with published results from several commercial reservoir simulators is presented on Figures 9.25 and 9.26. Our wellbore model improved these results over BOAST II alone. However, the results from our wellbore model were not as accurate as the commercial simulator results. The reason for this is that BOAST II was badly in error and this error is passed through the interface to our model. Although our model improved these results, the error in BOAST II continues to negatively affect the results throughout the simulation. Unfortunately, the SPE paper [12] results presented were much more accurate than BOAST II and our model.

Additionally, parallel experiments were conducted using an IBM SP/2, a cluster of IBM RS/6000 computers, and an SGI parallel computer. These parallel experiments indicated that the most intensive computer use was in our solver. However, the other parts of the calculation do not use significant computing. Because parallel solvers are ready available, developing another parallel solver was considered of little value. The decision was to concentrate the parallel implementation on other parts of the problem and use a publicly available parallel solver.

A prototype was developed [91,92] that displays the simulator's results in a table format and dynamically animates these results using Java or VRML to use the world wide web (with its distributed nature) for reservoir simulation. We suggested using a web-enabled database engine, such as Oracle, to read in the simulator's output data, store it in tables, then serve it to the user.

With this, users can locate needed data more efficiently in a presentable format. Furthermore, it is possible to make the simulation process interactive. After a user reviews the results, s/he can modify certain input data and proceed with the simulation. Some of the benefits to this approach are to view simulation results from any part of the world, to display these results in a native language, to visualize results interactively, and to distribute data and results on many different computers (distributed computing).

Even though our boundary conditions worked properly, our model is more sensible to the conditions used than FDM is. The use of 3D dynamically adaptive grid refinement; the use of fully implicit solver with FEM for the entire reservoir (i.e. rewrite BOAST II); and the use of iterative solvers would produce an excellent reservoir simulation.

Several hurdles remain to be overcome before parallel computing may become commonplace as a tool for applications. These are: load balancing, a data structure for the reservoir data, and adapting of the mesh. An active research area is in domain decomposition techniques for these problems. Another area is the development of 3D dynamic adaptive grid refinement. A third area of active research is

the development of robust parallel solvers for complex systems of equations, like those FEM equations found in Chapter 5.

Iterative solvers, such as GMRES, SOR, and bi-CG, have parallel implementations and would be an excellent improvement to BOAST II. Any improvement in the implementation of these solvers on shared-memory systems and distributed-memory systems would improve parallelism. The matrix-vector products used in these simulations are easily parallelized on shared-memory system by splitting the matrix into strips corresponding to the vector segment. Depending on bandwidth of the stiffness matrix, communication between processors may lead to communication bottlenecks.

The following are the major contributions of this research:

- The 3-D, 3-phase FEM equations, presented in Chapter 9, are original and to the best of our knowledge have not appeared in print. These equations have been tested with computer programs and are consistent. A manuscript is in preparation.
- The wellblock model, presented in Chapters 5 and 6, is a significant improvement to simulation results. Since the region around any well is the most active, an accurate model for this has an important value in the study of its flow [93].
- The web-based multimedia prototype, presented in Chapter 8, for interactive visualization of the simulation results has not been implemented

before. Two manuscripts have been published and presented at conferences [90,91].

- The successful coupling of an FEM model with an FDM model, presented in Chapter 6, is original. This is a difficult numerical problem. A manuscript to published this effort is in preparation.

BIBLIOGRAPHY

[1] Khalid Aziz and Antonin Settari. Petroleum Reservoir Simulation. Applied Science Publishers, Ltd, London, (1979).

[2] R. E. Ewing (Editor). The Mathematics of Reservoir Simulation. SIAM, (1983).

[3] Donald J. Morton, Jr. An Adaptive Finite Element Methodology for the High-Performance Computer Simulation of Multiphase Flow Processes (Ph.D.Dissertation). Louisiana State University, Baton Rouge, (1994).

[4] D.W. Peaceman. Fundamentals of Numerical reservoir Simulation. Elsevier, (1977).

[5] Fanchi, Harpole, and Bujinowski, "BOAST: A Three-Dimensional, Three-Phase Black Oil Applied Simulation Tool," Report, Department of Energy (Contract Number DE-AC19-80BC10033), (September 1982).

[6] H.B. Crichlow, "Modern Reservoir Engineering - a Simulation Approach," Prentice Hall, Englewood Cliffs, New Jersey (1977).

[7] X. Gu, "A Finte Element Simulation System in Reservoir Engineering," Thesis, LSU, 1996.

[8] K. Wang, "An Oil Reservoir Simulation by Using Finite Element Method and Boundary Conditions," Theis, LSU, 1997.

[9] G. Gottardi and E Mesin. "A Two-Phase Finite-Element Program for Displacement Simulation Processes in Porous Media." Computers and Geosciences, 12(5). 1986.

[10] L.C. Young, "A Finite Element Method for Reservoir Simulation," Soc. Pet. Eng. Journal, 21, pp. 115-128 (1981).

[11] C.S. Matthews and D.G. Russell. "Pressure Buildup and Flow Tests in Wells," SPE Monograph Series, 1967.

[12] D. Babu, A. Odeh, A. Al-Khalifa, and R. McCann, "The Relation Between Wellblock and wellbore Pressure in Numerical Simulation of Horizontal Wells - General Formulas for Arbitrary Well Locations in Grids," SPE 20161, 1989.

[13] P. Blair and C. Weinaug, "Solution of Two-Phase Problems Using Implicit Difference Equations," Society of Petroleum Engineers Journal, 246, pp. 417-424, 1969.

[14] R. MacDonald and K. Coats, "Methods for Numerical Simulation of Water and Gas Conning," Society of Petroleum Engineers Journal, pp. 425-436, 1970.

[15] A. Behie, et al. "Composition of Nested Factorization, Constrained Pressure Residual, and Incomplete Factorization Preconditionings," SPE 13531, the 8th SPE Symposium on Reservoir Simulation, Dallas, February 10-13, 1985.

[16] E. Nacul, "Use of Domain Decomposition and Local Grid Refinement in Reservoir Simulation," Ph.D. Thesis, Stanford University, 1991.

[17] A. Spivak and K. Coats, "Numerical Simulation of Coning Using Implicit Production Terms," Society of Petrolum Engineers, pp. 257-267, Sept. 1970.

[18] J. Letkeman and R Ridings, "A Numerical Coning Model," Society of Petroleum Engineers Journal, pp. 418-424, December 1970.

[19] J. Chappelear and W. Rogers, "Some Practical Considerations in the Construction of a Semi-Implicit Simulator," Society of Petroleum Engineers Journal, pp. 216-220, June 1974.

[20] O. C. Shaw, "The Treatment of Wells, Faults, and Other Singularities in a Black-Oil, Finite-Element Reservoir Simulator," SPE 25246, the 12th SPE Symposium on Reservoir Simulation, New Orleans, La., Feb. 28 - Mar. 3, 1993.

[21] K.H. Coats. "An equation-of-state Compositional Model." Society of Petroleum Engineers Journal, 20, pp. 363-376, 1980.

[22] L. Young and R. Dtephenson, "A Generalized Compositional Approach for Reservoir Simulation," Society of Petroleum Engineering Journal, 1983.

[23] J. Bramble, J. Pasciak, and A. Schatz, "An Iterative Method for Elliptic Problems on Regions Partitioned into Substructures," Mathematics of Computation, 46, pp. 361-369, April 1986.

[24] R. Bhogeswara, "Simulation of Numerically Intensive Multiphase Flow in Porous Media on Advanced Parallel Computers Using Domain Decomposition and Multigrid Methods," Ph.D. Thesis, University of Houston, 1991.

[25] M. Kaiho, M. Ikegawa, and C. Kato. "Parallel Overlapping Scheme for Viscous Incompressible Flows." International Journal for Numerical Methods in Fluids, Vol. 24, 1341-1352, 1997.

[26] M. Ikegawa, et al. "FEM/FDM Composite Scheme for Viscous Incompressible Flow Analysis." Computational Methods for Applied Mechanical Engineering, 112, pp. 149-163, 1994.

[27] K. Nakahashi and S. Obayashi, "FDM-FEM Zonal Approach for Viscous Flow Computations over Multiple Bodies," AIAA paper No. 87-0604, 1987.

[28] M.K. Deb, P.R. Mahendu, et. al , "A New Generation Solution Adaptive Reservoir Simulator," SPE 30720, Annual Technical Conference & Exhibition, Dallas, Texas (Oct. 22-25,1995).

[29] A. Coutinho, J. Alves, E. Garcia, and A. Loula, "Solution of Miscible and Immiscible Flows Employing Element-by-Element Iterative Strategies," SPE 27050, Proc. 3rd SPE Latin American/Caribbean Petroleum Engineering Conference (1994).

[30] A. Coutinho and J. Alves, "Parallel Finite Element Solution of Miscible Displacements in Porous Media," SPE 37399, SPE Journal, (Dec. 1996).

[31] A. Coutinho, A. Alves, L. Landau, and N. Ebecken, "A Dynamic Mesh Partition Algorithm forthe FE Solution of Miscible Displacement in Porous Media," Research Report, Department of Civil Engineering, COPPE/Federal University of Rio de Janeiro, 1992.

[32] A. Coutinho, J. Alves, L. Landau, and N. Ebecken, "A Dynamic Mesh Partition Algorithm for the Finite Element Solution of Two-Phase Immiscible Flow in Oil Reservoirs," Finite Elements in Fluids: New Trends and Applications, Editor K. Morgan et al., CIMNE/Pineridge Press. Barcelona, Vol. II, pp. 856-865, 1993.

[33] S. Malta, A. Loula, and E. Garcia, "A Post-Processing Technique to Approximate the Velocity Field in Miscible Displacement Simulation," Contemporary Mathematics, 8, pp 239-268, 1995.

[34] R. Bhogeswara and J. Killough. "Parallel Linear Solvers for Reservoir Simulation: A Generic Approach for Existing and Emerging Computer Architectures." SPE 25240, The 12th SPE Symposium on Reservoir Simulation, New Orleans, Louisiana, February 28 - March 3, 1993.

[35] J. Killough, "The Application of Parallel Computing to the Flow of Fluids in Porous Media," Computers in Chemical Engineering, Vol. 19, No. 6/7. pp. 775-786, 1995.

[36] M. Zhiyuan, et. al, "Simulation of Black Oil Reservoir on Distributed Memory Parallel Computers and Workstation Cluster," SPE 29937, the International Meeting on Petroleum Engineering, Beijing, PR China, Nov. 14-17, 1995.

[37] J. Wallis, J. Foster, and R. Kendall, "A New Parallel Iterative Linear Solution Method for Large-Scale Reservoir Simulation," SPE 21209, the 11th SPE Symposium on Reservoir Simulation, Anaheim, Ca., Feb 17-20, 1991.

[38] J. Killough. "Is Parallel Computing Ready for Reservoir Simulation ?" SPE 26634, The 68th Annual Technical Conference and Exhibit of SPE, Houston, Texas, Oct. 3-6, 1993.

[39] O. Zienkiewics. The Finite Element Method in Engineering Science, 3rd. Edition, McGraw-Hill, London, 1977.

[40] G. Strang and G. Fix. An Analysis of the Finite Element Method. Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

[41] R. Cook. Concepts and Applications of Finite Element Analysis, 2nd Edition. Wiley, New York, 1981.

[42] S. Rao. The Finite Element Method in Engineering. Pergamon Press, Oxford, 1982.

[43] D. Burnett. Finite Element Analysis: from Concepts to Applications. Addison Wesley, 1988.

[44] F. Kucuk and W. Brigham, "Transient Flow in Elliptical Systems," SPE Journal, pp. 401-410, December 1979.

[45] H.S. Price, et. al., "Numerical Methods of Higher Order Accuracy for Convection Equations," Soc. Pet. Eng. Journal, pp. 293-303 (1968)

[46] G.F. Pindu, "A Galerkin-Finite Element Simulation of Groundwater Contamination on Long Island, New York," Water Resources Research, 9, pp. 1657-1669 (1973).

[47] P.S. Hayak and G.F. Pindu, "Computational Methods in Subsurface Flow," Academic Press, New York, N.Y. (1983).

[48] W.L. Miranker, "A Survey of Parallelism in Numerical Analysis," SIAM Review, Vol. 13, No. 4, October 1971, pp. 524-547.

[49] S. Scott, R. Wainwright, et. al, "Application of Parallel Computers to Reservoir Simulation," the 9th SPE Symposium on Reservoir Simulation," San Antonio, Texas, Feb. 1-4, 1987.

[50] M. Chien, M. Wasserman, et. al., "The Use of Vectorization and Parallel Processing for Reservoir Simulation," SPE 16025, the 9th SPE Symposium on Reservoir Simulation," San Antonio, Texas, Feb. 1-4, 1987.

[51] J. Wheeler and R. Smith, "Reservoir Simulation on a Hypercube," SPE 19804, the 64th SPE Annual Conf. and Exhib., San Antonio, Texas, October 1989.

[52] J. Barua and R. Horne, "Improving Performance of Parallel (and Serial) Reservoir Simulators," SPE 18408, the 10th Symposium on Reservoir Simulation, Houston, Texas, Feb. 6-8, 1989.

[53] A. Odeh, "Comparison of Solutions to a Three-Dimensional Black-Oil Reservoir Simulation Problem," SPE 9723, SPE Journal, January 1981.

[54] O. Pedrosa, "Use of Hybrid Grid in Reservoir Simulation," Ph.D. Thesis, Stanford University, 1985.

[55] J. Chappelear and J. Nolen, "Second Comparative Solution Project: A Three-Phase Coning Study," SPE 10489, the Sixth SPE Symposium on Reservoir Simulation, New Orleans, Jan. 31- Feb. 3, 1982.

[56] H. Weinstein, J. Chappelear, and J. Nolen, "Second Comparative Solution Project: A Three-Phase Coning Study," SPE Journal, March 1986.

[57] M. Chang, P. Sarathi, et. al., "User's Guide and Documentation Manual for BOAST-VHS for the PC," NIPER-542, NIPER, January 1992.

[58] R. Barrett, et. al. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. SIAM, 1994.

[59] G. Golub and C. Van Loan. Matrix Computations. The John Hopkins University Press, Baltimore, 1989.

[60] L. N. Smith, et al. Vectorization of a Reservoir Simulator. F.A. Matsen and T. Tajima, editor, Supercomputers: Algorithms, Architectures, and Science Computation.University of Texas Press, (1986).

[61] B. Galerkin. On Some Problems of Elastic Equilibration of Rods and Plates. Vestnik Ingenerov, SPB 19, (1915).

[62] N. Bubnov. The Report About Professor Timoshenko's Articles Honoured By Zhuravsky's Prize. Sbornik Institute Putey Soobshcheniia, SPB 18, No. 4, 1913.

[63] L. Oganesjan and V. Rivkind. On the Finite Element Method Research in St. Petersburg. Finite Element Methods: Fifth Year of the Courant Element, Vol. 164. Marcel Dekker, (1994).

[64] E. Hinton and D. Owen. Finite Element Programming. Academic Press, (1992).

[65] L.P. Dakei. Fundamentals of Reservoir Engineering, Vol. 8. Elsevier, (1980).

[66] P. Bansal, J. Harper, A. McDonald, E. Moreland, A. Odeh, and R. Trimble. A Strongly Coupled Fully Implicit, Three-Dimensional, Three-Phase Reservoir Simulator. SPE 8329, 54th SPE Annual Technical Conference, Las Vegas, (1979).

[67] R. Collins. Flow of Fluids Through Porous Materials. Van Nostrand-Reinhold, New York, (1961).

[68] R. Kendall, G. Morrel, D. Peaceman, W. Silliman, and J. Watts. Development of a Multiple Application reservoir Simulator For Use on a Vector Computer. SPE 11483, SPE Middle East Oil Technical Conference, Manama, Bahrain, (1983).

[69] Philippe Devloo, J.T. Oden and T. Strouboulis, "Implementation Of An Adaptive Refinement Technique For The {SUPG} Algorithm," Computer Methods in Applied Mechanics and Engineering, (1987).

[70] A.K. Gupta, "A Finite Element For Transition From a Fine Grid To A Coarse Grid," International Journal For Numerical Methods In Engineering, (1978).

[71] D.J. Morton, J.M. Tyler, and J.R. Dorroh, "A New 3-D Finite Element For Adaptive h-Refinement in 1-Irregular Meshes," International Journal For Numerical Methods In Engineering, accepted for publication (1995).

[72] Lance H. Hebert, "An experimental study of the effects of water coning on a horizontal well in a bottom drive reservoir," Master's thesis, Louisiana State University, Baton Rouge, Louisiana, (May 1990).

[73] K.C. Reddy and J.N. Reddy, "Finite Element Solver for 3-D Compressible Viscous Flows," Interim Report, NASA/MSFC (Contract No. NAS8-36555), (December 1986).
[74] J. Dongarra, A. Karp, K. Kennedy, and D. Kuck, "1989 Gordon Bell Prize Report," IEEE Software, May 1990.

[75] P.A. Forsyth, "A Control Volume Finite Element Method for Local Mesh Refinement," Paper SPE 18415 presented at the 1989 Symposium on Reservoir Simulation, Houston, Texas (Feb. 6-8, 1989).

[76] F.X. Deimbacher and Z.E. Heineman, "Time-Dependent Incorporation of Locally Irregular Grids in Large Reservoir Simulation Models," Paper SPE 25260 presented at the 1993 SPE Symposium on Reservoir Simulation, New Orleans, Louisiana (Feb. 28 - Mar. 3, 1993).

[77] L.J. Durlofsky, "A Triangle Based Mixed Finite Element - Finite Volume for Modeling Two Phase Flow Through Porous Media," Journal of Computational Physics, 105, pp. 252-266 (1993).

[78] M. Behr and T. Tezduyar, "Finite Element Solution Strategies for Large-Scale Flow Simulations," Comp. Meth. Appl. Mech. Engr., 112, pp. 3-24 (1994).

[79] A. Brooks and T. Hughes. "Streamline Upwind/Petrov-Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations." Computer Methods in Applied Mechanics and Engineering, 32:199-259, 1982.

[80] P. Gresho and R. Lee. "Don't Supress the Wiggles - they're trying to tell you somethis !' T.J.R Hughes, editor, finite element methods for convection Dominated Flow," 37-61, N.Y 1979.

[81] Y. Sukirman and R. Lewis. "A Finite Element Solution of a Fully Couples Implicit Formulation for Reservoir Simulation," International Journal for Numerical and Analytical Methods in Geomechanics, 17:698, 1993.

[82] H. Van Poolen, E. Breitenbach, and D. Thurnau. "Treatment of Individual Wells and Grids in Reservoir Modeling." SPE Journal, pp. 341-346, December 1968.

[83] D. Peaceman. "Interpretation of Well-Block Pressure in Numerical Reservoir Simulation." SPE Journal, pp. 183-194, June 1978.

[84] J.A. Trangestein and J.B. Bell. "Mathematical Structure of Compositional Reservoir Simulation," SIAM Journal on Scientific Statistical Computing, 10, pp. 817-845, 1989.

[85] J. Douglas, D. Peaceman and H. Rachford. "A Method of Calculating Multidimensional Immiscible Displacement," Transactions of AIME, vol. 216, 1959.

[86] K. Schwabe and J. Brand, "Prediction of Reservoir Behavior Using Numerical Simulators," SPE 1857, SPE 42nd Annual Meeting, Houston, October 1967.

[87] R. Timble and A. McDonald, "A Strongly couples, fully implicit, three-dimensional, three-phase well coning model," Society of Petroleum Engineering Journal, 21, pp. 454-458, 1981.

[88] A. Coutinho and J. Alves." Parallel Finite Element Simulation of Miscible Displacements in Porous Media." SPE 37399, SPE Journal, December 1996.

[89] J Tyler, "3D Three Phase Finite Element Method in Porous Media," Technical Report, LSU, 1994.

[90] H. Yaghi and J. Tyler, "Setting Up a Web Server for Interactive Engineering Applications," WebNet World Conference, Toronto, Canada, Oct. 31, 1997.

[91] H. Yaghi and J. Tyler, "Multilingual Multimedia Visualization of Hydrocarbon Reservoir Simulation," 6th International Conference on Multilingual Computing, Cambridge, UK, April 11-13, 1998.

[92] H. Yaghi and J. Tyler, "Multidimensional Multiphase Flow of Compressible Fluids," Water Resources '97, Beirut, Lebanon, June 16-18, 1997.

[93] H. Yaghi and J. Tyler, "Triphasic Fluid Flow Simulation: IMPES vs. Fully-Implicit 3D FEM," Parallel CFD '96 Conference, Capri, Italy, May 20-23, 1996.

[94] H. Yaghi and J. Tyler, "3-D Finite Element IMPES Formulation of Oil Reservoir Simulation," 5th ICEMC, Cambridge, UK, April 11-13, 1996.

[95] H. Yaghi and J. Tyler, "Reservoir Simulation Using Finite Element Method and Multimedia," Bahrain International Conference on Computers in Education & Industry, Manama, Bahrain, November 1995.

[96] H. Yaghi and J. Tyler, "Notes on Adaptive Mesh Generation Using 3D Three Phase Finite Elements," ASEE Gulf South Conference, Beaumont, Texas, March 28-30, 1995.

NOMENCLATURE

Symbols and abbreviations used repeatedly are defined below

c_t	total compressibility of rock and fluid
k, $k_{x,y,z}$	permeability, or the component of the permeability tensor
k_{rl}	relative permeability of phase l
k_{ro}	oil relative permeability
P_c	capillary pressure
P_{cog}	oil-gas capillary pressure
P_{cow}	oil-water capillary pressure
P	pressure
P_p	phase pressure
Q_i	injection flow rate
Q_p	production flow rate
r_w	radius of the well
S_g	gas saturation
S_l	saturation of phase l
S_o	oil saturation
S_w	water saturation
v_i	phase velocity vector
x_i	value of x at any grid point i

Symbols

Δt	time increment
$(\Delta x, \Delta y, \Delta z)$	mesh size in feet
λ_i	eigenvalues
ϕ	porosity
μ	viscosity
ρ	fluid density

Subscripts

g	gas
o	oil
w	water

APPENDIX A

SOURCE CODE

```
C   NIPER BLACK OIL PRIMARY AND SECONDARY RECOVERY MODEL
C    (HORIZONTAL/SLANT WELL & CONVENTIONAL VERTICAL WELL MODELS)
C
C      Modified by: Husam M. Yaghi
C      Last Update: December 23, 1997
C
C      --- AN UPDATED SLANT WELL MODEL  INSTALLED
C      --- 'RESTART' SUBROUTINE  INSTALLED
C      --- BOTH 'PRESSURE' AND 'MAX. RATE' CONSTRAINTS APPLIED WHEN
C          IMPLICIT PRESSURE OPTION ACTIVATED
C      --- SAME AS "BESTVHS.FOR" EXCEPT THAT INPUT/OUTPUT FILE NAMES
C          NEED TO BE SPECIFIED BY SUBMITTING A COMMAND FILE
C    SEPTEMBER 1989
C ****** DIMENSIONS SPECIFIED IN "PARAMETR.FOR" ******************
C
      INCLUDE 'PARAMETR.FOR'
      INCLUDE 'PARAMETR.FEM'
      REAL KX,KY,KZ,KROT,KRWT,KRGT,MUOT,MUWT,MUGT,MCFG,MCFG1,MCFGT,
     & MCFGI,MBEO,MBEW,MBEG,MIN
      CHARACTER*2 TITLE(80),FNAME,MNAME,TAG
      CHARACTER*5 WELLID
      CHARACTER*4 LNAME
      DIMENSION OOIP(NZ),OWIP(NZ),ODGIP(NZ),OFGIP(NZ)
      COMMON /IDEN/ WELLID(NW)
      COMMON /IO/ NI,NO
      COMMON /BUBBLE/ PBO,VSLOPE,BSLOPE,RSLOPE,PMAXT,IREPRS,
     & RHOSCO,RHOSCG,RHOSCW,MSAT,MPOT,MPWT,MPGT,PBOT(NX,NY,NZ)
      COMMON /COEF/ AW(NX,NY,NZ),AE(NX,NY,NZ),AN(NX,NY,NZ),
     & AS(NX,NY,NZ),AB(NX,NY,NZ),AT(NX,NY,NZ),E(NX,NY,NZ),B(NX,NY,NZ)
      COMMON /SARRAY/ PN(NX,NY,NZ),
     & SON(NX,NY,NZ),SWN(NX,NY,NZ),SGN(NX,NY,NZ),
     & SO1(NX,NY,NZ),SW1(NX,NY,NZ),SG1(NX,NY,NZ),
     & A1(NX,NY,NZ),A2(NX,NY,NZ),A3(NX,NY,NZ),
     & SUM(NX,NY,NZ),GAM(NX,NY,NZ),QS(NX,NY,NZ)
      COMMON /PERM/ KX(NX,NY,NZ),KY(NX,NY,NZ),KZ(NX,NY,NZ)
      COMMON /TRAN/ TX(NX+1,NY,NZ),TY(NX,NY+1,NZ),TZ(NX,NY,NZ+1)
      COMMON /ELEV/ EL(NX,NY,NZ)
      COMMON /PRTP/ P(NX,NY,NZ)
      COMMON /PRTS/ SO(NX,NY,NZ),SW(NX,NY,NZ),SG(NX,NY,NZ)
      COMMON /SPVT/ SAT(NTE),KROT(NTE),KRWT(NTE),KRGT(NTE),PCOWT(NTE),
     & PCGOT(NTE),POT(NTE),MUOT(NTE),BOT(NTE),BOPT(NTE),RSOT(NTE),RSOPT
     & (NTE),PWT(NTE),MUWT(NTE),BWT(NTE),BWPT(NTE),RSWT(NTE),RSWPT(NTE),
     & PGT(NTE),MUGT(NTE),BGT(NTE),BGPT(NTE),CRT(NTE)
      COMMON /SRATE/ PID(NW,NL),PWF(NW,NL),PWFC(NW,NL),KIP(NW),
     & GMO(NW,NL),GMW(NW,NL),GMG(NW,NL),LAYER(NW),QVO(NW),
     & QVW(NW),QVG(NW),QVT(NW),CUMO(NW,NL),CUMW(NW,NL),CUMG(NW,NL)
      COMMON /VOLFAC/ BO(NX,NY,NZ),BW(NX,NY,NZ),BG(NX,NY,NZ)
      COMMON /RATE/ QO(NX,NY,NZ),QW(NX,NY,NZ),QG(NX,NY,NZ)
```

```
      COMMON /TERM1/ GOWT(NX,NY,NZ),GWWT(NX,NY,NZ),GGWT(NX,NY,NZ),
     & OW(NX+1,NY,NZ),OE(NX+1,NY,NZ),WW(NX+1,NY,NZ),WE(NX+1,NY,NZ),
     & OS(NX,NY+1,NZ),ON(NX,NY+1,NZ),WS(NX,NY+1,NZ),WN(NX,NY+1,NZ),
     & OT(NX,NY,NZ+1),OB(NX,NY,NZ+1),WT(NX,NY,NZ+1),WB(NX,NY,NZ+1)
     & ,GW(NX+1,NY,NZ),GE(NX+1,NY,NZ),
     & GS(NX,NY+1,NZ),GN(NX,NY+1,NZ),
     & GT(NX,NY,NZ+1),GB(NX,NY,NZ+1)
      COMMON /TERM2/ QOWG(NX,NY,NZ)
      COMMON /COMPRS/ CT(NX,NY,NZ)
      COMMON /PORE/ VP(NX,NY,NZ)
      COMMON /VECTOR/ DX(NX,NY,NZ),DY(NX,NY,NZ),DZ(NX,NY,NZ)
      COMMON /IQN/ IQN1(NW),IQN2(NW),IQN3(NW)
      COMMON /IQH/ IQH1(NW,NL),IQH2(NW,NL),IQH3(NW,NL),COND(NW)
      COMMON /CODE/ KSM1,KSN1,KCO1,NN,FACT1,FACT2,TMAX,KSOL,MITER,
     & OMEGA,TOL,TOL1,KSN,KSM,KCO,KTR,KCOFF,DSMAX,DPMAX.WORMAX,
     & GORMAX,PAMIN,PAMAX
      COMMON /ADD1/ IM,JM,KM,ETI,FT,FTMAX
      COMMON /ADD2/ COP,CWP,CGP,CWI,CGI
      COMMON /PSCNTL/ KPI,KSI
      COMMON /MBE/ MBEO,MBEW,MBEG
      COMMON /NUMBER/ II,JJ,KK
      COMMON /VOL/ SCFO,SCFW,SCFG,SCFG1,STBO,STBW,MCFG,MCFG1,MCFGT,
     & MCFGI,STBOI,STBWI,RESVOL
      COMMON /BAL/ OP,WP,GP,WI,GI,PAVG0.PAVG,OPR,WPR,GPR,WIR,GIR,CWOR,
     & WOR,CGOR,GOR
      COMMON /PRTCNT/ IWLCNG,ICHANG.IWLREP,ISUMRY,
     & IPMAP,ISOMAP,ISWMAP,ISGMAP,IPBMAP
      COMMON /IO2/ NO2
      COMMON /CHRCTR/ FNAME,MNAME(NUM),LNAME,TAG
      COMMON /TTEST/ NUMPRD,SONTVL
      COMMON /SWTCH/ NGRSW,NTRSW,NRESTART,NTS
      COMMON /R1XYZ/ R11X,R21X,R31X,R41X,R11Y,R21Y,R31Y,R41Y,
     & R11Z,R21Z,R31Z,R41Z
      COMMON /T1XYZ/ T1X,T2X,T3X,T4X,T1Y,T2Y,T3Y,T4Y,T1Z,T2Z,T3Z,T4Z
      COMMON /DXYZ/ DX0,DXP,DXM,XW,DY0,DYP,DYM,YW,DZ0,DZP,DZM,ZW
      COMMON /RXYZ/ RX,RY,RZ,AX,AY,AZ,I,J,K
      COMMON /COUNT/ N1READ,N2READ
      COMMON /RS/ ROS
      COMMON /DECLINE/ DEC(NW,NL)
      COMMON /FEMRATE/      QWWW(NZ), QWE(NZ), QWS(NZ), QWN(NZ),
     &  QWT(NZ), QWB(NZ),
     &  QOW(NZ),QOE(NZ),QOS(NZ),QON(NZ),QOT(NZ),QOB(NZ),
     &  QGW(NZ),QGE(NZ),QGS(NZ),QGN(NZ),QGT(NZ),QGB(NZ)
      COMMON/MESHPTS/ i1(num_elemts),i2(num_elemts),
     & i3(num_elemts),i4(num_elemts), r1
c#######################################################################

      common/glstiff/stiff(num_nodes*nzp,num_nodes*nzp),
     &  fl(num_nodes*nzp),xl(num_nodes*nzp),ipvt(num_nodes*nzp),
     &  sofl(num_nodes*nzp),sofl1(num_nodes*nzp),sofl2(num_nodes*nzp),
```

```
     & sgfl(num_nodes*nzp),sgfl1(num_nodes*nzp),sgfl2(num_nodes*nzp),
     & swfl(num_nodes*nzp),swfl1(num_nodes*nzp),swfl2(num_nodes*nzp)

      common/stiff0/stiff0(num_nodes*nzp,num_nodes*nzp),
     & sostiff(num_nodes*nzp,num_nodes*nzp),
     & sgstiff(num_nodes*nzp,num_nodes*nzp),
     & swstiff(num_nodes*nzp,num_nodes*nzp)

      common/oldpwg/pold(num_nodes*nzp),swold(num_nodes*nzp),
     & soold(num_nodes*nzp), sgold(num_nodes*nzp)

      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)

c#####################################################################
      IFEM = 2
      i_signal = -1
      NI=5
      NO=6
      N1READ=0
      N2READ=0
      DIFCNT = 0
      DIFTOT = 0
C     OPEN(NI,FILE='tyler2.dat',STATUS='OLD')
          OPEN(NI,FILE='tyler2.dat')
          open(30,file='Set1-fem-s.out')
          open(60,file='Set1-fem-p.out')
       open(61,file='nodal-p.out')
       open(62,file='nodal-so.out')
       open(63,file='nodal-sw.out')
       open(64,file='nodal-sg.out')
          open(71,file='node285.out')
          open(72,file='node287.out')
          open(73,file='node300.out')
          open(74,file='node310.out')
          open(75,file='node350.out')
          open(76,file='node360.out')
          open(77,file='node370.out')
          open(78,file='node376.out')
          open(79,file='node380.out')
      OPEN(NO,FILE='boast.out')
      OPEN(14,FILE='boast.rates.out')

        t0=MCLOCK()

C
C**** ESTABLISH RESERVOIR AND BLOCK DIMENSIONS
```

```
C
      CALL GRID1

C
C**** ESTABLISH POROSITY AND PERMEABILITY AT EACH ZONE
C
      CALL PARM1

C
C**** CALCULATE INTERBLOCK TRANSMISSIBILITIES
C
      CALL TRAN1
C
C**** EMPIRICAL DATA
C
      CALL TABLE
C
C**** ESTABLISH INITIAL CONDITIONS
C
      CALL UINIT1
C
C**** SOLUTION METHOD, DEBUG PRINT, AND TIME-STEP CONTROL PARAMETERS.
C
      CALL CODES

      READ(NI,69)(TITLE(IH),IH=1,40)
69    FORMAT(40A2)
      D5615 = 1./5.615
      D288  = 1./288.
      D114  = 1./144.
      IW1=II/2+1
      JW1=JJ/2+1
      KW1=KK/2+1
       NMAX=80000
c      NMAX=NN+1
       ttltime=0.
      DO 1000 N=1,NMAX
C
C*** NLOOP DEFINED TO AVOID INDEX MODIFICATION WARNINGS IN
C   CALLS TO MATBAL, SOLMAT, LSOR.
      NLOOP=N
C
C**** RECURRENT DATA.
C
      IF(FT.LT.FTMAX) GO TO 46
           IF(N.EQ.1) THEN
      READ(NI,*) IWLCNG,ICHANG,IWLREP,ISUMRY,
     & IPMAP,ISOMAP,ISWMAP,ISGMAP,IPBMAP
      READ(NI,*) DAY,DTMIN,DTMAX,HOUR,MIN,SEC
           ENDIF
```

```
c    enforce delt = 0.01 day
c     DAY=1.00
          DAY=1.00
          DTMIN=DAY
          DTMAX=DAY
          HOUR=0.
          MIN=0.
          SEC=0.
     IF(IWLCNG.EQ.0) GO TO 45
     CALL NODES(NVQN,NVQNH,NVQNS)
          IWLCNG=0
c
#########################################################################
C    ESTABLISH WELL-BLOCK FEM MESH
C
     if (i_signal.lt.0) then
     CALL MESHGEN (IFEM)
     endif
c########################################################################
  45 DELT=DAY+HOUR/24.+MIN/1440.+SEC/86400.
     FTMAX=ETI+ICHANG*DELT
  46 IF(N.EQ.1)DELT0=DELT
     IF(N.EQ.1)GO TO 1050
c         DELT NOT CHANGED
          FACT1=1.
          FACT2=1.

     IF(DSMC.LT.DSMAX.AND.DPMC.LT.DPMAX)DELT=DELT*FACT1
     IF(DSMC.GT.DSMAX.OR.DPMC.GT.DPMAX)DELT=DELT*FACT2
     IF(DELT.LT.DTMIN)DELT=DTMIN
     IF(DELT.GT.DTMAX)DELT=DTMAX
     IF(ETI+DELT.GT.FTMAX)DELT=FTMAX-ETI
1050  FT=ETI+DELT

c    enforce delt = 0.01
c ==== Yaghi
          DELT= 10.00
     DO 99 J=1,NW
     DO 99 K=1,NL
 99  IF(KIP(J).EQ.-110)PWF(J,K)=PWF(J,K)*EXP(DEC(J,K)*DELT)

     IF(ETI+DELT*0.5.GE.TMAX) GO TO 1010
C****ITFLAG COUNTS THE NUMBER OF TIME STEP REPITITIONS.
     ITFLAG=0
C****REENTRY POINT FOR REPEATED TIME STEP.
1060  CONTINUE
     DIVI = 1./DELT
     IF(N.GT.1.OR.ITFLAG.GT.0) GO TO 105
     RESVOL=0.0
     SCFO=0.0
```

```
      SCFG=0.0
      SCFG1=0.0
      DO 102 K=1,KK
      OOIP(K)=0.
      OWIP(K)=0.
      ODGIP(K)=0.
      OFGIP(K)=0.
      DO 100 J=1,JJ
      DO 100 I=1,II
      PP=P(I,J,K)
      BPT=PBOT(I,J,K)
c#########################################################################
c  This needs to be changed to allow more than one layer  phi(#layers).
      if (i_signal.lt.0) then
c           if (i.eq.iqn1(ifem).and.j.eq.iqn2(ifem)) then
            if (i.eq.10.and.j.eq.10) then
            phi(k) = vp (i,j,k)
            endif
      endif
c#########################################################################
      VP(I,J,K)=VP(I,J,K)*DX(I,J,K)*DY(I,J,K)*DZ(I,J,K)
      RESVOL=RESVOL+VP(I,J,K)
C**** NOTE: WE ARE ASSUMING INITIAL PHI IS AT INITIAL RESERVOIR PRESSURE
      CALL INTPVT(BPT,RSLOPE,POT,RSOT,MPOT,PP,RSO)
      CALL INTERP(PWT,RSWT,MPWT,PP,RSW)
      CALL INTPVT(BPT,BSLOPE,POT,BOT,MPOT,PP,BO(I,J,K))
      CALL INTERP(PWT,BWT,MPWT,PP,BW(I,J,K))
      CALL INTERP(PGT,BGT,MPGT,PP,BG(I,J,K))
 1    FORMAT(5X,'RSO = ',F15.5,'RSW = ',F15.5,'BO = ',F15.5,'BW = ',
     & F15.5,'BG = ',F15.5)
      FF1=SO(I,J,K)/BO(I,J,K)
      FF2=SW(I,J,K)/BW(I,J,K)
      SCFO=SCFO+VP(I,J,K)*FF1
      SCFW=SCFW+VP(I,J,K)*FF2
      SCFG=SCFG+VP(I,J,K)*SG(I,J,K)/BG(I,J,K)
      SCFG1=SCFG1+VP(I,J,K)*(RSO*FF1+RSW*FF2)
      CALL INTERP(POT,BOPT,MPOT,PP,BODER)
      CALL INTERP(POT,RSOPT,MPOT,PP,RSODER)
      CALL INTERP(PWT,BWPT,MPWT,PP,BWDER)
      CALL INTERP(PWT,RSWPT,MPWT,PP,RSWDER)
      CALL INTERP(PGT,BGPT,MPGT,PP,BGDER)
      IF(PP.GT.PBOT(I,J,K))BODER=BSLOPE
      IF(PP.GT.PBOT(I,J,K))RSODER=RSLOPE
      CO=-(BODER-BG(I,J,K)*RSODER)/BO(I,J,K)
      CW=-(BWDER-BG(I,J,K)*RSWDER)/BW(I,J,K)
      CG=-BGDER/BG(I,J,K)
      CALL INTERP(PGT,CRT,MPGT,PP,CR)
      CT(I,J,K)=CR + CO*SO(I,J,K) +CW*SW(I,J,K) + CG*SG(I,J,K)
      OOIP(K)=OOIP(K)+D5615*.000001*SON(I,J,K)*VP(I,J,K)/BO(I,J,K)
      OWIP(K)=OWIP(K)+D5615*.000001*SWN(I,J,K)*VP(I,J,K)/BW(I,J,K)
```

```
      ODGIP(K)=ODGIP(K)+.001*.000001*(RSO*SON(I,J,K)*VP(I,J,K)/BO(I,J,K)
     & +RSW*SWN(I,J,K)*VP(I,J,K)/BW(I,J,K))
      OFGIP(K)=OFGIP(K)+.001*.000001*SGN(I,J,K)*VP(I,J,K)/BG(I,J,K)
      IF(KCO1.EQ.0)GO TO 100
   21 FORMAT(1X,3I3,8E15.6)
  100 CONTINUE
  110 FORMAT(/,1X,'LAYER',I3,' INITIAL FLUID VOLUMES:',
     & /,10X,'OIL IN PLACE (MILLION STB)',T60,F10.4,
     & /,10X,'WATER IN PLACE (MILLION STB)',T60,F10.4,
     & /,10X,'SOLUTION GAS IN PLACE (BILLION SCF)',T60,F10.4,
     & /,10X,'FREE GAS IN PLACE (BILLION SCF)',T60,F10.4/)
  102 CONTINUE
      TOOIP=0.
      TOWIP=0.
      TODGIP=0.
      TOFGIP=0.
      DO 103 K=1,KK
      TOOIP=TOOIP+OOIP(K)
      TOWIP=TOWIP+OWIP(K)
      TODGIP=TODGIP+ODGIP(K)
  103 TOFGIP=TOFGIP+OFGIP(K)
  115 FORMAT(/,1X,'TOTAL INITIAL FLUID VOLUMES IN RESERVOIR:',
     & /,10X,'OIL IN PLACE (MILLION STB)',T60,F10.4,
     & /,10X,'WATER IN PLACE (MILLION STB)',T60,F10.4,
     & /,10X,'SOLUTION GAS IN PLACE (BILLION SCF)',T60,F10.4,
     & /,10X,'FREE GAS IN PLACE (BILLION SCF)',T60,F10.4/)
      STBO=SCFO*D5615
      STBW=SCFW*D5615
      MCFG=SCFG*0.001
      MCFG1=SCFG1*0.001
      STBOI=STBO
      STBWI=STBW
      MCFGI=MCFG+MCFG1
      IF(MCFGI.LE.1.D-7.AND.MCFGT.LE.1.D-7)MBEG=0.0
  105 IF(N.EQ.1.AND.ITFLAG.LE.0) CALL PRTPS(N,DELT0)
      IF(N.EQ.NMAX) GO TO 1010
C
C**** ESTABLISH RATES & CALCULATE BHFP(IF PID IS NONZERO)
C
      IF(NVQN.NE.0)CALL QRATE(1,NVQN,NVQNH)
      IF(NVQNH.NE.0)CALL QRATE(2,NVQN,NVQNH)
      IF(NVQNS.NE.0)CALL QRATE(2,NVQN+NVQNH,NVQNS)
C     CALL QRATE(NVQN)
C
C**** CALCULATE SEVEN DIAGONAL MATRIX FOR PRESSURE SOLUTION
C
 1160 CALL SOLMAT(DIV1,D288,D144,N)
C
C**** MODIFY MATRIX ELEMENTS FOR WELLS UNDER IMPLICIT CONTROL.
C
```

```
      IF(NVQN.NE.0) CALL PRATEI(1,NVQN,NVQNH)
      IF(NVQNH.NE.0)CALL PRATEI(2,NVQN,NVQNH)
      IF(NVQNS.NE.0)CALL PRATEI(2,NVQN+NVQNH,NVQNS)
C     CALL PRATEI(NVQN)
C
C**** CALCULATE NEW PRESSURE DISTRIBUTION
C
1170  CALL LSOR(DELT,DELT0,N)
C
C**** CALCULATE IMPLICIT RATES.
C
      IF(NVQN.NE.0) CALL PRATEO(1,NVQN,NVQNH)
      IF(NVQNH.NE.0)CALL PRATEO(2,NVQN,NVQNH)
      IF(NVQNS.NE.0)CALL PRATEO(2,NVQN+NVQNH,NVQNS)
C     CALL PRATEO(NVQN)
 2051 CONTINUE
C
C**** CALCULATE NEW FLUID SATURATIONS
C
      SCFO=0.0
      SCFW=0.0
      SCFG=0.0
      SCFG1=0.0
      RESVOL=0.0
      DO 400 K=1,KK
      DO 400 J=1,JJ
      DO 400 I=1,II
      PPN=PN(I,J,K)
      PP=P(I,J,K)
      BPT=PBOT(I,J,K)
      CALL INTPVT(BPT,RSLOPE,POT,RSOT,MPOT,PP,RSO)
      CALL INTERP(PWT,RSWT,MPWT,PP,RSW)
      CALL INTERP(PGT,CRT,MPGT,PPN,CR)
      BPT=PBOT(I,J,K)
      CALL INTPVT(BPT,BSLOPE,POT,BOT,MPOT,PP,BBO)
      CALL INTERP(PWT,BWT,MPWT,PP,BBW)
      CALL INTERP(PGT,BGT,MPGT,PP,BBG)
      VPP=VP(I,J,K) * (1.0+CR*(P(I,J,K)-PPN))
      RESVOL=RESVOL+VPP
C
      DP1=0.0
      DP2=0.0
      DP3=0.0
      DP4=0.0
      DP5=0.0
      DP6=0.0
      IF((I-1).GT.0)  DP1=P(I-1,J,K)-PP
      IF((I+1).LE.II) DP2=P(I+1,J,K)-PP

c  Yaghi. Testing.... Added next line to fix a problem with I+1
```

```
      IF((I+1).LE.II) DP2=P(I,J,K)-PP

      IF((J-1).GT.0)  DP3=P(I,J-1,K)-PP
      IF((J+1).LE.JJ) DP4=P(I,J+1,K)-PP

c  Yaghi. Testing.... Added next line to fix a problem with I+1
      IF((J+1).LE.JJ) DP2=P(I,J,K)-PP

      IF((K-1).GT.0)  DP5=P(I,J,K-1)-PP
      IF((K+1).LE.KK) DP6=P(I,J,K+1)-PP
C
c#########################################################################
c  This was added to capture the phase volumes that are associated with each
c  face of the well-block.  These are to be used later in an FEM well-model.
c#########################################################################
c           if (i.eq.iqn1(ifem).and.j.eq.iqn2(ifem)) then
            if (i.eq.10.and.j.eq.10) then
            qwww(k) = dp1 * ww (i,j,k)
            qwe(k) = dp2 * we (i,j,k)
            qws(k) = dp3 * ws (i,j,k)
            qwn(k) = dp4 * wn (i,j,k)
            qwt(k) = dp5 * wt (i,j,k)
            qwb(k) = dp6 * wb (i,j,k)
            qow(k) = dp1 * ow (i,j,k)
            qoe(k) = dp2 * oe (i,j,k)
            qos(k) = dp3 * os (i,j,k)
            qon(k) = dp4 * on (i,j,k)
            qot(k) = dp5 * ot (i,j,k)
            qob(k) = dp6 * ob (i,j,k)
            qgw(k) = dp1 * gw (i,j,k)
            qge(k) = dp2 * ge (i,j,k)
            qgs(k) = dp3 * gs (i,j,k)
            qgn(k) = dp4 * gn (i,j,k)
            qgt(k) = dp5 * gt (i,j,k)
            qgb(k) = dp6 * gb (i,j,k)
c     write(60,*)
c     write(60,*)' ww(10,10,k)=',ww(10,10,k)
c     write(60,*)' we(10,10,k)=',we(10,10,k)
c     write(60,*)' ws(10,10,k)=',ws(10,10,k)
c     write(60,*)' wn(10,10,k)=',wn(10,10,k)
c     write(60,*)' wt(10,10,k)=',wt(10,10,k)
c     write(60,*)' wb(10,10,k)=',wb(10,10,k)
c     write(60,*)
c     write(60,*)' ow(10,10,k)=',ow(10,10,k)
c     write(60,*)' oe(10,10,k)=',oe(10,10,k)
c     write(60,*)' os(10,10,k)=',os(10,10,k)
c     write(60,*)' on(10,10,k)=',on(10,10,k)
c     write(60,*)' ot(10,10,k)=',ot(10,10,k)
c     write(60,*)' ob(10,10,k)=',ob(10,10,k)
c     write(60,*)
```

```
c     write(60,*)' gw(10,10,k)=',gw(10,10,k)
c     write(60,*)' ge(10,10,k)=',ge(10,10,k)
c     write(60,*)' gs(10,10,k)=',gs(10,10,k)
c     write(60,*)' gn(10,10,k)=',gn(10,10,k)
c     write(60,*)' gt(10,10,k)=',gt(10,10,k)
c     write(60,*)' gb(10,10,k)=',gb(10,10,k)

c ===== Yaghi. testing
c           write(60,*)
c           write(60,*) ' from Main Prog. while K=',k,' .'
c           write(60,*)'dp1=',dp1,' dp2=',dp2,' dp3=',dp3
c           write(60,*)'dp4=',dp4,' dp5=',dp5,' dp6=',dp6
c           write(60,*)' we(10,10,k)= ',we(i,j,k)
c           write(60,*)' qwww(',k,')=',qwww(k)
c           write(60,*)' qwe(',k,')=', qwe(k)
c           write(60,*)' qws(',k,')=', qws(k)
c           write(60,*)' qwn(',k,')=', qwn(k)
c           write(60,*)' qwt(',k,')=', qwt(k)
c           write(60,*)' qwb(',k,')=', qwb(k)
c           write(60,*)' qow(',k,')=', qow(k)
c           write(60,*)' qoe(',k,')=', qoe(k)
c           write(60,*)' qos(',k,')=', qos(k)
c           write(60,*)' qon(',k,')=', qon(k)
c           write(60,*)' qot(',k,')=', qot(k)
c           write(60,*)' qob(',k,')=', qob(k)
c           write(60,*)' qgw(',k,')=', qgw(k)
c           write(60,*)' qge(',k,')=', qge(k)
c           write(60,*)' qgs(',k,')=', qgs(k)
c           write(60,*)' qgn(',k,')=', qgn(k)
c           write(60,*)' qgt(',k,')=', qgt(k)
c           write(60,*)' qgb(',k,')=', qgb(k)
c           write(60,*)
         endif
c#########################################################################
    DAODP=OW(I,J,K)*DP1+OE(I,J,K)*DP2+OS(I,J,K)*DP3
   &    +ON(I,J,K)*DP4+OT(I,J,K)*DP5+OB(I,J,K)*DP6
    DAWDP=WW(I,J,K)*DP1+WE(I,J,K)*DP2+WS(I,J,K)*DP3
   &    +WN(I,J,K)*DP4+WT(I,J,K)*DP5+WB(I,J,K)*DP6
    SW(I,J,K)=((DAWDP+GWWT(I,J,K)-QW(I,J,K))*DELT +VP(I,J,K)*
   & SWN(I,J,K)/BW(I,J,K)) * BBW/VPP
    SO(I,J,K)=((DAODP+GOWT(I,J,K)-QO(I,J,K))*DELT +VP(I,J,K)*
   & SON(I,J,K)/BO(I,J,K)) * BBO/VPP
    SG(I,J,K)=1.0-SO(I,J,K)-SW(I,J,K)
    IF(SG(I,J,K).GT.0.0) GO TO 404
     SG(I,J,K)=0.0
     SO(I,J,K)=1.0-SW(I,J,K)
 404 IF(SO(I,J,K).LT.ROS) SO(I,J,K)=ROS
    IF(SW(I,J,K).GT.(1.-ROS)) SW(I,J,K)=1.-ROS
 405 CONTINUE
C
```

```
      IF(KCOFF.EQ.0) GO TO 397
      RHO1=VPP*SO(I,J,K)/BBO
      RHO2=VP(I,J,K)*SON(I,J,K)/BO(I,J,K)
      DIFFO=RHO1-RHO2
      RHW1=VPP*SW(I,J,K)/BBW
      RHW2=VP(I,J,K)*SWN(I,J,K)/BW(I,J,K)
      DIFFW=RHW1-RHW2
      RHG1=VPP*SG(I,J,K)/BBG
      RHG2=VP(I,J,K)*SGN(I,J,K)/BG(I,J,K)
      DIFFG=RHG1-RHG2
C
 397  VP(I,J,K)=VPP
      BO(I,J,K)=BBO
      BW(I,J,K)=BBW
      BG(I,J,K)=BBG
      FF1=SO(I,J,K)/BO(I,J,K)
      FF2=SW(I,J,K)/BW(I,J,K)
      SCFO=SCFO+VP(I,J,K)*FF1
      SCFW=SCFW+VP(I,J,K)*FF2
      SCFG=SCFG+VP(I,J,K)*SG(I,J,K)/BG(I,J,K)
      SCFG1=SCFG1+VP(I,J,K)*(RSO*FF1+RSW*FF2)
      CALL INTERP(FOT,BOPT,MPOT,PP,BODER)
      CALL INTERP(POT,RSOPT,MPOT,PP,RSODER)
      CALL INTERP(PWT,BWPT,MPWT,PP,BWDER)
      CALL INTERP(PWT,RSWPT,MPWT,PP,RSWDER)
      CALL INTERP(PGT,BGPT,MPGT,PP,BGDER)
      IF(PP.GT.PBOT(I,J,K))BODER=BSLOPE
      IF(PP.GT.PBOT(I,J,K))RSODER=RSLOPE
      CO=-(BODER-BG(I,J,K)*RSODER)/BO(I,J,K)
      CW=-(BWDER-BG(I,J,K)*RSWDER)/BW(I,J,K)
      CG=-BGDER/BG(I,J,K)
      CALL INTERP(PGT,CRT,MPGT,PP,CR)
      CT(I,J,K)=CR + CO*SO(I,J,K) +CW*SW(I,J,K) + CG*SG(I,J,K)
C
 400 CONTINUE
C     AUTO. TIME STEP CONTROL CALC. OF PRESSURE AND SAT. MAXIMA.
      PPM=0.
      SOM=0.
      SWM=0.
      SGM=0.
      DO 240 K=1,KK
      DO 240 J=1,JJ
      DO 240 I=1,II
      DPO=P(I,J,K)-PN(I,J,K)
      DSO=SO(I,J,K)-SON(I,J,K)
      DSW=SW(I,J,K)-SWN(I,J,K)
      DSG=SG(I,J,K)-SGN(I,J,K)
      IF(ABS(DPO).GT.ABS(PPM)) PPM=DPO
      IF(ABS(DSO).GT.ABS(SOM)) SOM=DSO
      IF(ABS(DSW).GT.ABS(SWM)) SWM=DSW
```

```
      IF(ABS(DSG).GT.ABS(SGM)) SGM=DSG
  240 CONTINUE
      DPMC=ABS(PPM)
      DSMC=ABS(SOM)
      IF(DSMC.LT.ABS(SGM))DSMC=ABS(SGM)
      IF(DSMC.LT.ABS(SWM))DSMC=ABS(SWM)
C****REPEAT TIME STEP?
      IF(DSMC.LT.DSMAX.AND.DPMC.LT.DPMAX) GO TO 402
      IF(DELT.LE.DTMIN.OR.FACT2.GE.1.0) GO TO 402
      ITFLAG=ITFLAG+1
      DELT=DELT*FACT2
      IF(DELT.LT.DTMIN) DELT=DTMIN
      FT=ETI+DELT
      IF(FT.GT.FTMAX) DELT=FTMAX-ETI
C****RESET VARIABLES.
      DO 250 I=1,II
      DO 250 J=1,JJ
      DO 250 K=1,KK
      P(I,J,K)=PN(I,J,K)
      SO(I,J,K)=SON(I,J,K)
      SW(I,J,K)=SWN(I,J,K)
      SG(I,J,K)=SGN(I,J,K)
  250 CONTINUE
C
   33 FORMAT(//)
C
      GO TO 1060
  402 CONTINUE
C
C****UNDERSATURATED GRID BLOCK SATURATION CALCULATION.
C
      DO 410 I=1,II
      DO 410 J=1,JJ
      DO 410 K=1,KK
      IF(P(I,J,K).GT.PN(I,J,K)) GO TO 410
      IF(P(I,J,K).LT.PBOT(I,J,K)) GO TO 410
      IP=I+1
      IM=I-1
      JP=J+1
      JM=J-1
      KP=K+1
      KM=K-1
      IF(IP GT.II) GO TO 412
      IF(SGN(IP,J,K).GT.0.0001) GO TO 410
  412 IF(IM.LT.1) GO TO 414
      IF(SGN(IM,J,K).GT.0.0001) GO TO 410
  414 IF(JP GT.JJ) GO TO 416
      IF(SGN(I,JP,K).GT.0.0001) GO TO 410
  416 IF(JM.LT.1) GO TO 418
      IF(SGN(I,JM,K).GT.0.0001) GO TO 410
```

```
 418 IF(KP.GT.KK) GO TO 420
     IF(SGN(I,J,KP).GT.0.0001) GO TO 410
 420 IF(KM.LT.1) GO TO 422
     IF(SGN(I,J,KM).GT.0.0001) GO TO 410
 422 SG(I,J,K)=0.0
     SO(I,J,K)=1.0-SW(I,J,K)
 410 CONTINUE
C**** REPRESSURIZATION ALGORITHM.
     IF(IREPRS.EQ.1) GO TO 51
     DO 50 I=1,II
     DO 50 J=1,JJ
     DO 50 K=1,KK
     IF(SG(I,J,K).LE.0.0001) GO TO 50
     PP=P(I,J,K)
     IF(P(I,J,K).GT.PBOT(I,J,K)) PP=PBOT(I,J,K)
     CALL INTERP(POT,BOT,MPOT,PP,BBO)
     CALL INTERP(POT,RSOT,MPOT,PP,RSO)
     CALL INTERP(PGT,BGT,MPGT,PP,BBG)
     IF(SO(I,J,K).EQ.0.0) GO TO 50
     RSONEW=RSO + SG(I,J,K)*BBO/(SO(I,J,K)*BBG)
     CALL INTERP(RSOT,POT,MPOT,RSONEW,PBONEW)
     PBOT(I,J,K)=PBONEW
  50 CONTINUE
  51 CONTINUE
C**** UPDATE OLD FLUID VOLUMES FOR MATERIAL BALANCE.
     STBOI=STBO
     STBWI=STBW
     MCFGI=MCFGT
C**** UPDATE NEW FLUID VOLUMES.
     STBO=SCFO*D5615
     STBW=SCFW*D5615
     MCFG=SCFG*0.001
     MCFG1=SCFG1*0.001
     MCFGT=MCFG+MCFG1
C
C***** DEBUG PRINT OF PRESENT AND FUTURE P,SO,SW,SG VALUES.
     IF(KCOFF.EQ.0)GO TO 291
     DO 290 K=1,KK
     DO 290 J=1,JJ
     DO 290 I=1,II
290  CONTINUE
291  CONTINUE
C
C**** WELL REPORT (ALL RATE & PRESSURE DATA APPLICABLE THIS STEP)
C
     IJ=0
     TOR=0.
     TGR=0.
     TWR=0.
     TOC=0.
```

```
      TGC=0.
      TWC=0.
      N2=NVQN+NVQNH+NVQNS
      PWFAVG1=0.

      DO 2059 J=1,N2
      GOR=0.
      WOR=0.
      IJ=IJ+1
      IF (J.LE.NVQN)THEN
       IQ1=IQN1(J)
       IQ2=IQN2(J)
       IQ3=IQN3(J)
       LAY=IQ3+(LAYER(J)-1)
      ELSE
       IQ3=1
       LAY=LAYER(J)
      ENDIF
      DO 2050 KKK=IQ3,LAY
      K=KKK
      IF (J GT.NVQN)THEN
       IQ1=IQH1(J,KKK)
       IQ2=IQH2(J,KKK)
       K=IQH3(J,KKK)
      ENDIF
      QOO=QO(IQ1,IQ2,K)*D5615
      QWW=QW(IQ1,IQ2,K)*D5615
      QGG=QG(IQ1,IQ2,K)*.001
      CUMO(J,KKK)=CUMO(J,KKK)+QOO*DELT*.001
      CUMW(J,KKK)=CUMW(J,KKK)+QWW*DELT*.001
      CUMG(J,KKK)=CUMG(J,KKK)+QGG*DELT*.001
      IF(J.EQ.1.AND.KIP(1).GT.0)PWFAVG1=PWFAVG1+PWFC(1,KKK)
      IF(J.EQ.1.AND.KIP(1).LT.0)PWFAVG1=PWFAVG1+PWF(1,KKK)
      IF(IWLREP.EQ.0) GO TO 2050
       IF(FT.GT.0.999.AND.FT.LT.1.001) GO TO 891
       IF(ABS(FT-NUMPRD*SONTVL).GT.0.001) GO TO 2050
  891 continue
 5911 FORMAT(/,T56,'------ RATE ------',22X,'--- CUMULATIVE ---',
     & /,13X,'WELL LOCATION',4X,'CALC   SPEC   SPEC',4X,
     & 'OIL   GAS   WATER   GOR   WOR',5X,
     & 'OIL   GAS   WATER',/,
     & 14X,'ID',3X,'I  J  K   BHFP   BHFP   PI',
     & 3X,'STB/D   MCF/D  STB/D',20X,'MSTB   MMCF   MSTB',/)
      IF(QOO.EQ.0.)GO TO 998
      GOR=QGG*1000./QOO
      WOR=QWW/QOO
  998 continue
  592 FORMAT(11X,A5,1X,3I3,F8.2,F8.0,F7.3,3F9.0,F7.0,F7.3,3F8.0)
      TOR=TOR+QOO
      TGR=TGR+QGG
```

```
      TWR=TWR+QWW
      TOC=TOC+CUMO(J,KKK)
      TGC=TGC+CUMG(J,KKK)
      TWC=TWC+CUMW(J,KKK)
 2050 CONTINUE
      IF(J.EQ.1)PWFAVG1=PWFAVG1/LAY
 2059 CONTINUE
      IF(IWLREP.EQ.0)GO TO 2052
       IF(FT.GT.0.999.AND.FT.LT.1.001) GO TO 791
       IF(ABS(FT-NUMPRD*SONTVL).GT.0.001) GO TO 2052
  791  continue
 5912 FORMAT(12X,102('-'),/,
     &12X,'TOTALS',31X,3F9.0,14X,3F8.0,/)
 2052 CONTINUE
C
C CALCULATE MATERIAL BALANCE ERRORS & AVERAGE RESERVOIR PRESSURE
C
      if (i_signal.lt.0) then
        CALL FEMWELL ( ifem,i_signal)
      endif
c#########################################################################
c  At this entry the fem can evaluate the well-block.  There should be a
c  switch to allow the bypass of this calculation if desired
c#########################################################################
c ==== Yaghi
c         write(30,*) ' This simulation is set in the data file'
c         write(30,*) '  to run for ttltime=5000, to change it,'
c         write(30,*) '  simply edit tyler2 dat and change the '
c         write(30,*) '  TMAX variable.                '
c         write(30,*) '                                '

          CALL FEMETHOD(ifem,delt,ttltime)

c Yaghi.  delete the next line (stop).
          if (ttltime .gt. 2500) then
            t=MCLOCK()-t0
            write(66,*)' The Yaghi-FEM Code took ',
     &         t, ' on the J30'
            write(66,*)' delt = ', delt,' ttltime = ',ttltime
            write(66,*)' Looped 3 times in FEMETHOD'
            STOP
          end if

           ttltime=ttltime+delt
c#########################################################################
      DELT0=DELT
      ETI=ETI+DELT
      CALL MATBAL(DELT0,D5615)
C
      IF(WOR.GT.WORMAX) GO TO 1002
```

```
      IF(GOR.GT.GORMAX) GO TO 1003
      IF(PAVG.LT.PAMIN) GO TO 1004
      IF(PAVG.GT.PAMAX) GO TO 1005
C
C***** SUMMARY REPORT.
      IF(MOD(N,NTS).EQ.0)THEN
       PIDRES1=(OPR+WPR)/(PAVG-PWFAVG1)
       ENDIF
 987  FORMAT(I6,2F8.2,F10.2,F9.0,F9.1,F10.0,F7.2,F7.1,F9.0,3F8.2,
     & F8.3)
      IF(NGRSW.EQ.1.AND.(ETI-NUMPRD*SONTVL).GE.0.0)CALL GRPHCS
      IF(ISUMRY.EQ.0) GO TO 2057
      NLP=N+1
 691  CALL PRTPS(NLP,DELT0)
 2057 IF(N.NE.KCO.OR.KCO1.EQ.0)GO TO 500
c    only write the value at well block
           GOTO 583
      DO 300 K=1,KK
      DO 300 J=1,JJ
      DO 300 I=1,II
300   CONTINUE
583    WRITE(NO,*)'AT WELL BLOCK'
c
c ====== Yaghi.  Check this for multiple layers
c
       i=10
       j=10
       k=3
           WRITE(NO,21)I,J,K,VP(I,J,K),CT(I,J,K),BO(I,J,K),SO(I,J,K),
     &BW(I,J,K),SW(I,J,K),BG(I,J,K),SG(I,J,K)
C
  500 CONTINUE
      IF(N.EQ.KSN)KSN=KSN+KSN1
      IF(N.EQ.KSM)KSM=KSM+KSM1
      IF(N.EQ.KCO)KCO=KCO+KCO1
C**** UPDATE ARRAYS.
      DO 1150 K=1,KK
      DO 1150 J=1,JJ
      DO 1150 I=1,II
      QO(I,J,K)=0.0
      QW(I,J,K)=0.0
      QG(I,J,K)=0.0
      PN(I,J,K)=P(I,J,K)
      SON(I,J,K)=SO(I,J,K)
      SWN(I,J,K)=SW(I,J,K)
      SGN(I,J,K)=SG(I,J,K)
1150  CONTINUE
C
591   FORMAT(///T5,10('*'),' WELL REPORT FOR ALL ACTIVE WELLS ',
     &4X,'ELAPSED TIME =',F11.6,' DAYS FROM BEGINNING OF SIMULATION ',
```

```
     &10('*'),//)
C
      IF(WOR.GT.WORMAX) GO TO 1002
      IF(GOR.GT.GORMAX) GO TO 1003
      IF(PAVG.LT.PAMIN) GO TO 1004
      IF(PAVG.GT.PAMAX) GO TO 1005
1000  CONTINUE
C
1002  WRITE(NO,2002)
      GO TO 1010
1003  WRITE(NO,2003)
      GO TO 1010
1004  WRITE(NO,2004)
      GO TO 1010
1005  WRITE(NO,2005)
1010  CLOSE(UNIT=NI)
      CLOSE(UNIT=NO)
      IF(NRESTART.EQ.1)CALL RESTART
      STOP
C
2002  FORMAT(/T15,'MAXIMUM WOR HAS BEEN EXCEEDED --- SIMULATION',
     &' IS BEING TERMINATED'//)
2003  FORMAT(/T15,'MAXIMUM GOR HAS BEEN EXCEEDED --- SIMULATION',
     &' IS BEING TERMINATED'//)
2004  FORMAT(/T15,'MINIMUM AVERAGE RESERVOIR PRESSURE WAS NOT',
     &' ACHIEVED --- SIMULATION IS BEING TERMINATED'//)
2005  FORMAT(/T15,'MAXIMUM AVERAGE RESERVOIR PRESSURE
     &HAS BEEN EXCEEDED --- SIMULATION IS BEING TERMINATED'//)
C
      END

c ==========================================================
      SUBROUTINE MESHGEN (ifem)
c ==========================================================
c     This is a subroutine that generates the mesh for a well-model that
c     interfaces with BOAST.  This generation is based on using a
c     radial approach out from the well.  The angles are in the first
c     quadrant.  That is, they start from the x-axis
      INCLUDE 'PARAMETR.FOR'
      INCLUDE 'PARAMETR.FEM'
      dimension itemp(4)
      dimension s(n_theta),f(nbands),k1(num_nodes),k2(num_nodes),
     &          k3(num_nodes),k4(num_nodes)
      COMMON /VECTOR/ DX(NX,NY,NZ),DY(NX,NY,NZ),DZ(NX,NY,NZ)
      COMMON /IQN/ IQN1(NW),IQN2(NW),IQN3(NW)
      COMMON/MESHPTS/ i1(num_elemts),i2(num_elemts),
     & i3(num_elemts),i4(num_elemts), r1
      common/gauss/gauss_pts(2),gauss_wts(2),num_gauss_pts
      common/shape/phif(lmt_nodes),
     & dphd(lmt_nodes,3),coords(lmt_nodes,3)
```

```
c#####################################################################

      common/glstiff/stiff(num_nodes*nzp,num_nodes*nzp),
     &  fl(num_nodes*nzp),xl(num_nodes*nzp),ipvt(num_nodes*nzp),
     &  sofl(num_nodes*nzp),sofl1(num_nodes*nzp),sofl2(num_nodes*nzp),
     &  sgfl(num_nodes*nzp),sgfl1(num_nodes*nzp),sgfl2(num_nodes*nzp),
     &  swfl(num_nodes*nzp),swfl1(num_nodes*nzp),swfl2(num_nodes*nzp)

      common/stiff0/stiff0(num_nodes*nzp,num_nodes*nzp),
     &  sostiff(num_nodes*nzp,num_nodes*nzp),
     &  sgstiff(num_nodes*nzp,num_nodes*nzp),
     &  swstiff(num_nodes*nzp,num_nodes*nzp)

      common/oldpwg/pold(num_nodes*nzp),swold(num_nodes*nzp),
     &  soold(num_nodes*nzp),  sgold(num_nodes*nzp)

      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)

      common/oilfem/ A(lmt_nodes,lmt_nodes),
     & COIJ(lmt_nodes,lmt_nodes),DO(lmt_nodes),
     & COIL(lmt_nodes,lmt_nodes),QOIL(lmt_nodes)

      common/waterfem/ H(lmt_nodes,lmt_nodes),
     & BWAT(lmt_nodes,lmt_nodes),DW(lmt_nodes),
     & CWIJ(lmt_nodes,lmt_nodes),QWAT(lmt_nodes),
     & CWAT(lmt_nodes,lmt_nodes)

      common/gasfem/ EGAS(lmt_nodes,lmt_nodes),
     & G(lmt_nodes,lmt_nodes),DG(lmt_nodes),
     & WX(lmt_nodes,lmt_nodes),QGAS(lmt_nodes),
     & rsodo(lmt_nodes),rswdw(lmt_nodes),
     & GX(lmt_nodes,lmt_nodes),rswb(lmt_nodes,lmt_nodes) ,
     & cacw(lmt_nodes,lmt_nodes),rsoa(lmt_nodes,lmt_nodes),
     & rswh(lmt_nodes,lmt_nodes) ,
     & fgas(lmt_nodes,lmt_nodes),fwat(lmt_nodes,lmt_nodes)

c#####################################################################

          mtemp = n_theta
          n_elms= (nbands-1)*(mtemp-1)
          xmax = DX (10,10,3)/2.0
          ymax = DY (10,10,3)/2.0
c     rstep=(xmax-rw)/(nbands-1)
          pi=3.1415926

          do j=1,mtemp
```

```
            angle=0.5*pi*(j-1)/(mtemp-1)
          rstep = 10.0
            do i=1,nbands-2
c           k is the node number along circle number "i"
              k=mtemp*(i-1)+j
           xstep = rstep

              if(2.0*angle .lt. 0.5*pi) then
                rl=xmax/cos(angle)
              else
             rl=ymax/sin(angle)
              endif
              if(i .ne. nbands) then
               xyzcds(k,1)=(rw+rstep*(i-1))*cos(angle);
               xyzcds(k,2)=(rw+rstep*(i-1))*sin(angle);
              else
            xyzcds(k,1)=rl*cos(angle);
           xyzcds(k,2)=rl*sin(angle);
              end if
           rstep = rstep + 5
             end do
           end do

        rstep = 32.0
        do j = 1, mtemp
          angle = 0.5*pi*(j-1)/(mtemp-1)
          k=mtemp*(nbands-2)+j
              if(2.0*angle .lt. 0.5*pi) then
            rl=xmax/cos(angle)
              else
            rl=ymax/sin(angle)
              endif
              if(i .ne. nbands) then
                xyzcds(k,1)=(rw+rstep*(i-1))*cos(angle);
                xyzcds(k,2)=(rw+rstep*(i-1))*sin(angle);
              else
             xyzcds(k,1)=rl*cos(angle);
            xyzcds(k,2)=rl*sin(angle);
              end if
        end do

        rstep = (xmax-rw) - rstep
        do j = 1, mtemp
          angle = 0.5*pi*(j-1)/(mtemp-1)
          k=mtemp*(nbands-1)+j
              if(2.0*angle .lt. 0.5*pi) then
            rl=xmax/cos(angle)
              else
            rl=ymax/sin(angle)
              endif
```

```
            if(i .ne. nbands) then
              xyzcds(k,1)=(rw+rstep*(i-1))*cos(angle);
              xyzcds(k,2)=(rw+rstep*(i-1))*sin(angle);
            else
           xyzcds(k,1)=rl*cos(angle);
          xyzcds(k,2)=rl*sin(angle);
            end if
      end do

c ==== Yaghi. testing
c      write(60,*)
c      write(60,*) ' xyzCoords in MESHGEN: '
     do 50 j = 1,num_nodes
       xyzcds(j,3) = Dz (10,10,1)
c       write(60,49)j,xyzcds(j,1),xyzcds(j,2),xyzcds(j,3)
49      format(5x,I4,5x,f12.4,5x,f12.4,5x,f12.4)
50    continue

c ==========================================================
c   The following determines the relationship between local nodes and
c   global nodes .

          do 60 i=1,nbands
          do 70 j=1,mtemp
          if (i.eq.1.and.j.eq.1) then
          k1(j)=1
          k2(j)=0
          k3(j)=0
          k4(j)=0
          endif
          if (i.eq.1.and.j.ne.1.and.j.ne.mtemp) then
          k1(j)=k1(j-1)+1
          k2(j)=k2(j-1)+1
          k3(j)=0
          k4(j)=0
          endif
          if (i.eq.1.and.j.eq.mtemp) then
          k1(j)=mtemp-1
          k2(j)=0
          k3(j)=0
          k4(j)=0
          endif
          if (i.eq.2.and.k2(j).ne.0) then
          k1((i-1)*mtemp+j)= k1(j)
          k2((i-1)*mtemp+j)= k2(j)
          k3((i-1)*mtemp+j)= k1(j)+mtemp-1
          k4((i-1)*mtemp+j)= k2(j)+mtemp-1
          endif
          if (i.gt.2.and.k3((i-2)*mtemp+j).ne.0) then
          k1((i-1)*mtemp+j)= k1((i-2)*mtemp+j)+mtemp-1
```

```
      k2((i-1)*mtemp+j)= k2((i-2)*mtemp+j)+mtemp-1
      k3((i-1)*mtemp+j)= k3((i-2)*mtemp+j)+mtemp-1
      k4((i-1)*mtemp+j)= k4((i-2)*mtemp+j)+mtemp-1
      endif
      if (i.ne.1.and.k2((i-2)*mtemp+j).eq.0) then
      k1((i-1)*mtemp+j)= k1((i-2)*mtemp+j)
      k2((i-1)*mtemp+j)= k1((i-2)*mtemp+j)+mtemp-1
      k3((i-1)*mtemp+j)=0
      k4((i-1)*mtemp+j)=0
      endif
      if (i.gt.2.and.k3((i-2)*mtemp+j).eq.0) then
      k1((i-1)*mtemp+j)=k1((i-2)*mtemp+j)+mtemp-1
      k2((i-1)*mtemp+j)=k2((i-2)*mtemp+j)+mtemp-1
      k3((i-1)*mtemp+j)=0
      k4((i-1)*mtemp+j)=0
      endif
      if (i.eq.nbands.and.k3((i-2)*mtemp+j).eq.0) then
      k1((i-1)*mtemp+j)=k2((i-2)*mtemp+j)
      k2((i-1)*mtemp+j)=0
      k3((i-1)*mtemp+j)=0
      k4((i-1)*mtemp+j)=0
      endif
      if (i.eq.nbands.and.k3((i-2)*mtemp+j).ne.0) then
      k1((i-1)*mtemp+j)= k3((i-2)*mtemp+j)
      k2((i-1)*mtemp+j)= k4((i-2)*mtemp+j)
      k3((i-1)*mtemp+j)=0
      k4((i-1)*mtemp+j)=0
      endif
70    continue
60    continue

      do 90 i=1,n_elms
      l=1
      do 80 j=1, num_nodes
      if(k1(j).ne.0.and.k1(j).eq.i) then
      itemp(l)=j
      l=l+1
      endif
      if(k2(j).ne.0.and.k2(j).eq.i) then
      itemp(l)=j
      l=l+1
      endif
      if(k3(j).ne.0.and.k3(j).eq.i) then
      itemp(l)=j
      l=l+1
      endif
      if(k4(j).ne.0.and.k4(j).eq.i) then
      itemp(l)=j
      l=l+1
      endif
```

```
 80      continue
         i1(i)=itemp(1)
         i2(i)=itemp(3)
         i3(i)=itemp(4)
         i4(i)=itemp(2)

 90      continue

      call gauss_3d8

         return
         end

c ================================================================
      SUBROUTINE elmtGEN (ifem,kount,k)
c ================================================================
c   This is a subroutine that generates the nodal points for one element
c   from the data that was generated for the whole mesh.  It may be
c   possible to do the global assembly without the need for doing
c   each element, later.  However, this is the standard approach.
      INCLUDE 'PARAMETR.FOR'
      INCLUDE 'PARAMETR.FEM'
      COMMON/MESHPTS/ i1(num_elemts),i2(num_elemts),
     & i3(num_elemts),i4(num_elemts), r1
      common/gauss/gauss_pts(2),gauss_wts(2),num_gauss_pts
      common/shape/phif(lmt_nodes),
     & dphd(lmt_nodes,3),coords(lmt_nodes,3)
      COMMON /VECTOR/ DX(NX,NY,NZ),DY(NX,NY,NZ),DZ(NX,NY,NZ)
c#################################################################

      common/glstiff/stiff(num_nodes*nzp,num_nodes*nzp),
     &  fl(num_nodes*nzp),xl(num_nodes*nzp),ipvt(num_nodes*nzp),
     &  sofl(num_nodes*nzp),sofl1(num_nodes*nzp),sofl2(num_nodes*nzp),
     &  sgfl(num_nodes*nzp),sgfl1(num_nodes*nzp),sgfl2(num_nodes*nzp),
     &  swfl(num_nodes*nzp),swfl1(num_nodes*nzp),swfl2(num_nodes*nzp)

      common/stiff0/stiff0(num_nodes*nzp,num_nodes*nzp),
     &  sostiff(num_nodes*nzp,num_nodes*nzp),
     &  sgstiff(num_nodes*nzp,num_nodes*nzp),
     &  swstiff(num_nodes*nzp,num_nodes*nzp)

      common/oldpwg/pold(num_nodes*nzp),swold(num_nodes*nzp),
     &  soold(num_nodes*nzp),  sgold(num_nodes*nzp)

      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)
```

```
      common/oilfem/ A(lmt_nodes,lmt_nodes),
     & COIJ(lmt_nodes,lmt_nodes),DO(lmt_nodes),
     & COIL(lmt_nodes,lmt_nodes),QOIL(lmt_nodes)

      common/waterfem/ H(lmt_nodes,lmt_nodes),
     & BWAT(lmt_nodes,lmt_nodes),DW(lmt_nodes),
     & CWIJ(lmt_nodes,lmt_nodes),QWAT(lmt_nodes),
     & CWAT(lmt_nodes,lmt_nodes)

      common/gasfem/ EGAS(lmt_nodes,lmt_nodes),
     & G(lmt_nodes,lmt_nodes),DG(lmt_nodes),
     & WX(lmt_nodes,lmt_nodes),QGAS(lmt_nodes),
     & rsodo(lmt_nodes),rswdw(lmt_nodes),
     & GX(lmt_nodes,lmt_nodes),rswb(lmt_nodes,lmt_nodes) ,
     & cacw(lmt_nodes,lmt_nodes),rsoa(lmt_nodes,lmt_nodes),
     & rswh(lmt_nodes,lmt_nodes) ,
     & fgas(lmt_nodes,lmt_nodes),fwat(lmt_nodes,lmt_nodes)

c#####################################################################

      coords(5,1) = xyzcds(i1(kount),1)
      coords(5,2) = xyzcds(i1(kount),2)
      coords(5,3) = 0.0d0
      coords(1,1) = xyzcds(i1(kount),1)
      coords(1,2) = xyzcds(i1(kount),2)
      coords(1,3) = xyzcds(i1(kount),3)

      coords(6,1) = xyzcds(i2(kount),1)
      coords(6,2) = xyzcds(i2(kount),2)
      coords(6,3) = 0.0d0
      coords(2,1) = xyzcds(i2(kount),1)
      coords(2,2) = xyzcds(i2(kount),2)
      coords(2,3) = xyzcds(i2(kount),3)

      coords(7,1) = xyzcds(i3(kount),1)
      coords(7,2) = xyzcds(i3(kount),2)
      coords(7,3) = 0.0d0
      coords(3,1) = xyzcds(i3(kount),1)
      coords(3,2) = xyzcds(i3(kount),2)
      coords(3,3) = xyzcds(i3(kount),3)

      coords(8,1) = xyzcds(i4(kount),1)
      coords(8,2) = xyzcds(i4(kount),2)
      coords(8,3) = 0.0d0
      coords(4,1) = xyzcds(i4(kount),1)
      coords(4,2) = xyzcds(i4(kount),2)
      coords(4,3) = xyzcds(i4(kount),3)

      if (k .gt. 1) then
       do i = 1, 8
```

```
        coords(i,3) = 0.0d0
      end do

      do i = 5, 8
        do j = 2, k
          coords(i,3) = dz(10,10,j-1)
     &                +coords(i,3)
        end do
      enddo

      do i =1, 4
        do j = 1, k
          coords(i,3) = dz(10,10,j)
     &                +coords(i,3)
        end do
      enddo
     endif
        return
        end
c =============================================================
      SUBROUTINE FEMWELL (ifem,i_signal)
c =============================================================
c  This subroutine initializes the fem mesh points for there initial
c  pressure, oil sat, gas sat, and water sat.
      INCLUDE 'PARAMETR.FOR'
      INCLUDE 'PARAMETR.FEM'
      COMMON /NUMBER/ II,JJ,KK
      COMMON /SPVT/ SAT(NTE),KROT(NTE),KRWT(NTE),KRGT(NTE),PCOWT(NTE),
     & PCGOT(NTE),POT(NTE),MUOT(NTE),BOT(NTE),BOPT(NTE),RSOT(NTE),RSOPT
     & (NTE),PWT(NTE),MUWT(NTE),BWT(NTE),BWPT(NTE),RSWT(NTE),RSWPT(NTE),
     & PGT(NTE),MUGT(NTE),BGT(NTE),BGPT(NTE),CRT(NTE)
      COMMON /RATE/ QO(NX,NY,NZ),QW(NX,NY,NZ),QG(NX,NY,NZ)
      COMMON /BUBBLE/ PBO,VSLOPE,BSLOPE,RSLOPE,PMAXT,IREPRS,
     & RHOSCO,RHOSCG,RHOSCW,MSAT,MPOT,MPWT,MPGT,PBOT(NX,NY,NZ)
      COMMON /PRTP/ P(NX,NY,NZ)
      COMMON /PRTS/ SO(NX,NY,NZ),SW(NX,NY,NZ),SG(NX,NY,NZ)
      COMMON /SRATE/ PID(NW,NL),PWF(NW,NL),PWFC(NW,NL),KIP(NW),
     & GMO(NW,NL),GMW(NW,NL),GMG(NW,NL),LAYER(NW),QVO(NW),
     & QVW(NW),QVG(NW),QVT(NW),CUMO(NW,NL),CUMW(NW,NL),CUMG(NW,NL)
      COMMON /IQN/ IQN1(NW),IQN2(NW),IQN3(NW)
c#################################################################

      common/glstiff/stiff(num_nodes*nzp,num_nodes*nzp),
     & fl(num_nodes*nzp),xl(num_nodes*nzp),ipvt(num_nodes*nzp),
     & sofl(num_nodes*nzp),sofl1(num_nodes*nzp),sofl2(num_nodes*nzp),
     & sgfl(num_nodes*nzp),sgfl1(num_nodes*nzp),sgfl2(num_nodes*nzp),
     & swfl(num_nodes*nzp),swfl1(num_nodes*nzp),swfl2(num_nodes*nzp)

      common/stiff0/stiff0(num_nodes*nzp,num_nodes*nzp),
     & sostiff(num_nodes*nzp,num_nodes*nzp),
```

```
     &  sgstiff(num_nodes*nzp,num_nodes*nzp),
     &  swstiff(num_nodes*nzp,num_nodes*nzp)

      common/oldpwg/pold(num_nodes*nzp),swold(num_nodes*nzp),
     &  soold(num_nodes*nzp),  sgold(num_nodes*nzp)

      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)

c#####################################################################

          i_signal = -1
          K1=10
          K2=10
          K3=3
c       write(60,*)
c        write(60,*) ' From FEMWEL:  p, so, sw, sg: '
         do jk = 1, 3
c          write(60,*)
c          write(60,*)' K = ', jk
c          write(60,*) p(10,10,jk), so(10,10,jk),
c    &         sw(10,10,k), sg(10,10,jk)
         end do

c        write(60,*)
c        write(60,*) ' Initializing pressure & saturations.'
c        write(60.*) ' From FEMWEL:  pfem, sofem, swfem, sgfem: '
          do j = 1,num_nodes*4
          pfem(j) =  p(k1,k2,k3)
          sofem(j) = so(K1,K2,K3)
          swfem(j) = sw(K1,K2,K3)
          sgfem(j) = sg(K1,K2,K3)
          pbotfem(j) = pbot(k1,k2,k3)

c           write(60,*)
c           write(60,*)j,pfem(j),sofem(j),swfem(j),
c    &          sgfem(j)

       end do

c#####################################################################
       pp=pfem(1)
       bpt=pbotfem(1)
       CALL INTPVT(BPT,RSLOPE,POT,RSOT,MPOT,PP,RSO)
       CALL INTERP(PWT,RSWT,MPWT,PP,RSW)
c  This computes BO above or below the bubble point pressure
       CALL INTPVT(BPT,BSLOPE,POT,BOT,MPOT,PP,BOfem)
```

```
c  This computes BW at PP pressure, this is not effected by the bubble
c  point.
      CALL INTERP(PWT,BWT,MPWT,PP,BWfem)
c  This computes BG at PP pressure, this is not effected by the bubble
c  point.
      CALL INTERP(PGT,BGT,MPGT,PP,BGfem)
C
c#####################################################################
      foil=qo(k1,k2,k3)*bofem
      fwat=qw(k1,k2,k3)*bwfem
      fgas=(qg(k1,k2,k3)
     & -qo(k1,k2,k3)*rso
     & -qw(k1,k2,k3)*rsw)*bgfem
      fnum=(qg(k1,k2,k3)
     & -qo(k1,k2,k3)*rso
     & -qw(k1,k2,k3)*rsw)*bgfem+foil+fwat
      sgas=fgas/fnum
      wgas=fwat/fnum
      ogas=foil/fnum
c#####################################################################
            return
          end


c=====================================================================
      SUBROUTINE SHAPE_3D8(xi,eta,zeta)
c=====================================================================
C  This subroutine has the shape functions for a trilinear hexahedral
C  element.  These are taken from "The Finite Element Method" Hughes.
C  and modified for the "z-axis" to be the negative as in BOAST.
      INCLUDE 'PARAMETR.FEM'
      common/shape/phif(lmt_nodes),
     & dphd(lmt_nodes,3),coords(lmt_nodes,3)
      xim = 1.0-xi
      xip = 1.0+xi
      etap = 1.0+eta
      etam = 1.0-eta
      zip = 1.0+zeta
      zim = 1.0-zeta
c
c************* Shape Functions  ***************
c
      phif(1) = 0.125*xim*etam*zip
      phif(2) = 0.125*xip*etam*zip
      phif(3) = 0.125*xip*etap*zip
      phif(4) = 0.125*xim*etap*zip
      phif(5) = 0.125*xim*etam*zim
      phif(6) = 0.125*xip*etam*zim
      phif(7) = 0.125*xip*etap*zim
      phif(8) = 0.125*xim*etap*zim
```

```
c
c************** Derivatives of Shape Functions  ***************
c
c------ Derivatives wrt xi ------------
      dphd(1,1) = -0.125*etam*zip
      dphd(2,1) = 0.125*etam*zip
      dphd(3,1) = 0.125*etap*zip
      dphd(4,1) = -0.125*etap*zip
      dphd(5,1) = -0.125*etam*zim
      dphd(6,1) = 0.125*etam*zim
      dphd(7,1) = 0.125*etap*zim
      dphd(8,1) = -0.125*etap*zim
c------ Derivatives wrt eta------------
      dphd(1,2) = -0.125*xim*zip
      dphd(2,2) = -0.125*xip*zip
      dphd(3,2) = 0.125*xip*zip
      dphd(4,2) = 0.125*xim*zip
      dphd(5,2) = -0.125*xim*zim
      dphd(6,2) = -0.125*xip*zim
      dphd(7,2) = 0.125*xip*zim
      dphd(8,2) = 0.125*xim*zim
c------ Derivatives wrt zeta --------------
      dphd(1,3) = 0.125*xim*etam
      dphd(2,3) = 0 125*xip*etam
      dphd(3,3) = 0.125*xip*etap
      dphd(4,3) = 0.125*xim*etap
      dphd(5,3) = -0.125*xim*etam
      dphd(6,3) = -0.125*xip*etam
      dphd(7,3) = -0.125*xip*etap
      dphd(8,3) = -0.125*xim*etap
     return
     end
c=======================================================================
     SUBROUTINE JACOBI_3D8(i_elm,det_jacobian,dn)
c=======================================================================
C  This subroutine uses the shape functions for a trilinear hexahedral
C  element.  These are taken from "The Finite Element Method" Hughes.
C  and modified for the "z-axis" to be the negative as in BOAST.
c  It calculates the 3-D Jacobian..
     INCLUDE 'PARAMETR.FEM'
     INCLUDE 'PARAMETR.FOR'
      common/shape/phif(lmt_nodes),
    & dphd(lmt_nodes,3),coords(lmt_nodes,3)
c#######################################################################

     common/glstiff/stiff(num_nodes*nzp,num_nodes*nzp),
    &  fl(num_nodes*nzp),xl(num_nodes*nzp),ipvt(num_nodes*nzp),
    &  sofl(num_nodes*nzp),sofl1(num_nodes*nzp),sofl2(num_nodes*nzp),
    &  sgfl(num_nodes*nzp),sgfl1(num_nodes*nzp),sgfl2(num_nodes*nzp),
    &  swfl(num_nodes*nzp),swfl1(num_nodes*nzp),swfl2(num_nodes*nzp)
```

```
      common/stiff0/stiff0(num_nodes*nzp,num_nodes*nzp),
     &  sostiff(num_nodes*nzp,num_nodes*nzp),
     &  sgstiff(num_nodes*nzp,num_nodes*nzp),
     &  swstiff(num_nodes*nzp,num_nodes*nzp)

      common/oldpwg/pold(num_nodes*nzp),swold(num_nodes*nzp),
     &  soold(num_nodes*nzp),  sgold(num_nodes*nzp)

      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)

      common/oilfem/ A(lmt_nodes,lmt_nodes),
     & COIJ(lmt_nodes,lmt_nodes),DO(lmt_nodes),
     & COIL(lmt_nodes,lmt_nodes),QOIL(lmt_nodes)

      common/waterfem/ H(lmt_nodes,lmt_nodes),
     & BWAT(lmt_nodes,lmt_nodes),DW(lmt_nodes),
     & CWIJ(lmt_nodes,lmt_nodes),QWAT(lmt_nodes),
     & CWAT(lmt_nodes,lmt_nodes)

      common/gasfem/ EGAS(lmt_nodes,lmt_nodes),
     & G(lmt_nodes,lmt_nodes),DG(lmt_nodes),
     & WX(lmt_nodes,lmt_nodes),QGAS(lmt_nodes),
     & rsodo(lmt_nodes),rswdw(lmt_nodes),
     & GX(lmt_nodes,lmt_nodes),rswb(lmt_nodes,lmt_nodes),
     & cacw(lmt_nodes,lmt_nodes),rsoa(lmt_nodes,lmt_nodes),
     & rswh(lmt_nodes,lmt_nodes),
     & fgas(lmt_nodes,lmt_nodes),fwat(lmt_nodes,lmt_nodes)
c=======================================================================
c  From Don Morton's PhD work ..this is the tie between this and his
c  PhD work.
c       el_calc_jacobian(elmt_num,jacobian, det_jacobian, jacobian_inv,
c                num_nodes_compct, coords, dnlcl)
c          integer elmt_num
c          double precision jacobian(GLB_NUM_DIMENSIONS,
c                           GLB_NUM_DIMENSIONS)
c          double precision det_jacobian
c          double precision jacobian_inv(GLB_NUM_DIMENSIONS,
c                             GLB_NUM_DIMENSIONS)
c          integer num_nodes_compct
c          double precision coords(GLB_MAX_NODES_PER_ELMT,
c                          GLB_NUM_DIMENSIONS)
c          double precision dnlcl(GLB_MAX_NODES_PER_ELMT,
c                        GLB_NUM_DIMENSIONS)
c    calculate jacobian, etc.
      double precision jacobian(3,3),    ! jacobian matrix
```

```
     &              det_jacobian,     ! determinant of jacobian
     &              jacobian_inv(3,3) ! inverse of jacobian matrix
c    double precision coords(8,3)       ! coordinates of nodes in compact list
     double precision dn(8,3)           !

     double precision cofactor(3,3)     ! cofactors of jacobian matrix
     num_nodes_compct = 8

c    initialize jacobian matrix
     do 5 i=1,3
       do 3 j=1,3
         jacobian(i,j) = 0.0d0
3      continue
5    continue

c    calculate jacobian matrix at point (xi, eta, zeta)

     do 20 k=1,num_nodes_compct
       do 15 i=1,3
         do 10 j=1,3
           jacobian(i,j) = jacobian(i,j) +
    &                  coords(k,j)*dphd(k,i)

10       continue

15     continue
20   continue

c    calculate cofactors of jacobian matrix
         cofactor(1,1) = jacobian(2,2)*jacobian(3,3) -
    &                    jacobian(2,3)*jacobian(3,2)
         cofactor(1,2) = -(jacobian(2,1)*jacobian(3,3) -
    &                     jacobian(2,3)*jacobian(3,1))
         cofactor(1,3) = jacobian(2,1)*jacobian(3,2) -
    &                    jacobian(2,2)*jacobian(3,1)

         cofactor(2,1) = -(jacobian(1,2)*jacobian(3,3) -
    &                     jacobian(1,3)*jacobian(3,2))
         cofactor(2,2) = jacobian(1,1)*jacobian(3,3) -
    &                    jacobian(1,3)*jacobian(3,1)
         cofactor(2,3) = -(jacobian(1,1)*jacobian(3,2) -
    &                     jacobian(1,2)*jacobian(3,1))

         cofactor(3,1) = jacobian(1,2)*jacobian(2,3) -
    &                    jacobian(1,3)*jacobian(2,2)
         cofactor(3,2) = -(jacobian(1,1)*jacobian(2,3) -
    &                     jacobian(1,3)*jacobian(2,1))
         cofactor(3,3) = jacobian(1,1)*jacobian(2,2) -
    &                    jacobian(1,2)*jacobian(2,1)
```

```
c    calculate determinant of jacobian matrix
      det_jacobian = 0.0d0
      do 30 i=1,3
        det_jacobian = det_jacobian + jacobian(1,i)*cofactor(1,i)
30     continue

     if(det_jacobian .lt. 1.0d-15) then
c      write(60,1000)  i_elm,det_jacobian
       stop
     endif

c    calculate inverse of jacobian - it is equal to the adjoint/determinant.
c    note that the adjoint is the transpose of the matrix of cofactors.

      do 45 i=1,3
        do 40 j=1,3
          jacobian_inv(i,j) = cofactor(j,i)/det_jacobian
40      continue

45     continue

1000  format(1x, 'jacobi_3D8(): Fatal error', /5x,
     &      'In Element #', i6, /5x,
     &      'Determinant of jacobian is not positive enough ',
     &      /5x, '(', e10.2, ')...', //5x, 'Coords', 10x,
     &      'Derivatives')
1050  format(5x, 6(e12.5,1x))

c    calculate derivatives wrt x,y,z at xi, eta, zeta
     do 310 i=1,num_nodes_compct
       dn(i,1) = jacobian_inv(1,1)*dphd(i,1) +
     &           jacobian_inv(1,2)*dphd(i,2) +
     &           jacobian_inv(1,3)*dphd(i,3)
       dn(i,2) = jacobian_inv(2,1)*dphd(i,1) +
     &           jacobian_inv(2,2)*dphd(i,2) +
     &           jacobian_inv(2,3)*dphd(i,3)
       dn(i,3) = jacobian_inv(3,1)*dphd(i,1) +
     &           jacobian_inv(3,2)*dphd(i,2) +
     &           jacobian_inv(3,3)*dphd(i,3)
310    continue
     return
     end
c=======================================================================
     SUBROUTINE GAUSS_3D8
c=======================================================================
c  This subroutine uses the shape functions for a trilinear hexahedral
c
c  description - 3 dimensions
c              8 nodes located at the vertices of a hexahedra
c              bilinear shape functions
```

```
c            see other documentation for node numbering conventions,
c            etc.
c     num_gauss_points(1) = 2
c     lmt_nodes(1) = 8
c     num_vertices(1) = 8
c     num_faces(1) = 6
      common/gauss/gauss_pts(2),gauss_wts(2),num_gauss_pts

      num_gauss_pts = 2                !That means that we have 8 for 3D
      gauss_pts(1) = -1.d0/dsqrt(3.d0)
      gauss_pts(2) = -gauss_pts(1)
      gauss_wts(1) = 1.d0
      gauss_wts(2) = 1.d0
      return
      end

c ======================================================================

      SUBROUTINE BOUNDARY_PRESS(ifem)
c=======================================================================
c  This subroutine initializes the pressures on the boundary
c  of the well-block.  These are estimated at each time-step from the
c  pressures in the reservoir blocks that adjoin the well-block.
      INCLUDE 'PARAMETR.FOR'
      INCLUDE 'PARAMETR.FEM'
      REAL pi(8), soi(8), swi(8), sgi(8)
      COMMON /PRTP/ P(NX,NY,NZ)
      COMMON /VECTOR/ DX(NX,NY,NZ),DY(NX,NY,NZ),DZ(NX,NY,NZ)
      COMMON /IQN/ IQN1(NW),IQN2(NW),IQN3(NW)
      COMMON /PRTS/ SO(NX,NY,NZ),SW(NX,NY,NZ),SG(NX,NY,NZ)
      COMMON /NUMBER/ II,JJ,KK
      COMMON/MESHPTS/ i1(num_elemts),i2(num_elemts),
     & i3(num_elemts),i4(num_elemts), r1
c#######################################################################

      common/glstiff/stiff(num_nodes*nzp,num_nodes*nzp),
     & fl(num_nodes*nzp),xl(num_nodes*nzp),ipvt(num_nodes*nzp),
     & sofl(num_nodes*nzp),sofl1(num_nodes*nzp),sofl2(num_nodes*nzp),
     & sgfl(num_nodes*nzp),sgfl1(num_nodes*nzp),sgfl2(num_nodes*nzp),
     & swfl(num_nodes*nzp),swfl1(num_nodes*nzp),swfl2(num_nodes*nzp)

      common/stiff0/stiff0(num_nodes*nzp,num_nodes*nzp),
     & sostiff(num_nodes*nzp,num_nodes*nzp),
     & sgstiff(num_nodes*nzp,num_nodes*nzp),
     & swstiff(num_nodes*nzp,num_nodes*nzp)

      common/oldpwg/pold(num_nodes*nzp),swold(num_nodes*nzp),
     & soold(num_nodes*nzp),  sgold(num_nodes*nzp)

      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
```

```
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)

      common/oilfem/ A(lmt_nodes,lmt_nodes),
     & COIJ(lmt_nodes,lmt_nodes),DO(lmt_nodes),
     & COIL(lmt_nodes,lmt_nodes),QOIL(lmt_nodes)

      common/waterfem/ H(lmt_nodes,lmt_nodes).
     & BWAT(lmt_nodes,lmt_nodes),DW(lmt_nodes),
     & CWIJ(lmt_nodes,lmt_nodes),QWAT(lmt_nodes),
     & CWAT(lmt_nodes.lmt_nodes)

      common/gasfem/ EGAS(lmt_nodes,lmt_nodes),
     & G(lmt_nodes,lmt_nodes),DG(lmt_nodes).
     & WX(lmt_nodes,lmt_nodes),QGAS(lmt_nodes),
     & rsodo(lmt_nodes),rswdw(lmt_nodes),
     & GX(lmt_nodes,lmt_nodes),rswb(lmt_nodes,lmt_nodes) ,
     & cacw(lmt_nodes,lmt_nodes),rsoa(lmt_nodes,lmt_nodes),
     & rswh(lmt_nodes,lmt_nodes) ,
     & fgas(lmt_nodes,lmt_nodes),fwat(lmt_nodes,lmt_nodes)

c#########################################################################
          i = 10
          j = 10

      do k = 1, 3

       do 20 k_el=1,8
          pi(k_el)=0.0
          soi(k_el) = 0.0
          swi(k_el) = 0.0
          sgi(k_el) = 0.0
 20    continue

          R1=(dx(i,j,k)**2+dy(i,j,k)**2)/4.0
       rtemp = alog (R1)

c ================ Nodes 87, 183, 279, and 375 ================

          R2 =((dx(10,10,k)+dx(9,9,k))**2
     &         + (dy(10,10,k)+dy(9,9,k))**2)/4.0
          pi(1)=(p(9,9,k)-p(10,10,k))*(rtemp/alog(R2))+p(10,10,k)
          soi(1)=(so(9,9,k)-so(10,10,k))*(rtemp/alog(R2))+so(10,10,k)
          swi(1)=(sw(9,9,k)-sw(10,10,k))*(rtemp/alog(R2))+sw(10,10,k)
          sgi(1)=(sg(9,9,k)-sg(10,10,k))*(rtemp/alog(R2))+sg(10,10,k)

          pfem((num_nodes-9)+num_nodes*(k-1)) = pi(1)
          pfem((num_nodes-9)+num_nodes*k)     = pi(1)
```

```
      sofem((num_nodes-9)+num_nodes*(k-1)) = soi(1)
      sofem((num_nodes-9)+num_nodes*k)   = soi(1)
      swfem((num_nodes-9)+num_nodes*(k-1)) = swi(1)
      swfem((num_nodes-9)+num_nodes*k)   = swi(1)
      sgfem((num_nodes-9)+num_nodes*(k-1)) = sgi(1)
      sgfem((num_nodes-9)+num_nodes*k)   = sgi(1)

c ================ Nodes 89, 185, 281, and 377 ================

      R2 =((dx(10,10,k)+dx(9,10,k))**2
     &     + (dy(10,10,k)+dy(9,10,k))**2)/4.0
      pi(2)=(p(9,10,k)-p(10,10,k))*(rtemp/alog(R2))+p(10,10,k)
      soi(2)=(so(9,10,k)-so(10,10,k))*(rtemp/alog(R2))+so(10,10,k)
      swi(2)=(sw(9,10,k)-sw(10,10,k))*(rtemp/alog(R2))+sw(10,10,k)
      sgi(2)=(sg(9,10,k)-sg(10,10,k))*(rtemp/alog(R2))+sg(10,10,k)
      pfem((num_nodes-7)+num_nodes*(k-1)) = pi(2)
      pfem((num_nodes-7)+num_nodes*k)   = pi(2)
      sofem((num_nodes-7)+num_nodes*(k-1)) = soi(2)
      sofem((num_nodes-7)+num_nodes*k)   = soi(2)
      swfem((num_nodes-7)+num_nodes*(k-1)) = swi(2)
      swfem((num_nodes-7)+num_nodes*k)   = swi(2)
      sgfem((num_nodes-7)+num_nodes*(k-1)) = sgi(2)
      sgfem((num_nodes-7)+num_nodes*k)   = sgi(2)

c ================ Nodes 85, 181, 277, and 373 ================

      R2 =((dx(10,10,k)+dx(10,9,k))**2
     &     + (dy(10,10,k)+dy(10,9,k))**2)/4.0
      pi(3)=(p(10,9,k)-p(10,10,k))*(rtemp/alog(R2))+p(10,10,k)
      soi(3)=(so(10,9,k)-so(10,10,k))*(rtemp/alog(R2))+so(10,10,k)
      swi(3)=(sw(10,9,k)-sw(10,10,k))*(rtemp/alog(R2))+sw(10,10,k)
      sgi(3)=(sg(10,9,k)-sg(10,10,k))*(rtemp/alog(R2))+sg(10,10,k)
      pfem((num_nodes-11)+num_nodes*(k-1)) = pi(3)
      pfem((num_nodes-11)+num_nodes*k)   = pi(3)
      sofem((num_nodes-11)+num_nodes*(k-1)) = soi(3)
      sofem((num_nodes-11)+num_nodes*k)   = soi(3)
      swfem((num_nodes-11)+num_nodes*(k-1)) = swi(3)
      swfem((num_nodes-11)+num_nodes*k)   = swi(3)
      sgfem((num_nodes-11)+num_nodes*(k-1)) = sgi(3)
      sgfem((num_nodes-11)+num_nodes*k)   = sgi(3)

c ============================================================

c     for nodes 83, 84 and 86 (and corresponding nodes for
c     k=2 & 3, we will use the equation of a straight line
c
c        ps = a * x + y
c
c        ps: pressure or saturation at a given node x
c        a : (2*(ps(85)-ps(87))/delta_x
```

```
c       y : ps(87)
c

c === Node 83
      pfem((num_nodes-13)+num_nodes*(k-1)) =
     &     2*pfem((num_nodes-11)+num_nodes*(k-1))-
     &     pfem((num_nodes-9)+num_nodes*(k-1))
      pfem((num_nodes-13)+num_nodes*k) =
     &     2*pfem((num_nodes-11)+num_nodes*k)-
     &     pfem((num_nodes-9)+num_nodes*k)

      sofem((num_nodes-13)+num_nodes*(k-1)) =
     &     2*sofem((num_nodes-11)+num_nodes*(k-1))-
     &     sofem((num_nodes-9)+num_nodes*(k-1))
      sofem((num_nodes-13)+num_nodes*k) =
     &     2*sofem((num_nodes-11)+num_nodes*k)-
     &     sofem((num_nodes-9)+num_nodes*k)

      swfem((num_nodes-13)+num_nodes*(k-1)) =
     &     2*swfem((num_nodes-11)+num_nodes*(k-1))-
     &     swfem((num_nodes-9)+num_nodes*(k-1))
      swfem((num_nodes-13)+num_nodes*k) =
     &     2*swfem((num_nodes-11)+num_nodes*k)-
     &     swfem((num_nodes-9)+num_nodes*k)

      sgfem((num_nodes-13)+num_nodes*(k-1)) =
     &     2*sgfem((num_nodes-11)+num_nodes*(k-1))-
     &     sgfem((num_nodes-9)+num_nodes*(k-1))
      sgfem((num_nodes-13)+num_nodes*k) =
     &     2*sgfem((num_nodes-11)+num_nodes*k)-
     &     sgfem((num_nodes-9)+num_nodes*k)

c === Node 84
      pfem((num_nodes-12)+num_nodes*(k-1)) =
     &     2*(pfem((num_nodes-11)+num_nodes*(k-1))-
     &     pfem((num_nodes-9)+num_nodes*(k-1)))*0.75+
     &     pfem((num_nodes-9)+num_nodes*(k-1))
      pfem((num_nodes-12)+num_nodes*k) =
     &     2*(pfem((num_nodes-11)+num_nodes*k)-
     &     pfem((num_nodes-9)+num_nodes*k))*0.75+
     &     pfem((num_nodes-9)+num_nodes*k)

      sofem((num_nodes-12)+num_nodes*(k-1)) =
     &     2*(sofem((num_nodes-11)+num_nodes*(k-1))-
     &     sofem((num_nodes-9)+num_nodes*(k-1)))*0.75+
     &     sofem((num_nodes-9)+num_nodes*(k-1))
      sofem((num_nodes-12)+num_nodes*k) =
     &     2*(sofem((num_nodes-11)+num_nodes*k)-
     &     sofem((num_nodes-9)+num_nodes*k))*0.75+
     &     sofem((num_nodes-9)+num_nodes*k)
```

```
      swfem((num_nodes-12)+num_nodes*(k-1)) =
     &     2*(swfem((num_nodes-11)+num_nodes*(k-1))-
     &     swfem((num_nodes-9)+num_nodes*(k-1)))*0.75+
     &     swfem((num_nodes-9)+num_nodes*(k-1))
      swfem((num_nodes-12)+num_nodes*k) =
     &     2*(swfem((num_nodes-11)+num_nodes*k)-
     &     swfem((num_nodes-9)+num_nodes*k))*0.75+
     &     swfem((num_nodes-9)+num_nodes*k)

      sgfem((num_nodes-12)+num_nodes*(k-1)) =
     &     2*(sgfem((num_nodes-11)+num_nodes*(k-1))-
     &     sgfem((num_nodes-9)+num_nodes*(k-1)))*0.75+
     &     sgfem((num_nodes-9)+num_nodes*(k-1))
      sgfem((num_nodes-12)+num_nodes*k) =
     &     2*(sgfem((num_nodes-11)+num_nodes*k)-
     &     sgfem((num_nodes-9)+num_nodes*k))*0.75+
     &     sgfem((num_nodes-9)+num_nodes*k)

c === Node 86
      pfem((num_nodes-10)+num_nodes*(k-1)) =
     &     2*(pfem((num_nodes-11)+num_nodes*(k-1))-
     &     pfem((num_nodes-9)+num_nodes*(k-1)))*0.25+
     &     pfem((num_nodes-9)+num_nodes*(k-1))
      pfem((num_nodes-10)+num_nodes*k) =
     &     2*(pfem((num_nodes-11)+num_nodes*k)-
     &     pfem((num_nodes-9)+num_nodes*k))*0.25+
     &     pfem((num_nodes-9)+num_nodes*k)

      sofem((num_nodes-10)+num_nodes*(k-1)) =
     &     2*(sofem((num_nodes-11)+num_nodes*(k-1))-
     &     sofem((num_nodes-9)+num_nodes*(k-1)))*0.25+
     &     sofem((num_nodes-9)+num_nodes*(k-1))
      sofem((num_nodes-10)+num_nodes*k) =
     &     2*(sofem((num_nodes-11)+num_nodes*k)-
     &     sofem((num_nodes-9)+num_nodes*k))*0.25+
     &     sofem((num_nodes-9)+num_nodes*k)

      swfem((num_nodes-10)+num_nodes*(k-1)) =
     &     2*(swfem((num_nodes-11)+num_nodes*(k-1))-
     &     swfem((num_nodes-9)+num_nodes*(k-1)))*0.25+
     &     swfem((num_nodes-9)+num_nodes*(k-1))
      swfem((num_nodes-10)+num_nodes*k) =
     &     2*(swfem((num_nodes-11)+num_nodes*k)-
     &     swfem((num_nodes-9)+num_nodes*k))*0.25+
     &     swfem((num_nodes-9)+num_nodes*k)

      sgfem((num_nodes-10)+num_nodes*(k-1)) =
     &     2*(sgfem((num_nodes-11)+num_nodes*(k-1))-
     &     sgfem((num_nodes-9)+num_nodes*(k-1)))*0.25+
```

```
     &       sgfem((num_nodes-9)+num_nodes*(k-1))
           sgfem((num_nodes-10)+num_nodes*k) =
     &       2*(sgfem(85+num_nodes*k)-
     &       sgfem((num_nodes-9)+num_nodes*k))*0.25+
     &       sgfem((num_nodes-9)+num_nodes*k)

c ============================================================

c     for nodes 88, 90, and 91 (and corresponding nodes for
c     k=2 & 3, we will use the equation of a straight line
c
c        ps = a * x + y
c
c        ps: pressure or saturation at a given node x
c        a : (2*(ps(89)-ps(87))/delta_x
c        y : ps(87)
c

c === Node 91
        pfem((num_nodes-5)+num_nodes*(k-1)) =
     &       2*pfem((num_nodes-7)+num_nodes*(k-1))-
     &       pfem((num_nodes-9)+num_nodes*(k-1))
        pfem((num_nodes-5)+num_nodes*k) =
     &       2*pfem((num_nodes-7)+num_nodes*k)-
     &       pfem((num_nodes-9)+num_nodes*k)

        sofem((num_nodes-5)+num_nodes*(k-1)) =
     &       2*sofem((num_nodes-7)+num_nodes*(k-1))-
     &       sofem((num_nodes-9)+num_nodes*(k-1))
        sofem((num_nodes-5)+num_nodes*k) =
     &       2*sofem((num_nodes-7)+num_nodes*k)-
     &       sofem((num_nodes-9)+num_nodes*k)

        swfem((num_nodes-5)+num_nodes*(k-1)) =
     &       2*swfem((num_nodes-7)+num_nodes*(k-1))-
     &       swfem((num_nodes-9)+num_nodes*(k-1))
        swfem((num_nodes-5)+num_nodes*k) =
     &       2*swfem((num_nodes-7)+num_nodes*k)-
     &       swfem((num_nodes-9)+num_nodes*k)

        sgfem((num_nodes-5)+num_nodes*(k-1)) =
     &       2*sgfem((num_nodes-7)+num_nodes*(k-1))-
     &       sgfem((num_nodes-9)+num_nodes*(k-1))
        sgfem((num_nodes-5)+num_nodes*k) =
     &       2*sgfem((num_nodes-7)+num_nodes*k)-
     &       sgfem((num_nodes-9)+num_nodes*k)

c === Node 88
        pfem((num_nodes-8)+num_nodes*(k-1)) =
     &       2*(pfem((num_nodes-7)+num_nodes*(k-1))-
```

```
     &      pfem((num_nodes-9)+num_nodes*(k-1)))*0.25+
     &      pfem((num_nodes-9)+num_nodes*(k-1))
        pfem((num_nodes-8)+num_nodes*k) =
     &      2*(pfem((num_nodes-7)+num_nodes*k)-
     &      pfem((num_nodes-9)+num_nodes*k))*0.25+
     &      pfem((num_nodes-9)+num_nodes*k)

        sofem((num_nodes-8)+num_nodes*(k-1)) =
     &      2*(sofem((num_nodes-7)+num_nodes*(k-1))-
     &      sofem((num_nodes-9)+num_nodes*(k-1)))*0.25+
     &      sofem((num_nodes-9)+num_nodes*(k-1))
        sofem((num_nodes-8)+num_nodes*k) =
     &      2*(sofem((num_nodes-7)+num_nodes*k)-
     &      sofem((num_nodes-9)+num_nodes*k))*0.25+
     &      sofem((num_nodes-9)+num_nodes*k)

        swfem((num_nodes-8)+num_nodes*(k-1)) =
     &      2*(swfem((num_nodes-7)+num_nodes*(k-1))-
     &      swfem((num_nodes-9)+num_nodes*(k-1)))*0.25+
     &      swfem((num_nodes-9)+num_nodes*(k-1))
        swfem((num_nodes-8)+num_nodes*k) =
     &      2*(swfem((num_nodes-7)+num_nodes*k)-
     &      swfem((num_nodes-9)+num_nodes*k))*0.25+
     &      swfem((num_nodes-9)+num_nodes*k)

        sgfem((num_nodes-8)+num_nodes*(k-1)) =
     &      2*(sgfem((num_nodes-7)+num_nodes*(k-1))-
     &      sgfem((num_nodes-9)+num_nodes*(k-1)))*0.25+
     &      sgfem((num_nodes-9)+num_nodes*(k-1))
        sgfem((num_nodes-8)+num_nodes*k) =
     &      2*(sgfem((num_nodes-7)+num_nodes*k)-
     &      sgfem((num_nodes-9)+num_nodes*k))*0.25+
     &      sgfem((num_nodes-9)+num_nodes*k)

c === Node 90
        pfem((num_nodes-6)+num_nodes*(k-1)) =
     &      2*(pfem((num_nodes-7)+num_nodes*(k-1))-
     &      pfem((num_nodes-9)+num_nodes*(k-1)))*0.75+
     &      pfem((num_nodes-9)+num_nodes*(k-1))
        pfem((num_nodes-6)+num_nodes*k) =
     &      2*(pfem((num_nodes-7)+num_nodes*k)-
     &      pfem((num_nodes-9)+num_nodes*k))*0.75+
     &      pfem((num_nodes-9)+num_nodes*k)

        sofem((num_nodes-6)+num_nodes*(k-1)) =
     &      2*(sofem((num_nodes-7)+num_nodes*(k-1))-
     &      sofem((num_nodes-9)+num_nodes*(k-1)))*0.75+
     &      sofem((num_nodes-9)+num_nodes*(k-1))
        sofem((num_nodes-6)+num_nodes*k) =
     &      2*(sofem((num_nodes-7)+num_nodes*k)-
```

```
     &      sofem((num_nodes-9)+num_nodes*k))*0.75+
     &      sofem((num_nodes-9)+num_nodes*k)

        swfem((num_nodes-6)+num_nodes*(k-1)) =
     &      2*(swfem((num_nodes-7)+num_nodes*(k-1))-
     &      swfem((num_nodes-9)+num_nodes*(k-1)))*0.75+
     &      swfem((num_nodes-9)+num_nodes*(k-1))
        swfem((num_nodes-6)+num_nodes*k) =
     &      2*(swfem((num_nodes-7)+num_nodes*k)-
     &      swfem((num_nodes-9)+num_nodes*k))*0.75+
     &      swfem((num_nodes-9)+num_nodes*k)

        sgfem((num_nodes-6)+num_nodes*(k-1)) =
     &      2*(sgfem((num_nodes-7)+num_nodes*(k-1))-
     &      sgfem((num_nodes-9)+num_nodes*(k-1)))*0.75+
     &      sgfem((num_nodes-9)+num_nodes*(k-1))
        sgfem((num_nodes-6)+num_nodes*k) =
     &      2*(sgfem((num_nodes-7)+num_nodes*k)-
     &      sgfem((num_nodes-9)+num_nodes*k))*0.75+
     &      sgfem((num_nodes-9)+num_nodes*k)
       end do

668    format(3x,4(f10.5,3x))
667    format(3x,f10.5,4x,f10.5,5x,f10.5,6x,f10.5)

       return
       end
c=======================================================================
c
       SUBROUTINE FEMETHOD(ifem,delt,ttltime)
c=======================================================================
c  This subroutine will be the driver for the FEM method on one well-block.
c  It will call other subroutines that will assemble, impose boundary
c  boundary conditions, and get a solution with the solver of "choice."
       INCLUDE 'PARAMETR.FOR'
       INCLUDE 'PARAMETR.FEM'
       real(8) t0,t
       double precision solution_ite(num_nodes*nzp,50)
       double precision stiff,fl,xl,ddd ,flux_tmp(num_nodes*nzp)
       COMMON /NUMBER/II,JJ,KK
       COMMON /PRTP/ P(NX,NY,NZ)
       COMMON /PRTS/ SO(NX,NY,NZ),SW(NX,NY,NZ),SG(NX,NY,NZ)
c######################################################################

       common/glstiff/stiff(num_nodes*nzp,num_nodes*nzp),
     & fl(num_nodes*nzp),xl(num_nodes*nzp),ipvt(num_nodes*nzp),
     & sofl(num_nodes*nzp),sofl1(num_nodes*nzp),sofl2(num_nodes*nzp),
     & sgfl(num_nodes*nzp),sgfl1(num_nodes*nzp),sgfl2(num_nodes*nzp),
     & swfl(num_nodes*nzp),swfl1(num_nodes*nzp),swfl2(num_nodes*nzp)
```

```
      common/stiff0/stiff0(num_nodes*nzp,num_nodes*nzp),
     &  sostiff(num_nodes*nzp,num_nodes*nzp),
     &  sgstiff(num_nodes*nzp,num_nodes*nzp),
     &  swstiff(num_nodes*nzp,num_nodes*nzp)

      common/oldpwg/pold(num_nodes*nzp),swold(num_nodes*nzp),
     &  soold(num_nodes*nzp),  sgold(num_nodes*nzp)

      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)

      common/oilfem/ A(lmt_nodes,lmt_nodes),
     & COIJ(lmt_nodes,lmt_nodes),DO(lmt_nodes),
     & COIL(lmt_nodes,lmt_nodes),QOIL(lmt_nodes)

      common/waterfem/ H(lmt_nodes,lmt_nodes),
     & BWAT(lmt_nodes,lmt_nodes),DW(lmt_nodes),
     & CWIJ(lmt_nodes,lmt_nodes),QWAT(lmt_nodes),
     & CWAT(lmt_nodes,lmt_nodes)

      common/gasfem/ EGAS(lmt_nodes,lmt_nodes),
     & G(lmt_nodes,lmt_nodes),DG(lmt_nodes),
     & WX(lmt_nodes,lmt_nodes),QGAS(lmt_nodes),
     & rsodo(lmt_nodes),rswdw(lmt_nodes),
     & GX(lmt_nodes,lmt_nodes),rswb(lmt_nodes,lmt_nodes) ,
     & cacw(lmt_nodes,lmt_nodes),rsoa(lmt_nodes,lmt_nodes),
     & rswh(lmt_nodes,lmt_nodes) ,
     & fgas(lmt_nodes,lmt_nodes),fwat(lmt_nodes,lmt_nodes)

c#####################################################################


      call boundary_press (ifem)

      niter=1
      delta = delt
      do in = 1, niter
c       t0=MCLOCK()
          do m=1, num_nodes*4
            pold(m) = pfem(m)
            swold(m)= swfem(m)
            soold(m)= sofem(m)
            sgold(m)= sgfem(m)
          end do

       call INIT_GLOBAL_STIFF
```

```
      call ELMT_ASSEMBLE(ifem,delta,ttltime)

      call LOAD_MODS (ifem,3,delta)

      call CONSTRAIN_SYSTEM

      call ludcmp (stiff,num_nodes*4,num_nodes*4,ipvt,ddd)
      call lubksb (stiff,num_nodes*4,num_nodes*4,ipvt,fl)
C      use essl solver for linear equation
c      call DGEF(stiff,num_nodes*4,num_nodes*4,ipvt)
c      call DGES(stiff,num_nodes*4,num_nodes*4,ipvt,fl,0)

     do 2000 j = 1, num_nodes*4
       xl(j) = pfem(j)
       pold(j) = pfem(j)
       if (fl(j) .le. 0.0) then
c        fl(j) = 0.0
         fl(j) = -fl(j)
       endif
       if (fl(j) .lt. pfem(j)) then
         pfem(j) = fl(j)
       endif
2000   continue

c     do 2001 j = 1, num_nodes*4
c       sosum=0.0
c       swsum=0.0
c       do jk = 1, num_nodes*4
c         sosum=sosum+sostiff(j,jk)*pfem(jk)
c         swsum=swsum+swstiff(j,jk)*pfem(jk)
c         write(62,*)' sostiff(', j, ',', jk, ') =',
c  &            sostiff(j,jk)
c       end do
c         sofl(j)=sofl(j)+sosum
c         swfl(j)=swfl(j)+swsum

     do 2001 j = 1, num_nodes*4
       sosum=0.0
       swsum=0.0
       do jk = 1, num_nodes*4
         sosum=sosum+sostiff(j,jk)
         swsum=swsum+swstiff(j,jk)
       end do
         sofl(j)=sofl(j)+sosum*pfem(j)
         swfl(j)=swfl(j)+swsum*pfem(j)

       sofl(j) = (sofl(j)
  +     +sofl1(j)+sofem(j))/(1.0+sofl2(j)*
  +     (pfem(j)-pold(j)))
```

```
      swfl(j) = (swfl(j)
     +    +swfl1(j)+swfem(j))/(1.0+swfl2(j)*
     +    (pfem(j)-pold(j)))

      sofem(j) = sofl(j)
      swfem(j) = swfl(j)
      if(swfem(j).lt.0.0 ) swfem(j)=1.0e-5
      if(sofem(j).lt.0.0) sofem(j)=1.0e-5
          sgfem(j)=1.0-swfem(j)-sofem(j)
      if(sgfem(j).lt.0.0 ) sgfem(j)=1.0e-5

2001  continue

     end do

      write(61,*)ttltime
      write(62,*)ttltime
      write(63,*)ttltime
      write(64,*)ttltime
      do ik = 1, 3
        do i = 1, num_nodes
          write(61,*)j,pfem(i+num_nodes*(ik-1)),
     +      pfem(i+num_nodes*ik)
          write(62,*)j,sofem(i+num_nodes*(ik-1)),
     +      sofem(i+num_nodes*ik)
          write(63,*)j,swfem(i+num_nodes*(ik-1)),
     +      swfem(i+num_nodes*ik)
          write(64,*)j,sgfem(i+num_nodes*(ik-1)),
     +      sgfem(i+num_nodes*ik)
        enddo
      enddo

      write(71,*)ttltime,pfem(285),sofem(285),
     +     swfem(285), sgfem(285)
      write(72,*)ttltime,pfem(287),sofem(287),
     +     swfem(287), sgfem(287)
      write(73,*)ttltime,pfem(300),sofem(300),
     +     swfem(300), sgfem(300)
      write(74,*)ttltime,pfem(310),sofem(310),
     +     swfem(310), sgfem(310)
      write(75,*)ttltime,pfem(350),sofem(350),
     +     swfem(350), sgfem(350)
      write(76,*)ttltime,pfem(360),sofem(360),
     +     swfem(360), sgfem(360)
      write(77,*)ttltime,pfem(370),sofem(370),
     +     swfem(370), sgfem(370)
      write(78,*)ttltime,pfem(376),sofem(376),
     +     swfem(376), sgfem(376)
      write(79,*)ttltime,pfem(380),sofem(380),
     +     swfem(380), sgfem(380)
```

```
c       call outpwg(delta,ttltime)

        call AVEPWG(ifem,ttltime)

c       t=MCLOCK()-t0

          return
          end

          subroutine outpwg(delt,ttltime)
          INCLUDE 'PARAMETR.FOR'
          INCLUDE 'PARAMETR.FEM'
          COMMON /FEMRATE/
           QWWW(NZ),QWE(NZ),QWS(NZ),QWN(NZ),QWT(NZ),QWB(NZ),
     &  QOW(NZ),QOE(NZ),QOS(NZ),QON(NZ),QOT(NZ),QOB(NZ),
     &  QGW(NZ),QGE(NZ),QGS(NZ),QGN(NZ),QGT(NZ),QGB(NZ)
      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)

        write(30,*)
        write(30,9909)delt, ttltime
9909      format(2x,'  dt= ',f8.4,'  ttltime= ',f12.4)
        write(30,*)'================ The Pressure ================='
        write(30,*)
        write(30,*)' Node     Layer1       Layer2       ',
     &    'Layer3       Layer3'
        write(30,*)' Number    Top          Top          ',
     &    ' Top         Bottom'
        do i = 1, num_nodes
         write(30,2) i,pfem(i),pfem(i+num_nodes),
     &      pfem(i+num_nodes*2),pfem(i+num_nodes*3)
2        format(2x,i4,3x, 4(2x,f12.5))

        end do

c       write(30,*)
c       write(30,*)'================ SOfem ====================='
c
c       write(30,*)' Node       Layer1         Layer2       ',
c     &    'Layer3       Layer3'
c       write(30,*)' Number    Top          Top          ',
c     &    ' Top         Bottom'
c       do i = 1, 16
c        write(30,2) i, sofem(i), sofem(i+num_nodes),
c     &      sofem(i+num_nodes*2), sofem(i+num_nodes*3)
c       end do
```

```
c      write(30,*)

c      write(30,*)' Node      Layer1       Layer2      ',
c     &   'Layer3      Layer3'
c      write(30,*)' Number    Top          Top         ',
c     &   ' Top         Bottom'
c      do i = num_nodes-15, num_nodes
c       write(30,3) i, sofem(i), sofem(i+num_nodes),
c     &       sofem(i+num_nodes*2), sofem(i+num_nodes*3)
c      end do
c      write(30,*)
c      write(30,*)'================== SWfem ==================='

c      write(30,*)
c      write(30,*)' Node      Layer1       Layer2      ',
c     &   'Layer3      Layer3'
c      write(30,*)' Number    Top          Top         ',
c     &   ' Top         Bottom'
c      do i = 1, 16
c       write(30,2) i, swfem(i), swfem(i+96),
c     &       swfem(i+192), swfem(i+288)
c      end do
c      write(30,*)

c      write(30,*)' Node      Layer1       Layer2      ',
c     &   'Layer3      Layer3'
c      write(30,*)' Number    Top          Top         ',
c     &   ' Top         Bottom'
c      do i = 81, 96
c       write(30,3) i, swfem(i), swfem(i+96),
c     &       swfem(i+192), swfem(i+288)
c      end do
c      write(30,*)
c      write(30,*)'================== SGfem ==================='

c      write(30,*)
c      write(30,*)' Node      Layer1       Layer2      ',
c     &   'Layer3      Layer3'
c      write(30,*)' Number    Top          Top         ',
c     &   ' Top         Bottom'
c      do i = 1, 16
c       write(30,2) i, sgfem(i), sgfem(i+96),
c     &       sgfem(i+192), sgfem(i+288)
c      end do
c      write(30,*)

c      write(30,*)' Node      Layer1       Layer2      ',
c     &   'Layer3      Layer3'
c      write(30,*)' Number    Top          Top         ',
c     &   ' Top         Bottom'
```

```
c      do i = 81, 96
c        write(30,3) i, sgfem(i), sgfem(i+96),
c  &         sgfem(i+192), sgfem(i+288)
c      end do
c      write(30,*)'*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+'.
c  &     '*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+'

         return
         end
c=======================================================================
c
      SUBROUTINE ELMT_ASSEMBLE(ifem,delt,ttltime)
c
c  This subroutine assembles one element for the FEM-method on one well-block.

      INCLUDE 'PARAMETR.FOR'
      INCLUDE 'PARAMETR.FEM'
c=======================================================================
      double precision det_jacobian      ! determinant of jacobian
      double precision dn(8,3), dpr(num_nodes*nzp)
      double precision stiff,fl,xl,fl_p(num_nodes*nzp)
      double precision swat,soil
      double precision oil_stiff,wat_stiff
           double precision stiff0,stiff_eg(num_nodes*nzp)
c=======================================================================
      REAL KX,KY,KZ,KROT,KRWT,KRGT,MUOT,MUWT,MUGT,MCFG,MCFG1,MCFGT,
     & MCFGI,MBEO,MBEW,MBEG
      REAL muo,muw,mug,kro,krw,krg
      INTEGER indx(lmt_nodes)
      double precision flux_tmp(num_nodes*nzp)
      COMMON /BUBBLE/ PBO,VSLOPE,BSLOPE,RSLOPE,PMAXT,IREPRS,
     & RHOSCO,RHOSCG,RHOSCW,MSAT,MPOT,MPWT,MPGT,PBOT(NX,NY,NZ)
      COMMON /COEF/ AW(NX,NY,NZ),AE(NX,NY,NZ),AN(NX,NY,NZ),
     & AS(NX,NY,NZ),AB(NX,NY,NZ),AT(NX,NY,NZ),E(NX,NY,NZ),B(NX,NY,NZ)
      COMMON /SARRAY/ PN(NX,NY,NZ),
     & SON(NX,NY,NZ),SWN(NX,NY,NZ),SGN(NX,NY,NZ),
     & SO1(NX,NY,NZ),SW1(NX,NY,NZ),SG1(NX,NY,NZ),
     & A1(NX,NY,NZ),A2(NX,NY,NZ),A3(NX,NY,NZ),
     & SUM(NX,NY,NZ),GAM(NX,NY,NZ),QS(NX,NY,NZ)
      COMMON /PERM/ KX(NX,NY,NZ),KY(NX,NY,NZ),KZ(NX,NY,NZ)
      COMMON /TRAN/ TX(NX+1,NY,NZ),TY(NX,NY+1,NZ),TZ(NX,NY,NZ+1)
      COMMON /ELEV/ EL(NX,NY,NZ)
      COMMON /PRTP/ P(NX,NY,NZ)
      COMMON /PRTS/ SO(NX,NY,NZ),SW(NX,NY,NZ),SG(NX,NY,NZ)
      COMMON /SPVT/ SAT(NTE),KROT(NTE),KRWT(NTE),KRGT(NTE),PCOWT(NTE),
     & PCGOT(NTE),POT(NTE),MUOT(NTE),BOT(NTE),BOPT(NTE),RSOT(NTE),RSOPT
     & (NTE),PWT(NTE),MUWT(NTE),BWT(NTE),BWPT(NTE),RSWT(NTE),RSWPT(NTE),
     & PGT(NTE),MUGT(NTE),BGT(NTE),BGPT(NTE),CRT(NTE)
      COMMON /SRATE/ PID(NW,NL),PWF(NW,NL),PWFC(NW,NL),KIP(NW),
     & GMO(NW,NL),GMW(NW,NL),GMG(NW,NL),LAYER(NW),QVO(NW),
```

```
& QVW(NW),QVG(NW),QVT(NW),CUMO(NW,NL),CUMW(NW,NL),CUMG(NW,NL)
 COMMON /VOLFAC/ BO(NX,NY,NZ),BW(NX,NY,NZ),BG(NX,NY,NZ)
 COMMON /RATE/ QO(NX,NY,NZ),QW(NX,NY,NZ),QG(NX,NY,NZ)
 COMMON /TERM1/ GOWT(NX,NY,NZ),GWWT(NX,NY,NZ),GGWT(NX,NY,NZ),
& OW(NX+1,NY,NZ),OE(NX+1,NY,NZ),WW(NX+1,NY,NZ),WE(NX+1,NY,NZ),
& OS(NX,NY+1,NZ),ON(NX,NY+1,NZ),WS(NX,NY+1,NZ),WN(NX,NY+1,NZ),
& OT(NX,NY,NZ+1),OB(NX,NY,NZ+1),WT(NX,NY,NZ+1),WB(NX,NY,NZ+1)
& ,GW(NX+1,NY,NZ),GE(NX+1,NY,NZ),
& GS(NX,NY+1,NZ),GN(NX,NY+1,NZ),
& GT(NX,NY,NZ+1),GB(NX,NY,NZ+1)
 COMMON /TERM2/ QOWG(NX,NY,NZ)
 COMMON /COMPRS/ CT(NX,NY,NZ)
 COMMON /PORE/ VP(NX,NY,NZ)
 COMMON /VECTOR/ DX(NX,NY,NZ),DY(NX,NY,NZ),DZ(NX,NY,NZ)
 COMMON /IQN/ IQN1(NW),IQN2(NW),IQN3(NW)
 COMMON /IQH/ IQH1(NW,NL),IQH2(NW,NL),IQH3(NW,NL),COND(NW)
 COMMON /CODE/ KSM1,KSN1,KCO1,NN,FACT1,FACT2,TMAX,KSOL,MITER,
& OMEGA,TOL,TOL1,KSN,KSM,KCO,KTR,KCOFF,DSMAX,DPMAX,WORMAX,
& GORMAX,PAMIN,PAMAX
 COMMON /ADD1/ IM,JM,KM,ETI,FT,FTMAX
 COMMON /ADD2/ COP,CWP,CGP,CWI,CGI
 COMMON /PSCNTL/ KPI,KSI
 COMMON /MBE/ MBEO,MBEW,MBEG
 COMMON /NUMBER/ II,JJ,KK
 COMMON /VOL/ SCFO,SCFW,SCFG,SCFGI,STEO,STBW,MCFG,MCFGI,MCFGT,
& MCFGI,STBOI,STBWI,RESVOL
 COMMON /BAL/ OP,WP,GP,WI,GI,PAVG0,PAVG,QPR,WPR,GPR,WIR,GIR,CWOR,
& WOR,CGOR,GOR
 COMMON /PRTCNT/ IWLCNG,ICHANG,IWLREP,ISUMRY,
& IPMAP,ISOMAP,ISWMAP,ISGMAP,IPBMAP
 COMMON /IO2/ NO2
 COMMON /TTEST/ NUMPRD,SONTVL
 COMMON /SWTCH/ NGRSW,NTRSW,NRESTART,NTS
 COMMON /R1XYZ/ R11X,R21X,R31X,R41X,R11Y,R21Y,R31Y,R41Y,
& R11Z,R21Z,R31Z,R41Z
 COMMON /T1XYZ/ T1X,T2X,T3X,T4X,T1Y,T2Y,T3Y,T4Y,T1Z,T2Z,T3Z,T4Z
 COMMON /DXYZ/ DX0,DXP,DXM,XW,DY0,DYP,DYM,YW,DZ0,DZP,DZM,ZW
 COMMON /RXYZ/ RX,RY,RZ,AX,AY,AZ,I,J,K
 COMMON /COUNT/ N1READ,N2READ
 COMMON /RS/ ROS
 COMMON /DECLINE/ DEC(NW,NL)
 COMMON /FEMRATE/
& QWWW(NZ),QWE(NZ),QWS(NZ),QWN(NZ),QWT(NZ),QWB(NZ),
&  QOW(NZ),QOE(NZ),QOS(NZ),QON(NZ),QOT(NZ),QOB(NZ),
&  QGW(NZ),QGE(NZ),QGS(NZ),QGN(NZ),QGT(NZ),QGB(NZ)
 COMMON/MESHPTS/ i1(num_elemts),i2(num_elemts),
& i3(num_elemts),i4(num_elemts), r1
 common/gauss/gauss_pts(2),gauss_wts(2),num_gauss_pts
 common/shape/phif(lmt_nodes),
& dphd(lmt_nodes,3),coords(lmt_nodes,3)
```

```
c#####################################################################

      common/glstiff/stiff(num_nodes*nzp,num_nodes*nzp),
     & fl(num_nodes*nzp),xl(num_nodes*nzp),ipvt(num_nodes*nzp),
     & sofl(num_nodes*nzp),sofl1(num_nodes*nzp),sofl2(num_nodes*nzp),
     & sgfl(num_nodes*nzp),sgfl1(num_nodes*nzp),sgfl2(num_nodes*nzp),
     & swfl(num_nodes*nzp),swfl1(num_nodes*nzp),swfl2(num_nodes*nzp)

      common/stiff0/stiff0(num_nodes*nzp,num_nodes*nzp),
     & sostiff(num_nodes*nzp,num_nodes*nzp),
     & sgstiff(num_nodes*nzp,num_nodes*nzp),
     & swstiff(num_nodes*nzp,num_nodes*nzp)

      common/oldpwg/pold(num_nodes*nzp),swold(num_nodes*nzp),
     & soold(num_nodes*nzp), sgold(num_nodes*nzp)

      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)

      common/oilfem/ A(lmt_nodes,lmt_nodes),
     & COIJ(lmt_nodes,lmt_nodes),DO(lmt_nodes),
     & COIL(lmt_nodes,lmt_nodes),QOIL(lmt_nodes)

      common/waterfem/ H(lmt_nodes,lmt_nodes),
     & BWAT(lmt_nodes,lmt_nodes),DW(lmt_nodes),
     & CWIJ(lmt_nodes,lmt_nodes),QWAT(lmt_nodes),
     & CWAT(lmt_nodes,lmt_nodes)

      common/gasfem/ EGAS(lmt_nodes,lmt_nodes),
     & G(lmt_nodes,lmt_nodes),DG(lmt_nodes),
     & WX(lmt_nodes,lmt_nodes),QGAS(lmt_nodes),
     & rsodo(lmt_nodes),rswdw(lmt_nodes),
     & GX(lmt_nodes,lmt_nodes),rswb(lmt_nodes,lmt_nodes) ,
     & cacw(lmt_nodes,lmt_nodes),rsoa(lmt_nodes,lmt_nodes),
     & rswh(lmt_nodes,lmt_nodes) ,
     & fgas(lmt_nodes,lmt_nodes),fwat(lmt_nodes,lmt_nodes)
c#####################################################################
c    This section (loop 50) will change the bubble point pressure in the
c    FEM well-block calculation, if required in the BOAST data.
c#####################################################################
C**** REPRESSURIZATION ALGORITHM. **********
C    get the well id, should be from the routines which call this one

        IWELL=10
        JWELL=10
        KWELL=3
```

```
      IF (IREPRS.NE.1) then
        DO 50 I=1,num_nodes*4
          IF(SGfem(I).LE.0.0001) GO TO 50
          PP=Pfem(i)
          BPT=PBotfem(i)
          IF(PP.GT.BPT) PP=BPT
          CALL INTERP(POT,BOT,MPOT,PP,BBO)
          CALL INTERP(POT,RSOT,MPOT,PP,RSO)
          CALL INTERP(PGT,BGT,MPGT,PP,BBG)
          IF(SOfem (I).EQ.0.0) GO TO 50
          RSONEW=RSO + SGfem(I)*BBO/(SOfem(I)*BBG)
          CALL INTERP(RSOT,POT,MPOT,RSONEW,PBONEW)
          PBOTfem(i)=PBONEW
 50     CONTINUE
      endif
c#####################################################################
c    This loop 700 initializes all the element matrices and vectors that
c     are required for the FEM well-block calculation.
c#####################################################################
      do 700 j=1,lmt_nodes
       do 710 i=1,lmt_nodes
         A(i,j) = 0.0
         COIJ(i,j) = 0.0
         COIL(i,j) = 0.0
         H(i,j) = 0.0
         BWAT(i,j) = 0.0
         CWIJ(i,j) = 0.0
         CWAT(i,j) = 0.0
         EGAS(i,j) = 0.0
         G(i,j) = 0.0
         WX(i,j) = 0.0
         GX(i,j) = 0.0
         fgas(i,j) = 0.0
         fwat(i,j) = 0.0
         rswb(i,j) = 0.0
         rsoa(i,j) = 0.0
         rswh(i,j) = 0.0
         cacw(i,j) = 0.0
 710   continue
      DO(j) = 0.0
      DW(j) = 0.0
      DG(j) = 0.0
      QOIL(j) = 0.0
      QWAT(j) = 0.0
      QGAS(j) = 0.0
      rsodo(j) = 0.0
      rswdw(j) = 0.0
 700  continue
c#####################################################################
```

```
      do in = 1, num_nodes*4
       qofem(in) = 0.0
       qwfem(in) = 0.0
       qgfem(in) = 0.0
      end do

c     write(30,*)'================================================='
c     write(30,*)' ttltime =', ttltime

      do 100 k = 1, 3

      do 100 k_el=1,num_elemts

        indx(1) = i1(k_el)+num_nodes*(k-1)
        indx(2) = i2(k_el)+num_nodes*(k-1)
        indx(3) = i3(k_el)+num_nodes*(k-1)
        indx(4) = i4(k_el)+num_nodes*(k-1)
        indx(5) = i1(k_el)+num_nodes*k
        indx(6) = i2(k_el)+num_nodes*k
        indx(7) = i3(k_el)+num_nodes*k
        indx(8) = i4(k_el)+num_nodes*k

c         write(60,606)(indx(i),i=1,8)
606       format(2x,8(i4,2x))

      call elmtGEN(ifem,k_el,k)

       do 300 jx_pt = 1,num_gauss_pts
       xi = gauss_pts(jx_pt)
       do 290 jy_pt = 1,num_gauss_pts
        eta = gauss_pts(jy_pt)
        do 280 jz_pt = 1,num_gauss_pts
         zeta = gauss_pts(jz_pt)
        call shape_3d8(xi,eta,zeta)
        call JACOBI_3D8(k_el,det_jacobian,dn)
c#######################################################################
c  Calculate the oil pressure, water saturation, gas saturation,
c  oil saturation, and the bubble point pressure at the Gauss point
c#######################################################################
        gauss_pt_press = 0.0
        gauss_pt_bpress = 0.0
        gauss_pt_oilsat = 0.0
        gauss_pt_watsat = 0.0
        gauss_pt_gassat = 0.0
        gauss_pt_qoil = 0.0
        gauss_pt_qwat = 0.0
        gauss_pt_qgas = 0.0
       do 60 i=1,lmt_nodes
         gauss_pt_press = gauss_pt_press + pfem(indx(i))*phif(i)
         gauss_pt_bpress = gauss_pt_bpress + pbotfem(indx(i))*phif(i)
```

```
      gauss_pt_oilsat = gauss_pt_oilsat + sofem(indx(i))*phif(i)
      gauss_pt_watsat = gauss_pt_watsat + swfem(indx(i))*phif(i)
      gauss_pt_gassat = gauss_pt_gassat + sgfem(indx(i))*phif(i)
      gauss_pt_qoil = gauss_pt_qoil + qofem(indx(i))*phif(i)
      gauss_pt_qwat = gauss_pt_qwat + qwfem(indx(i))*phif(i)
      gauss_pt_qgas = gauss_pt_qgas + qgfem(indx(i))*phif(i)
      gauss_pt_qoil = 0.
      gauss_pt_qwat = 0.
      gauss_pt_qgas = 0.
60    continue
      pp = gauss_pt_press
      bpt = gauss_pt_bpress
      CALL INTPVT(BPT,RSLOPE,POT,RSOT,MPOT,PP,RSO)
      CALL INTPVT(BPT,VSLOPE,POT,MUOT,MPOT,PP,MUO)
      CALL INTERP(PWT,RSWT,MPWT,PP,RSW)
      CALL INTERP(PWT,MUWT,MPWT,PP,MUW)
      CALL INTERP(PGT,MUGT,MPGT,PP,MUG)
      CALL INTPVT(BPT,BSLOPE,POT,BOT,MPOT,PP,BOfem)
      CALL INTERP(PWT,BWT,MPWT,PP,BWfem)
      CALL INTERP(PGT,BGT,MPGT,PP,BGfem)
      SSO = gauss_pt_oilsat
      SSW = gauss_pt_watsat
      SSG = gauss_pt_gassat
      CALL INTERP(SAT,PCOWT,MSAT,SSW,PCOW)
      CALL INTERP(SAT,PCGOT,MSAT,SSG,PCGO)
      if (ssw.gt.0.5543) then
        dpcow_dsw = 0.0
      else
        dpcow_dsw = -1200.0
      endif
      if (ssg.gt.0.5543) then
        dpcgo_dsg = 0.0
      else
        dpcgo_dsg = -1200.0
      endif
c     This calculates dBo/dPo.
      CALL INTERP(POT,BOPT,MPOT,PP,BODER)
c     This calculates dRso/dPo.
      CALL INTERP(POT,RSOPT,MPOT,PP,RSODER)
c     This calculates dBw/dPo.
      CALL INTERP(PWT,BWPT,MPWT,PP,BWDER)
c     This calculates dRsw/dPo.
      CALL INTERP(PWT,RSWPT,MPWT,PP,RSWDER)
c     This calculates dBg/dPo.
      CALL INTERP(PGT,BGPT,MPGT,PP,BGDER)

      IF(PP.GT.BPT) then
        BODER=BSLOPE
        RSODER=RSLOPE
      endif
```

```
      CO=-(BODER-BGfem*RSODER)/BOfem
      CW=-(BWDER-BGfem*RSWDER)/BWfem
      CG=-BGDER/BGfem
      CALL INTERP(PGT,CRT,MPGT,PP,CR)

      tc = cr+CO*sofem(indx(i))+CW*swfem(indx(i))+CG*sgfem(indx(i))

      alpha0 = cr - boder/bofem
      alpha1 = cr - bwder/bwfem
      calpha = (rso*cr + rsoder - rso*boder/bofem)*bgfem/bofem
      calpha1 = calpha -bgfem*rsw*cr/bwfem-bgfem*rswder/bwfem
     &   +bgfem*rsw*bwder/(bwfem*bwfem)

      CALL INTERP(SAT,KROT,MSAT,SSO,KRO)
      CALL INTERP(SAT,KRWT,MSAT,SSW,KRW)
      CALL INTERP(SAT,KRGT,MSAT,SSG,KRG)

        do 210 i=1,lmt_nodes
        do 200 j=1,lmt_nodes

c#####################  oil  equations  ################################
          dni_mob_dnj = 0.0
          phase_mob = -kro/(muo*bofem)*ffactor*phif(k)
          dni_mob_dnj = dni_mob_dnj +
     &      dn(i,1)*phase_mob*dn(j,1)*kx(iwell,jwell,kwell)+
     &      dn(i,2)*phase_mob*dn(j,2)*ky(iwell,jwell,kwell)+
     &      dn(i,3)*phase_mob*dn(j,3)*kz(iwell,jwell,kwell)
          a(i,j) = a(i,j)+det_jacobian*dni_mob_dnj
          rsoa(i,j) = rsoa(i,j)+rso*det_jacobian*dni_mob_dnj
          f2 = det_jacobian*phif(i)*phif(j)
          coil(i,j) = coil(i,j)+f2*phi(k)/bofem
          fgas(i,j) = fgas(i,j)+(rso-bofem/bgfem)*f2*phi(k)/bofem
          fsum = 0.0
        do 130 l=1,lmt_nodes
          fsum = fsum + (swfem(indx(l))+sgfem(indx(l)))*f2*phif(l)
  130   continue
        coij(i,j) = coij(i,j)+ (fsum-f2)*phi(k)*alpha0/bofem

c#####################  water equations  ################################
        dni_mob_dnj = 0.0
        phase_mob = -krw/(muw*bwfem)*ffactor*phif(k)
        dni_mob_dnj = dni_mob_dnj +
     &            dn(i,1)*phase_mob*dn(j,1)*kx(iwell,jwell,kwell)+
     &            dn(i,2)*phase_mob*dn(j,2)*ky(iwell,jwell,kwell)+
     &            dn(i,3)*phase_mob*dn(j,3)*kz(iwell,jwell,kwell)
        h(i,j) = h(i,j)+det_jacobian*dni_mob_dnj
        rswh(i,j) = rswh(i,j)+rsw*det_jacobian*dni_mob_dnj
        bwat(i,j) = bwat(i,j)-dpcow_dsw*det_jacobian*dni_mob_dnj
     &     *ffactor
        rswb(i,j) = rswb(i,j)-rsw*dpcow_dsw*det_jacobian*dni_mob_dnj
```

```
     &              *ffactor
c      cwat(i,j) = cwat(i,j)-f2*phi(k)/bwfem
       cwat(i,j) = cwat(i,j)+f2*phi(k)*d_bw
       cacw(i,j) = cacw(i,j)-calpha*f2*phi(k)/bgfem
       fwat(i,j) = fwat(i,j)-(rsw-rso*bwfem/bofem)*f2*phi(k)/bwfem
       fsum = 0.0
       do 230 l=1,lmt_nodes
        fsum = fsum + swfem(indx(l))*f2*phif(l)
  230   continue
       cwij(i,j) = cwij(i,j)- fsum*phi(k)*alpha1/bwfem
c#####################  gas equations  ################################
       dni_mob_dnj = 0.0
       phase_mob = -krg/(mug*bgfem)*ffactor*phif(k)
       dni_mob_dnj = dni_mob_dnj +
     &      dn(i,1)*phase_mob*dn(j,1)*kx(iwell,jwell,kwell)+
     &      dn(i,2)*phase_mob*dn(j,2)*ky(iwell,jwell,kwell)+
     &      dn(i,3)*phase_mob*dn(j,3)*kz(iwell,jwell,kwell)

       egas(i,j) = egas(i,j)+det_jacobian*dni_mob_dnj
       g(i,j) = g(i,j)+det_jacobian*dni_mob_dnj*dpcgo_dsg*ffactor
       wx(i,j) = wx(i,j)+ fsum*phi(k)*calpha1/bgfem
       fsum = 0.0
       do 330 l=1,lmt_nodes
        fsum = fsum + sgfem(indx(l))*f2*phif(l)
  330   continue
       gx(i,j) = gx(i,j)- fsum*phi(k)*(cr+cg-calpha)/bgfem
  200   continue

  210 continue
  280 continue
  290 continue
  300 continue
c=================================================================
C  This section assembles the global system for the FEM method on
c  the entire well-block.
c      do i=1,lmt_nodes
        do(i)=0.0
            dw(i)=0.0
            dg(i)=0.0
        do j=1,lmt_nodes
          pp=pfem(indx(j))
          bpt=pbotfem(indx(j))
          CALL INTPVT(BPT,RSLOPE,POT,RSOT,MPOT,PP,RSO)
          CALL INTPVT(BPT,BSLOPE,POT,BOT,MPOT,PP,BOfem)
              CALL INTERP(PGT,BGT,MPGT,PP,BGfem)
          rho_oil=-(rhosco+rso*rhoscg)/(144.*bofem)
          rho_wat=-rhoscw/144.0
              rho_gas=-rhoscg/(144.0*BGfem)
          do(i)=do(i)-a(i,j)*coords(j,3)*rho_oil
              dw(i)=dw(i)-h(i,j)*coords(j,3)*rho_wat
```

```
            dg(i)=dg(i)-egas(i,j)*coords(j,3)*rho_gas
          end do
              rsodo(i)=rso*do(i)
              rswdw(i)=rsw*dw(i)
        end do

        do 1000 i=1,lmt_nodes
          coil_sum = 0.0
          coij_sum = 0.0
          cwat_sum = 0.0
          cwij_sum = 0.0
          gasy_sum = 0.0
          gasf_sum = 0.0
          gasw_sum = 0.0

        do 2000 j=1,lmt_nodes
          temp = coil(i,j)
          stiff(indx(i),indx(j))=stiff(indx(i),indx(j))
     &     +delt*(bofem*a(i,j)+bwfem*h(i,j)
     &     +bgfem*egas(i,j)-phi(k)*tc)

          sostiff(indx(i),indx(j))=sostiff(indx(i),indx(j))
     &     +(bofem*delt/phi(k))*a(i,j)

          swstiff(indx(i),indx(j))=swstiff(indx(i),indx(j))
     &     +(bwfem*delt/phi(k))*h(i,j)

          gasy_sum = gasy_sum+(gx(i,j)+wx(i,j)+cacw(i,j))*pold(indx(j))
          gasf_sum = gasf_sum+fgas(i,j)*sgold(indx(j))
          gasw_sum = gasw_sum+fwat(i,j)*swold(indx(j))
2000  continue

        do(i) = 0.0

        fl(indx(i)) = fl(indx(i))+delt*((bofem-rso*bgfem)*
     &  (do(i)+qofem(indx(i)))+(bwfem-rsw*bgfem)*
     &  (dw(i)+qwfem(indx(i)))+
     &  bgfem*(dg(i)+qgfem(indx(i))))-phi(k)*tc*pfem(indx(i))

        sofl1(indx(i))=sofl1(indx(i))+(bofem*delt/phi(k))*
     & (do(i)+qofem(indx(i)))

        sofl2(indx(i))=sofl2(indx(i))+alpha0

        swfl1(indx(i))=swfl1(indx(i))+(bwfem*delt/phi(k))*
     & (dw(i)+qwfem(indx(i)))

        swfl2(indx(i))=swfl2(indx(i))+alpha1

1000  continue
```

```
 100  continue
      return
      end
c=======================================================================
C
      SUBROUTINE INIT_GLOBAL_STIFF
C
c=======================================================================
C  This subroutine initializes the global system for the FEM method on one
c  of the well-block.
      INCLUDE 'PARAMETR.FOR'
      INCLUDE 'PARAMETR.FEM'
      double precision stiff,fl,xl
      COMMON /NUMBER/ II,JJ,KK
c#######################################################################

      common/glstiff/stiff(num_nodes*nzp,num_nodes*nzp),
     &  fl(num_nodes*nzp),xl(num_nodes*nzp),ipvt(num_nodes*nzp),
     &  sofl(num_nodes*nzp),sofl1(num_nodes*nzp),sofl2(num_nodes*nzp),
     &  sgfl(num_nodes*nzp),sgfl1(num_nodes*nzp),sgfl2(num_nodes*nzp),
     &  swfl(num_nodes*nzp),swfl1(num_nodes*nzp),swfl2(num_nodes*nzp)

      common/stiff0/stiff0(num_nodes*nzp,num_nodes*nzp),
     &  sostiff(num_nodes*nzp,num_nodes*nzp),
     &  sgstiff(num_nodes*nzp,num_nodes*nzp),
     &  swstiff(num_nodes*nzp,num_nodes*nzp)

      common/oldpwg/pold(num_nodes*nzp),swold(num_nodes*nzp),
     &  soold(num_nodes*nzp),  sgold(num_nodes*nzp)

      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)

      common/oilfem/ A(lmt_nodes,lmt_nodes),
     & COIJ(lmt_nodes,lmt_nodes),DO(lmt_nodes),
     & COIL(lmt_nodes,lmt_nodes),QOIL(lmt_nodes)

      common/waterfem/ H(lmt_nodes,lmt_nodes),
     & BWAT(lmt_nodes,lmt_nodes),DW(lmt_nodes),
     & CWIJ(lmt_nodes,lmt_nodes),QWAT(lmt_nodes),
     & CWAT(lmt_nodes,lmt_nodes)

      common/gasfem/ EGAS(lmt_nodes,lmt_nodes),
     & G(lmt_nodes,lmt_nodes),DG(lmt_nodes),
     & WX(lmt_nodes,lmt_nodes),QGAS(lmt_nodes),
     & rsodo(lmt_nodes),rswdw(lmt_nodes),
```

```
     & GX(lmt_nodes,lmt_nodes),rswb(lmt_nodes,lmt_nodes) ,
     & cacw(lmt_nodes,lmt_nodes),rsoa(lmt_nodes,lmt_nodes),
     & rswh(lmt_nodes,lmt_nodes) ,
     & fgas(lmt_nodes,lmt_nodes),fwat(lmt_nodes,lmt_nodes)

c#################################################################

      do 10 i=1,num_nodes*4
       do 20 j=1,num_nodes*4
         stiff(i,j) = 0.0
         sostiff(i,j) = 0.0
         swstiff(i,j) = 0.0
  20   continue
      fl(i) = 0.0
c     fl(i) = pfem(i)
      xl(i) = 0.0
      sofl(i)=0.0
      swfl(i)=0.0
      sofl1(i)=0.0
      swfl1(i)=0.0
      sofl2(i)=0.0
      swfl2(i)=0.0
  10  continue
     return
     end
C
c=================================================================
C
     SUBROUTINE LOAD_MODS (ifem,k3,delta)
C
c=================================================================
C  This subroutine interfaces the boast q's to the FEM equations for
c  the full well-block.

     INCLUDE 'PARAMETR.FOR'
     INCLUDE 'PARAMETR.FEM'
     double precision stiff,fl,xl,fl_p(num_nodes*nzp)
          REAL KX,KY,KZ,KROT,KRWT,KRGT,MUOT,MUWT,MUGT,MCFG,MCFG1,MCFGT,
    & MCFGI,MBEO,MBEW,MBEG,
    & qok(num_nodes*nzp), qgk(num_nodes*nzp),
    & qwk(num_nodes*nzp), qsum, qosum, qwsum, qgsum
     REAL muo,muw,mug,kro,krw,krg
     COMMON /BUBBLE/ PBO,VSLOPE,BSLOPE,RSLOPE,PMAXT,IREPRS,
    & RHOSCO,RHOSCG,RHOSCW,MSAT,MPOT,MPWT,MPGT,PBOT(NX,NY,NZ)
     common /perm/ kx(nx,ny,nz),ky(nx,ny,nz),kz(nx,ny,nz)
     COMMON /PRTP/ P(NX,NY,NZ)
     COMMON /SPVT/ SAT(NTE),KROT(NTE),KRWT(NTE),KRGT(NTE),PCOWT(NTE),
    & PCGOT(NTE),POT(NTE),MUOT(NTE),BOT(NTE),BOPT(NTE),RSOT(NTE),RSOPT
    & (NTE),PWT(NTE),MUWT(NTE),BWT(NTE),BWPT(NTE),RSWT(NTE),RSWPT(NTE),
    & PGT(NTE),MUGT(NTE),BGT(NTE),BGPT(NTE),CRT(NTE)
```

```
      COMMON /IQN/ IQN1(NW),IQN2(NW),IQN3(NW)
      COMMON /RATE/ QO(NX,NY,NZ),QW(NX,NY,NZ),QG(NX,NY,NZ)
      COMMON /FEMRATE/ QWWW(NZ),QWE(NZ),QWS(NZ),QWN(NZ),QWT(NZ),QWB(NZ),
     &  QOW(NZ),QOE(NZ),QOS(NZ),QON(NZ),QOT(NZ),QOB(NZ),
     &  QGW(NZ),QGE(NZ),QGS(NZ),QGN(NZ),QGT(NZ),QGB(NZ)
      common /shape/ phif(lmt_nodes),
     & dphd(lmt_nodes,3),coords(lmt_nodes,3)
c######################################################################

      common/glstiff/stiff(num_nodes*nzp,num_nodes*nzp),
     &  fl(num_nodes*nzp),xl(num_nodes*nzp),ipvt(num_nodes*nzp),
     &  sofl(num_nodes*nzp),sofl1(num_nodes*nzp),sofl2(num_nodes*nzp),
     &  sgfl(num_nodes*nzp),sgfl1(num_nodes*nzp),sgfl2(num_nodes*nzp),
     &  swfl(num_nodes*nzp),swfl1(num_nodes*nzp),swfl2(num_nodes*nzp)

      common/stiff0/stiff0(num_nodes*nzp,num_nodes*nzp),
     &  sostiff(num_nodes*nzp,num_nodes*nzp),
     &  sgstiff(num_nodes*nzp,num_nodes*nzp),
     &  swstiff(num_nodes*nzp,num_nodes*nzp)

      common/oldpwg/pold(num_nodes*nzp),swold(num_nodes*nzp),
     &  soold(num_nodes*nzp),  sgold(num_nodes*nzp)

      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)
c######################################################################
c     oil into the well-block
c     There are 96 nodes at the well
c
      if (k3 .eq. 3) then
      pp=pfem(9+num_nodes*k)
      bpt=pbotfem(9+num_nodes*k)

      CALL INTPVT(BPT,RSLOPE,POT,RSOT,MPOT,PP,RSO)
      CALL INTERP(PWT,RSWT,MPWT,PP,RSW)
      CALL INTPVT(BPT,BSLOPE,POT,BOT,MPOT,PP,BOfem)
      CALL INTERP(PWT,BWT,MPWT,PP,BWfem)
      CALL INTERP(PGT,BGT,MPGT,PP,BGfem)
      CALL INTPVT(BPT,VSLOPE,POT,MUOT,MPOT,PP,MUO)
      CALL INTERP(PWT,MUWT,MPWT,PP,MUW)
      CALL INTERP(PGT,MUGT,MPGT,PP,MUG)
      SSO=sofem(num_nodes*3+7)
      SSW=swfem(num_nodes*3+7)
      SSG=sgfem(num_nodes*3+7)
      CALL INTERP(SAT,KRWT,MSAT,SSW,KRW)
      CALL INTERP(SAT,KROT,MSAT,SSO,KRO)
      CALL INTERP(SAT,KRGT,MSAT,SSG,KRG)
```

```
      gasoilr=(krg/mug/bgfem)/(kro/muo/bofem)+rso

      foil=-delta*qo(10,10,k)*5.6146
      fwat=-delta*qw(10,10,k)*5.6146
      fgas=-delta*((qg(10,10,k)
     &     +qo(10,10,k)*rso
     &     +qw(10,10,k)*rsw))
      frho = (foil+fwat+fgas)/32.0

      do ii = 1, 16
        fl(ii+num_nodes*(k-1)) = fl(ii+num_nodes*(k-1))-frho
        fl(ii+num_nodes*k) = fl(ii+num_nodes*k) - frho

        qofem(ii+num_nodes*(k-1)) = qofem(ii+num_nodes*(k-1))-foil
        qofem(ii+num_nodes*k) = qofem(ii+num_nodes*k) - foil
        qwfem(ii+num_nodes*(k-1)) = qwfem(ii+num_nodes*(k-1))-fwat
        qwfem(ii+num_nodes*k) = qwfem(ii+num_nodes*k) - fwat
        qgfem(ii+num_nodes*(k-1)) = qgfem(ii+num_nodes*(k-1))-fgas
        qgfem(ii+num_nodes*k) = qgfem(ii+num_nodes*k) - fgas

      end do
      end if

c#####################################################################
c     outer boundry  layer
c#####################################################################

     do k = 1, 3

c ===== NORTH FLOW

      pp= pfem((num_nodes-11) + num_nodes*k)
      bpt=pbotfem((num_nodes-11)+ num_nodes*k)
      CALL INTPVT(BPT,RSLOPE,POT,RSOT,MPOT,PP,RSO)
      CALL INTERP(PWT,RSWT,MPWT,PP,RSW)
      CALL INTPVT(BPT,BSLOPE,POT,BOT,MPOT,PP,BOfem)
      CALL INTERP(PWT,BWT,MPWT,PP,BWfem)
      CALL INTERP(PGT,BGT,MPGT,PP,BGfem)

     foil = -delta*qon(k)*5.6146
     fwat = -delta*qwn(k)*5.6146
     fgas = -delta*(qgn(k)+rso*qon(k)+rsw*qwn(k))
     frho = (foil+fwat+fgas)/10.0

     fl((num_nodes-13)+num_nodes*(k-1)) =
    &  fl((num_nodes-13)+num_nodes*(k-1))+frho
     fl((num_nodes-13)+num_nodes*k) =
    &  fl((num_nodes-13)+num_nodes*k)+frho
     fl((num_nodes-12)+num_nodes*(k-1)) =
    &  fl((num_nodes-12)+num_nodes*(k-1))+frho
```

```
 fl((num_nodes-12)+num_nodes*k) =
&  fl((num_nodes-12)+num_nodes*k)+frho
 fl((num_nodes-11)+num_nodes*(k-1)) =
&  fl((num_nodes-11)+num_nodes*(k-1))+frho
 fl((num_nodes-11)+num_nodes*k) =
&  fl((num_nodes-11)+num_nodes*k)+frho
 fl((num_nodes-10)+num_nodes*(k-1)) =
&  fl((num_nodes-10)+num_nodes*(k-1))+frho
 fl((num_nodes-10)+num_nodes*k) =
&  fl((num_nodes-10)+num_nodes*k)+frho
 fl((num_nodes-9)+num_nodes*(k-1)) =
&  fl((num_nodes-9)+num_nodes*(k-1))+frho
 fl((num_nodes-9)+num_nodes*k) =
&  fl((num_nodes-9)+num_nodes*k)+frho

 sofem((num_nodes-13)+num_nodes*(k-1)) =
&  sofem((num_nodes-13)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-13)+num_nodes*k) =
&  sofem((num_nodes-13)+num_nodes*k) + foil/10.0
 sofem((num_nodes-12)+num_nodes*(k-1)) =
&  sofem((num_nodes-12)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-12)+num_nodes*k) =
&  sofem((num_nodes-12)+num_nodes*k) + foil/10.0
 sofem((num_nodes-11)+num_nodes*(k-1)) =
&  sofem((num_nodes-11)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-11)+num_nodes*k) =
&  sofem((num_nodes-11)+num_nodes*k) + foil/10.0
 sofem((num_nodes-10)+num_nodes*(k-1)) =
&  sofem((num_nodes-10)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-10)+num_nodes*k) =
&  sofem((num_nodes-10)+num_nodes*k) + foil/10.0
 sofem((num_nodes-9)+num_nodes*(k-1)) =
&  sofem((num_nodes-9)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-9)+num_nodes*k) =
&  sofem((num_nodes-9)+num_nodes*k) + foil/10.0

 swfem((num_nodes-13)+num_nodes*(k-1)) =
&  swfem((num_nodes-13)+num_nodes*(k-1)) + fwat/10.0
 swfem((num_nodes-13)+num_nodes*k) =
&  swfem((num_nodes-13)+num_nodes*k) + fwat/10.0
 swfem((num_nodes-12)+num_nodes*(k-1)) =
&  swfem((num_nodes-12)+num_nodes*(k-1)) + fwat/10.0
 swfem((num_nodes-12)+num_nodes*k) =
&  swfem((num_nodes-12)+num_nodes*k) + fwat/10.0
 swfem((num_nodes-11)+num_nodes*(k-1)) =
&  swfem((num_nodes-11)+num_nodes*(k-1)) + fwat/10.0
 swfem((num_nodes-11)+num_nodes*k) =
&  swfem((num_nodes-11)+num_nodes*k) + fwat/10.0
 swfem((num_nodes-10)+num_nodes*(k-1)) =
&  swfem((num_nodes-10)+num_nodes*(k-1)) + fwat/10.0
```

```
      swfem((num_nodes-10)+num_nodes*k) =
     &   swfem((num_nodes-10)+num_nodes*k) + fwat/10.0
      swfem((num_nodes-9)+num_nodes*(k-1)) =
     &   swfem((num_nodes-9)+num_nodes*(k-1)) + fwat/10.0
      swfem((num_nodes-9)+num_nodes*k) =
     &   swfem((num_nodes-9)+num_nodes*k) + fwat/10.0


c ===== SOUTH FLOW

        pp= pfem(num_nodes-3 + num_nodes*k)
        bpt=pbotfem(num_nodes-3 + num_nodes*k)
        CALL INTPVT(BPT,RSLOPE,POT,RSOT,MPOT,PP,RSO)
        CALL INTERP(PWT,RSWT,MPWT,PP,RSW)
        CALL INTPVT(BPT,BSLOPE,POT,BOT,MPOT,PP,BOfem)
        CALL INTERP(PWT,BWT,MPWT,PP,BWfem)
        CALL INTERP(PGT,BGT,MPGT,PP,BGfem)

       foil = -delta*qos(k)*5.6146
       fwat = -delta*qws(k)*5.6146
       fgas = -delta*(qgs(k)+rso*qos(k)+rsw*qws(k))
       frho = (foil+fwat+fgas)/10.0

       fl((num_nodes-4)+num_nodes*(k-1)) =
     &   fl((num_nodes-4)+num_nodes*(k-1))+frho
       fl((num_nodes-4)+num_nodes*k) =
     &   fl((num_nodes-4)+num_nodes*k)+frho
       fl((num_nodes-3)+num_nodes*(k-1)) =
     &   fl((num_nodes-3)+num_nodes*(k-1))+frho
       fl((num_nodes-3)+num_nodes*k) =
     &   fl((num_nodes-3)+num_nodes*k)+frho
       fl((num_nodes-2)+num_nodes*(k-1)) =
     &   fl((num_nodes-2)+num_nodes*(k-1))+frho
       fl((num_nodes-2)+num_nodes*k) =
     &   fl((num_nodes-2)+num_nodes*k)+frho
       fl((num_nodes-5)+num_nodes*(k-1)) =
     &   fl((num_nodes-5)+num_nodes*(k-1))+frho
       fl((num_nodes-5)+num_nodes*k) =
     &   fl((num_nodes-5)+num_nodes*k)+frho
       fl((num_nodes-1)+num_nodes*(k-1)) =
     &   fl((num_nodes-1)+num_nodes*(k-1))+frho
       fl((num_nodes-1)+num_nodes*k) =
     &   fl((num_nodes-1)+num_nodes*k)+frho

       sofem((num_nodes-4)+num_nodes*(k-1)) =
     &   sofem((num_nodes-4)+num_nodes*(k-1)) + foil/10.0
       sofem((num_nodes-4)+num_nodes*k) =
     &   sofem((num_nodes-4)+num_nodes*k) + foil/10.0
       sofem((num_nodes-3)+num_nodes*(k-1)) =
     &   sofem((num_nodes-3)+num_nodes*(k-1)) + foil/10.0
```

```
      sofem((num_nodes-3)+num_nodes*k) =
     &  sofem((num_nodes-3)+num_nodes*k) + foil/10.0
      sofem((num_nodes-2)+num_nodes*(k-1)) =
     &  sofem((num_nodes-2)+num_nodes*(k-1)) + foil/10.0
      sofem((num_nodes-2)+num_nodes*k) =
     &  sofem((num_nodes-2)+num_nodes*k) + foil/10.0
      sofem((num_nodes-5)+num_nodes*(k-1)) =
     &  sofem((num_nodes-5)+num_nodes*(k-1)) + foil/10.0
      sofem((num_nodes-5)+num_nodes*k) =
     &  sofem((num_nodes-5)+num_nodes*k) + foil/10.0
      sofem((num_nodes-1)+num_nodes*(k-1)) =
     &  sofem((num_nodes-1)+num_nodes*(k-1)) + foil/10.0
      sofem((num_nodes-1)+num_nodes*k) =
     &  sofem((num_nodes-1)+num_nodes*k) + foil/10.0

      swfem((num_nodes-4)+num_nodes*(k-1)) =
     &  swfem((num_nodes-4)+num_nodes*(k-1)) + fwat/10.0
      swfem((num_nodes-4)+num_nodes*k) =
     &  swfem((num_nodes-4)+num_nodes*k) + fwat/10.0
      swfem((num_nodes-3)+num_nodes*(k-1)) =
     &  swfem((num_nodes-3)+num_nodes*(k-1)) + fwat/10.0
      swfem((num_nodes-3)+num_nodes*k) =
     &  swfem((num_nodes-3)+num_nodes*k) + fwat/10.0
      swfem((num_nodes-2)+num_nodes*(k-1)) =
     &  swfem((num_nodes-2)+num_nodes*(k-1)) + fwat/10.0
      swfem((num_nodes-2)+num_nodes*k) =
     &  swfem((num_nodes-2)+num_nodes*k) + fwat/10.0
      swfem((num_nodes-5)+num_nodes*(k-1)) =
     &  swfem((num_nodes-5)+num_nodes*(k-1)) + fwat/10.0
      swfem((num_nodes-5)+num_nodes*k) =
     &  swfem((num_nodes-5)+num_nodes*k) + fwat/10.0
      swfem((num_nodes-1)+num_nodes*(k-1)) =
     &  swfem((num_nodes-1)+num_nodes*(k-1)) + fwat/10.0
      swfem((num_nodes-1)+num_nodes*k) =
     &  swfem((num_nodes-1)+num_nodes*k) + fwat/10.0

c ===== EAST FLOW

        pp= pfem((num_nodes-13) + num_nodes*k)
        bpt=pbotfem((num_nodes-13) + num_nodes*k)
        CALL INTPVT(BPT,RSLOPE,POT,RSOT,MPOT,PP,RSO)
        CALL INTERP(PWT,RSWT,MPWT,PP,RSW)
        CALL INTPVT(BPT,BSLOPE,POT,BOT,MPOT,PP,BOfem)
        CALL INTERP(PWT,BWT,MPWT,PP,BWfem)
        CALL INTERP(PGT,BGT,MPGT,PP,BGfem)

       foil = -delta*qoe(k)*5.6146
       fwat = -delta*qwe(k)*5.6146
       fgas = -delta*(qge(k)+rso*qoe(k)+rsw*qwe(k))
       frho = (foil+fwat+fgas)/10.0
```

```
 fl(num_nodes+num_nodes*(k-1)) =
&  fl(num_nodes+num_nodes*(k-1))+frho
 fl(num_nodes+num_nodes*k) =
&  fl(num_nodes+num_nodes*k)+frho
 fl((num_nodes-15)+num_nodes*(k-1)) =
&  fl((num_nodes-15)+num_nodes*(k-1))+frho
 fl((num_nodes-15)+num_nodes*k) =
&  fl((num_nodes-15)+num_nodes*k)+frho
 fl((num_nodes-14)+num_nodes*(k-1)) =
&  fl((num_nodes-14)+num_nodes*(k-1))+frho
 fl((num_nodes-14)+num_nodes*k) =
&  fl((num_nodes-14)+num_nodes*k)+frho
 fl((num_nodes-13)+num_nodes*(k-1)) =
&  fl((num_nodes-13)+num_nodes*(k-1))+frho
 fl((num_nodes-13)+num_nodes*k) =
&  fl((num_nodes-13)+num_nodes*k)+frho
 fl((num_nodes-1)+num_nodes*(k-1)) =
&  fl((num_nodes-1)+num_nodes*(k-1))+frho
 fl((num_nodes-1)+num_nodes*k) =
&  fl((num_nodes-1)+num_nodes*k)+frho
 fl((num_nodes-0)+num_nodes*(k-1)) =
&  fl((num_nodes-0)+num_nodes*(k-1))+frho
 fl((num_nodes-0)+num_nodes*k) =
&  fl((num_nodes-0)+num_nodes*k)+frho

 sofem((num_nodes-15)+num_nodes*(k-1)) =
&  sofem((num_nodes-15)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-15)+num_nodes*k) =
&  sofem((num_nodes-15)+num_nodes*k) + foil/10.0
 sofem((num_nodes-14)+num_nodes*(k-1)) =
&  sofem((num_nodes-14)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-14)+num_nodes*k) =
&  sofem((num_nodes-14)+num_nodes*k) + foil/10.0
 sofem((num_nodes-13)+num_nodes*(k-1)) =
&  sofem((num_nodes-13)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-13)+num_nodes*k) =
&  sofem((num_nodes-13)+num_nodes*k) + foil/10.0
 sofem((num_nodes-1)+num_nodes*(k-1)) =
&  sofem((num_nodes-1)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-1)+num_nodes*k) =
&  sofem((num_nodes-1)+num_nodes*k) + foil/10.0
 sofem((num_nodes-0)+num_nodes*(k-1)) =
&  sofem((num_nodes-0)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-0)+num_nodes*k) =
&  sofem((num_nodes-0)+num_nodes*k) + foil/10.0

 swfem((num_nodes-15)+num_nodes*(k-1)) =
&  swfem((num_nodes-15)+num_nodes*(k-1)) + fwat/10.0
 swfem((num_nodes-15)+num_nodes*k) =
```

```
     &   swfem((num_nodes-15)+num_nodes*k) + fwat/10.0
         swfem((num_nodes-14)+num_nodes*(k-1)) =
     &   swfem((num_nodes-14)+num_nodes*(k-1)) + fwat/10.0
         swfem((num_nodes-14)+num_nodes*k) =
     &   swfem((num_nodes-14)+num_nodes*k) + fwat/10.0
         swfem((num_nodes-13)+num_nodes*(k-1)) =
     &   swfem((num_nodes-13)+num_nodes*(k-1)) + fwat/10.0
         swfem((num_nodes-13)+num_nodes*k) =
     &   swfem((num_nodes-13)+num_nodes*k) + fwat/10 0
         swfem((num_nodes-1)+num_nodes*(k-1)) =
     &   swfem((num_nodes-1)+num_nodes*(k-1)) + fwat/10.0
         swfem((num_nodes-1)+num_nodes*k) =
     &   swfem((num_nodes-1)+num_nodes*k) + fwat/10.0
         swfem((num_nodes-0)+num_nodes*(k-1)) =
     &   swfem((num_nodes-0)+num_nodes*(k-1)) + fwat/10.0
         swfem((num_nodes-0)+num_nodes*k) =
     &   swfem((num_nodes-0)+num_nodes*k) + fwat/10.0

c ===== WEST FLOW

          pp= pfem((num_nodes-7) + num_nodes*k)
          bpt=pbotfem((num_nodes-7) + num_nodes*k)
          CALL INTPVT(BPT,RSLOPE,POT,RSOT,MPOT,PP,RSO)
          CALL INTERP(PWT,RSWT,MPWT,PP,RSW)
          CALL INTPVT(BPT,BSLOPE,POT,BOT,MPOT,PP,BOfem)
          CALL INTERP(PWT,BWT,MPWT,PP,BWfem)
          CALL INTERP(PGT,BGT,MPGT,PP,BGfem)

         foil = -delta*qow(k)*5.6146
         fwat = -delta*qwww(k)*5.5146
         fgas = -delta*(qgw(k)+rso*qow(k)+rsw*qwww(k))
         frho = (foil+fwat+fgas)/10.0

         fl((num_nodes-9)+num_nodes*(k-1)) =
     &   fl((num_nodes-9)+num_nodes*(k-1))+frho
         fl((num_nodes-9)+num_nodes*k) =
     &   fl((num_nodes-9)+num_nodes*k)+frho
         fl((num_nodes-8)+num_nodes*(k-1)) =
     &   fl((num_nodes-8)+num_nodes*(k-1))+frho
         fl((num_nodes-8)+num_nodes*k) =
     &   fl((num_nodes-8)+num_nodes*k)+frho
         fl((num_nodes-7)+num_nodes*(k-1)) =
     &   fl((num_nodes-7)+num_nodes*(k-1))+frho
         fl((num_nodes-7)+num_nodes*k) =
     &   fl((num_nodes-7)+num_nodes*k)+frho
         fl((num_nodes-6)+num_nodes*(k-1)) =
     &   fl((num_nodes-6)+num_nodes*(k-1))+frho
         fl((num_nodes-6)+num_nodes*k) =
     &   fl((num_nodes-6)+num_nodes*k)+frho
         fl((num_nodes-5)+num_nodes*(k-1)) =
```

```
&   fl((num_nodes-5)+num_nodes*(k-1))+frho
 fl((num_nodes-5)+num_nodes*k) =
&   fl((num_nodes-5)+num_nodes*k)+frho

 sofem((num_nodes-9)+num_nodes*(k-1)) =
&   sofem((num_nodes-9)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-9)+num_nodes*k) =
&   sofem((num_nodes-9)+num_nodes*k) + foil/10.0
 sofem((num_nodes-8)+num_nodes*(k-1)) =
&   sofem((num_nodes-8)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-8)+num_nodes*k) =
&   sofem((num_nodes-8)+num_nodes*k) + foil/10.0
 sofem((num_nodes-7)+num_nodes*(k-1)) =
&   sofem((num_nodes-7)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-7)+num_nodes*k) =
&   sofem((num_nodes-7)+num_nodes*k) + foil/10.0
 sofem((num_nodes-6)+num_nodes*(k-1)) =
&   sofem((num_nodes-6)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-6)+num_nodes*k) =
&   sofem((num_nodes-6)+num_nodes*k) + foil/10.0
 sofem((num_nodes-5)+num_nodes*(k-1)) =
&   sofem((num_nodes-5)+num_nodes*(k-1)) + foil/10.0
 sofem((num_nodes-5)+num_nodes*k) =
&   sofem((num_nodes-5)+num_nodes*k) + foil/10.0

 swfem((num_nodes-9)+num_nodes*(k-1)) =
&   swfem((num_nodes-9)+num_nodes*(k-1)) + fwat/10.0
 swfem((num_nodes-9)+num_nodes*k) =
&   swfem((num_nodes-9)+num_nodes*k) + fwat/10.0
 swfem((num_nodes-8)+num_nodes*(k-1)) =
&   swfem((num_nodes-8)+num_nodes*(k-1)) + fwat/10.0
 swfem((num_nodes-8)+num_nodes*k) =
&   swfem((num_nodes-8)+num_nodes*k) + fwat/10.0
 swfem((num_nodes-7)+num_nodes*(k-1)) =
&   swfem((num_nodes-7)+num_nodes*(k-1)) + fwat/10.0
 swfem((num_nodes-7)+num_nodes*k) =
&   swfem((num_nodes-7)+num_nodes*k) + fwat/10.0
 swfem((num_nodes-6)+num_nodes*(k-1)) =
&   swfem((num_nodes-6)+num_nodes*(k-1)) + fwat/10.0
 swfem((num_nodes-6)+num_nodes*k) =
&   swfem((num_nodes-6)+num_nodes*k) + fwat/10.0
 swfem((num_nodes-5)+num_nodes*(k-1)) =
&   swfem((num_nodes-5)+num_nodes*(k-1)) + fwat/10.0
 swfem((num_nodes-5)+num_nodes*k) =
&   swfem((num_nodes-5)+num_nodes*k) + fwat/10.0

end do
return
end
```

```
c=============================================================
C
      SUBROUTINE CONSTRAIN_SYSTEM
c=============================================================
C   This subroutine constrains the FEM equations for
c   the well-block.
      INCLUDE 'PARAMETR.FEM'
      INCLUDE 'PARAMETR.FOR'
      double precision stiff,fl,xl
      dimension ix(num_nodes)
      COMMON /NUMBER/ II,JJ,KK
c#############################################################

      common/glstiff/stiff(num_nodes*nzp,num_nodes*nzp),
     &  fl(num_nodes*nzp),xl(num_nodes*nzp),ipvt(num_nodes*nzp),
     &  sofl(num_nodes*nzp),sofl1(num_nodes*nzp),sofl2(num_nodes*nzp),
     &  sgfl(num_nodes*nzp),sgfl1(num_nodes*nzp),sgfl2(num_nodes*nzp),
     &  swfl(num_nodes*nzp),swfl1(num_nodes*nzp),swfl2(num_nodes*nzp)

      common/stiff0/stiff0(num_nodes*nzp,num_nodes*nzp),
     &  sostiff(num_nodes*nzp,num_nodes*nzp),
     &  sgstiff(num_nodes*nzp,num_nodes*nzp),
     &  swstiff(num_nodes*nzp,num_nodes*nzp)

      common/oldpwg/pold(num_nodes*nzp),swold(num_nodes*nzp),
     &  soold(num_nodes*nzp),  sgold(num_nodes*nzp)

      common /global/ xyzcds(num_nodes,3),pfem(num_nodes*nzp),
     & sofem(num_nodes*nzp),swfem(num_nodes*nzp),
     & sgfem(num_nodes*nzp),pbotfem(num_nodes*nzp),
     & qofem(num_nodes*nzp),qwfem(num_nodes*nzp),
     & qgfem(num_nodes*nzp),phi(nz)

      common/oilfem/ A(lmt_nodes,lmt_nodes),
     & COIJ(lmt_nodes,lmt_nodes),DO(lmt_nodes),
     & COIL(lmt_nodes,lmt_nodes),QOIL(lmt_nodes)

      common/waterfem/ H(lmt_nodes,lmt_nodes),
     & BWAT(lmt_nodes,lmt_nodes),DW(lmt_nodes),
     & CWIJ(lmt_nodes,lmt_nodes),QWAT(lmt_nodes),
     & CWAT(lmt_nodes,lmt_nodes)

      common/gasfem/ EGAS(lmt_nodes,lmt_nodes),
     & G(lmt_nodes,lmt_nodes),DG(lmt_nodes),
     & WX(lmt_nodes,lmt_nodes),QGAS(lmt_nodes),
     & rsodo(lmt_nodes),rswdw(lmt_nodes),
     & GX(lmt_nodes,lmt_nodes),rswb(lmt_nodes,lmt_nodes) ,
     & cacw(lmt_nodes,lmt_nodes),rsoa(lmt_nodes,lmt_nodes),
     & rswh(lmt_nodes,lmt_nodes) ,
     & fgas(lmt_nodes,lmt_nodes),fwat(lmt_nodes,lmt_nodes)
```

```
c ===== Fix the pressure at the outer boundary on layer 3 (top & bottom)
      do k = 1, 3
        do i = 1, 16
          ix(i+16*(k-1)) = (num_nodes-16) + i + num_nodes*(k-1)
          if (k .eq. 3) then
            ix(i+16*k) = (num_nodes-16) + i + num_nodes*k
          end if
c          write(60,*) ix(i), ix(i+16)
        end do
      end do

c ======= for each node on the outer boundary, subtract from
c its corresponding "fl" the pressure value from BOAST
c
      do i = 1, (num_nodes-32)
        do j = 1, num_nodes*4
          fl(j) = fl(j)-stiff(j,ix(i))*pfem(ix(i))
        end do
        fl(ix(i)) = pfem(ix(i))
      end do

c ======= in the stiffness matrix for each outer boundary
c node replace all nodes on the corresponding row and column
c by zero
      .
      do node = 1, (num_nodes-32)
        do i = 1, num_nodes*4
          do j = 1, num_nodes*4
            stiff(ix(node),j) = 0.0
            stiff(i,ix(node)) = 0.0
          end do
        end do
      end do

c ======= in the stiffness matrix replace the outer boundary
c nodes with "1"
c
      do i = 1, (num_nodes-32)
        stiff(ix(i),ix(i)) = 1.0
      end do

c       write(60,*)' fl(i) on boundary (outer circle):'
c       write(60,*)' after modification of load. '
c       write(60,*)
c       write(60,655)
c655     format(3x, 'nodes 81-96', 3x,'nodes 177-192',
c   &    3x,'nodes 273-288', 3x,'nodes 369-384')
c       write(60,*)
c       do ij = 1, 16
```

```
c         write(60,*)fl(80+ij), fl(176+ij),
c   &          fl(272+ij), fl(368+ij)
c667       format(3x,f14.7,4x,f14.7,5x,f14.7,6x,f14.7)
c       end do

      return
      end

      SUBROUTINE AVEPWG(ifem,ttltime)

      INCLUDE 'PARAMETR.FOR'
      INCLUDE 'PARAMETR.FEM'
      INTEGER indx(lmt_nodes)
      double precision det_jacobian      ! determinant of jacobian
      double precision dn(8,3)          !
      COMMON/MESHPTS/ i1(num_elemts),i2(num_elemts),
     & i3(num_elemts),i4(num_elemts), r1
      common/gauss/gauss_pts(2),gauss_wts(2),num_gauss_pts
       common/shape/phif(lmt_nodes),
     & dphd(lmt_nodes,3),coords(lmt_nodes,3)
c#####################################################################
            pave=0.0
            soave=0.0
            swave=0.0
            sgave=0.0
            volume=0.0
      do 100 k = 1, 3
      do 100 k_el=1,num_elemts
      indx(1) = i1(k_el)+num_nodes*(k-1)
      indx(2) = i2(k_el)+num_nodes*(k-1)
      indx(3) = i3(k_el)+num_nodes*(k-1)
      indx(4) = i4(k_el)+num_nodes*(k-1)
      indx(5) = i1(k_el)+num_nodes*k
      indx(6) = i2(k_el)+num_nodes*k
      indx(7) = i3(k_el)+num_nodes*k
      indx(8) = i4(k_el)+num_nodes*k
       call elmtGEN(ifem,k_el,k)
        do 300 jx_pt = 1,num_gauss_pts
         xi = gauss_pts(jx_pt)
         do 290 jy_pt = 1,num_gauss_pts
          eta = gauss_pts(jy_pt)
          do 280 jz_pt = 1,num_gauss_pts
           zeta = gauss_pts(jz_pt)
          call shape_3d8(xi,eta,zeta)
          call JACOBI_3D8(k_el,det_jacobian,dn)
         gauss_pt_press = 0.0
         gauss_pt_oilsat = 0.0
         gauss_pt_watsat = 0.0
         do 60 i=1,lmt_nodes
          gauss_pt_press = gauss_pt_press + pfem(indx(i))*phif(i)
```

```
      gauss_pt_oilsat = gauss_pt_oilsat + sofem(indx(i))*phif(i)
      gauss_pt_watsat = gauss_pt_watsat + swfem(indx(i))*phif(i)
 60   continue
        pave=pave+gauss_pt_press*det_jacobian
        soave=soave+gauss_pt_oilsat*det_jacobian
        swave=swave+gauss_pt_watsat*det_jacobian
        volume=volume+det_jacobian
280   continue
290   continue
300   continue
100    continue
        pave=pave/volume
        soave=soave/volume
        swave=swave/volume
        sgave=1.0 - soave - swave
     write(30,*)ttltime,soave,swave,sgave
     write(60,*)ttltime,pave
    return
    end

        subroutine eigen_ab(a,b,d)
        parameter(n=60)
        dimension a(n,n),b(n,n),u(n,n),w(n),v(n,n)
        dimension ab(n,n),ainv(n,n)
        dimension wr(n),wi(n)
        dimension d(n),e(n),f(n)
c       u=a
        do i=1,n
        do j=1,n
        u(i,j)=a(i,j)
        end do
        end do
        call svdcmp(u,n,n,n,n,w,v)
c       ainv = v * 1/w * u T
        do i=1,n
        do j=1,n
        ainv(i,j)=0.0d0
        do k=1,n
        ainv(i,j)=ainv(i,j)+v(i,k)*u(j,k)/w(k)
        end do
        end do
        end do
c       ab= ainv *b
        do i=1,n
        do j=1,n
        ab(i,j)=0.0d0
        do k=1,n
        ab(i,j)=ab(i,j)+ainv(i,k)*b(k,j)
        end do
        end do
```

```
end do
call balanc(ab,n,n)
call elmhes(ab,n,n)
call hqr(ab,n,n,wr,wi)
do i=1,n
d(i)=wr(i)
end do
return
end
```

APPENDIX B

INPUT DATA: CASE 1

Case 1
SIMULATION COMPARISON PROBLEM - 1 Layer Model

11 9 1

SO SHALE RESTART DT (SWITCH)
0 0 1 1

GRID BLOCK LENTTH
-1 -1 -1
120.00
146.67
50.00

GRID BLOCK LENGTH MODIFICATION (NONE)
0 0 0 0
CAPROCK BASE DEPTH TO TOP OF SAND
0
2000.00

POROSITY AND PERMEABILITY DISTRIBUTIONS
-1 -1 -1 -1
0.20
50.00
50.00
5 00

POROSITY AND PERMEABILITY MODIFICATION (NONE)
0 0 0 0 0

TRANSMISSIBILITY MODIFICATIONS (NONE)
0 0 0 0

SAT	KRO	KRW	KRG	PCOW	PCGO
0.00	0.00	0.00	0.00	669.19	669.19
0.10	0.00	0.00	0.00	549.19	549 19
0.20	0.00	0.00	0.01	429.19	429.19
0.30	0.02	0.00	0.05	309.19	309.19
0.40	0.06	0.00	0.10	189.19	189.19
0.50	0.15	0.02	0.19	69.19	69.19
0.60	0.32	0.06	0.30	4.00	4.00
0.70	0.59	0.15	0.45	4.00	4.00
0.80	1.00	0.32	0.61	4.00	4.00
0.90	1.00	0.59	0 80	4.00	4.00
1.10	1.00	1.00	1.00	4.00	4.00

PBO	MUSLOPE	BSLOP	RSLOPE	PMAX	IREPRS
1000.0	0.0000460	-0.0000232	0.0	6000.0	0

P	MUO	BO	RSO
15.	16.77	1.034	1.7
500.	8.58	1.068	80.8
1000.	5.34	1.108	172.5
1500.	3.80	1.150	268.8
2000.	2.92	1.193	368.2
2500.	2.37	1.237	469.9
3000.	1.98	1.282	573.6
3500.	1.71	1.327	679.0
4000.	1.50	1.374	785.7
4500.	1.34	1.421	893.7
5000.	1.21	1.468	1002.9
5500.	1.10	1.516	1113.1
6000.	1.01	1.564	1224.2

P	MUW	BW	RSW
0	1.0	1.0	0.
6000.	1.0	1.0	0.

P	MUG	BG	CR	
15.	0.01128	1.053958	0.0000030	0.9974957
500.	0.01191	0.028420	0.0000030	0 9148788
1000	.. 0.01312	0 012958	0.0000030	0.8342582
1500.	0.01500	0.007977	0.0000030	0.7703535
2000.	0.01752	0.005728	0.0000030	0.7375196
2500.	0.02034	0.004580	0.0000030	0.7371771
3000.	0.02316	0.003934	0.0000030	0.7597957
3500.	0.02580	0.003534	0.0000030	0.7963983
4000.	0.02823	0.003267	0.0000030	0 8412924
4500.	0.03043	0.003076	0.0000030	0.8911152
5000.	0.03244	0.002932	0.0000030	0.9438882
5500.	0.03430	0.002820	0.0000030	0.9984262
6000.	0.03601	0.002728	0.0000030	1.0540071

RHOSCO RHOSCW RHOSCG
54.651 62.300 0.056

EQUILIBRIUM PRESSURE INITIALIZATION/CONSTANT SATURATION
0 0
1000.00 920.00 2000.00 1800.00
0.70 0.30 0.00

KSN1	KSM1	KCO1	KTR	KCOF
0	0	0	0	0

NMAX	FACT1	FACT2	TMAX	WORMAX	GORMAX	PAMIN	PAMAX
1000	1.2	.5	1830.0	50.	100000.	0.	10000

KSOL MITR OMEGA TOL TOL1 DSMAX DPMAX

```
  2          3000     1.7          .1        0.          .05          50.

RECURRENT DATA
   1   1   1   1   1   1   1   1   0
 .01  .001  3.  0. 0. 0.
RATES --
   1   0   0
PROD1
   6   5   1   1  -11    1000.0  0.  0.  0.
   3.88  500.00
  0    30.0   0   0   0   0   0   0   0
  1. .01  30.  0.  0.  0.
  0   1   1   1   1   1   1   1   0
  1. .01  3.  0.  0.  0.
  0   150.0   0   0   0   0   0   0   0
  1. .01  30.  0.  0.  0
  0   1   1   1   1   1   1   1   0
  1. .01  3.  0.  0.  0.
  0   180.0   0   0   0   0   0   0   0
  1. .01  30.  0.  0.  0.
  0   1   1   1   1   1   1   1   0
  1. .01  3.  0.  0.  0.
  0   364.0   0   0   0   0   0   0   0
  1. .01  30.  0.  0.  0
  0   1   1   1   1   1   1   1   0
  1. .01  3.  0.  0.  0.
  0   364.0   0   0   0   0   0   0   0
  1. .01  30.  0.  0.  0.
  0   1   1   1   1   1   1   1   0
  1. .01  3.  0.  0.  0.
  0   364.0   0   0   0   0   0   0   0
  1. .01  30.  0.  0.  0.
  0   1   1   1   1   1   1   1   0
  1. .01  3.  0.  0.  0.
  0   364.0   0   0   0   0   0   0   0
  1. .01  30.  0.  0.  0.
  0   1   1   1   1   1   1   1   0
  1. .01  3.  0   0.  0.
  0   728.0   0   0   0   0   0   0   0
  1.  01  30.  0.  0.  0.
  0   1   1   1   1   1   1   1   0
  1. .01  3.  0.  0.  0.
  0   1092.0   0   0   0   0   0   0   0
  1. .01  30.  0.  0   0.
  0   1   1   1   1   1   1   1   0
  1. .01  3.  0.  0.  0.
  0   1092.0   0   0   0   0   0   0   0
  1. .01  30.  0.  0.  0.
  0   1   1   1   1   1   1   1   0
  1. .01  3.  0.  0.  0.
```

```
 0   1092.0   0   0   0   0   0   0   0
1. .01  30.  0.  0.  0.
 0   1   1   1   1   1   1   1   0
1. .01  3.  0.  0.  0.
 0   1456.0   0   0   0   0   0   0   0
1. .01  30.  0.  0.  0.
 0   1   1   1   1   1   1   1   0
1. .01  3.  0.  0.  0.
 0   1820.0   0   0   0   0   0   0   0
1. .01  30.  0.  0.  0.
 0   1   1   1   1   1   1   1   0
1  .01  3.  0.  0.  0.
 0   1820.0   0   0   0   0   0   0   0
1. .01  30.  0.  0.  0.
```

APPENDIX C

INPUT DATA: CASE 2

Case 2
SPE#1 SIMULATION COMPARISON PROBLEM - GAS INJECTION

```
 10        10        3

 SO  SHALE  RESTART DT
 0   0   0   5

GRID BLOCK LENGTH
      -1        -1        0
      1000.
      1000.
      20.       30.       50.

GRID BLOCK LENGTH MODIFICATION (NONE)
      0         0         0         0

CAPROCK BASE DEPTH TO TOP OF SAND
      0
      8325.

POROSITY AND PERMEABILITY DISTRIBUTIONS
      -1        0         0         0
      0.3
      500.      50.       200.
      500.      50.       200.
      50.       50.       19.23

POROSITY AND PERMEABILITY MODIFICATIONS (NONE)
      0         0         0         0         0

TRANSMISSIBILITY MODIFICATIONS (NONE)
      0         0         0         0
      SAT       KRO       KRW       KRG       PCOW      PCGO
      0.00      0.0       0.00      0.0       0.0       0.0
      0.02      0.000     0.0       0.0       0.0       0.0
      0.05      0.000     0.0       0.005     0.0       0.0
      0 12      0.0       0.0       0.025     0.0       0.0
      0.2       0.000     0.0       0.075     0.0       0.0
      0.25      0.000     0.0       0.125     0.0       0.0
      0.3       0.000     0.0       0.19      0.0       0.0
      0.4       0.0001    0.0       0.410     0.0       0.0
      0.45      0.0005    0.0       0.60      0.0       0.0
      0.5       0 001     0.0       0.72      0.0       0.0
      0.6       0.021     0.0       0.87      0.0       0.0
      0.7       0.09      0.0       0.94      0.0       0.0
      0.8       0.350     0.0       0.96      0.0       0.0
      0.85      0.7       0.0       0.98      0.0       0.0
      0.95      0.98      0.0       1.0       0.0       0.0
      1.0       1.0       1.0       1.0       0.0       0.0
```

1.1	1.0	1.0	1.0	0.0	0.0	
PBO		MUSLOPE	BSLOP	RSLOPE	PMAX	IREPRS
4014.7		0.0000460	-0.0000232	0.0	9014.7	1

P	MUO	BO	RSO
14.7	1.0400	1.0620	1.0
264.7	0.975	1.1500	90.5
514.7	0.9100	1.2070	180.0
1014.7	0.8300	1.2950	371.0
2014.7	0.6950	1.4350	636.0
2514.7	0.6410	1.5000	775.0
3014.7	0.5940	1.5650	930.0
4014.7	0.5100	1.6950	1270.0
5014.7	0.4490	1.8270	1618.0
9014.7	0.2030	2.3570	2984.0

P	MUW	BW	RSW
14.7	0.3100	1.0410	0.0
264.7	0.310	1.0403	0.0
514.7	0.3100	1.0395	0.0
1014.7	0.3100	1.0380	.0
2014.7	0.3100	1.0350	.0
2514.7	0.3100	1.0335	.0
3014.7	0.3100	1.0320	.0
4014.7	0.3100	1.0290	0.0
5014.7	0.3100	1.0258	0
9014.7	0.3100	1.0130	.0

P	MUG	BG	CR
14.7	0.008000	0.9358	0.000003
264.7	0.009600	0.067902	0.000003
514.7	0.011200	0.035228	0.000003
1014.7	0.014000	0.017951	.000003
2014.7	0.018000	0.009063	.000003
2514.7	0.020800	0.007726	.000003
3014.7	0.022800	0.006064	.000003
4014.7	0.026800	0.004554	0.000003
5014.7	0.030900	0.003644	.000003
9014.7	0.047000	0.002167	.000003

RHOSCO	RHOSCW	RHOSCG
46.244	62.200	0.065

EQUILIBRIUM PRESSURE INITIALIZATION / CONSTANT SATURATION

0	0			
4800.00	0.00	8450.00	3800.00	
0.88	0.12	0.00		
KSN1	KSM1	KCO1	KTR	KCOF
0	0	0	0	0

NMAX	FACT1	FACT2	TMAX	WORMAX	GORMAX	PAMIN	PAMAX
1000	1.2	.5	2000	50.	20000.	1000.	10000

KSOL	MITR	OMEGA	TOL	TOL1	DSMAX	DPMAX

```
2          3100          1.7      .1          0.00          .05          50.

RECURRENT DATA
        1         1         1         1         1         1         1         1         0
        .0100     .001      3.        0.        0.        0.
RATES
        2         0         0
IN WE
        1         1         1         1         3         0.0       0.0  -100000.       0.
        20.6      0.0
PR WE
        10        10        3         1         1    20000.       .0        0.0       .0
        20.60     3000.0
        0         360       0         0         0         0         0         0         0
        1.        .01       5.        0.        0.        0.
        0         1         1         1         1         1         1         1         0
        1.        .01       5.        0.        0.        0.
        0         700       0         0         0         0         0         0         0
        1.        .01       5.        0.        0.        0.
        1         1         1         1         1         1         1         1         0
        .1        .001      3.        0.        0.        0.
RATES
        2         0         0
IN WE
        1         1         1         1         3         0.0       0.0   100000.       0.
        20.6      0.0
PR WE
        10        10        3         1    -11  0.        .0        0.0       .0
        20.60     1500.0
        0         200       0         0         0         0         0         0         0
        1         .01       5.        0.        0.        0.
        1         1         1         1         1         1         1         1         0
        .1        .001      3.        0.        0.        0.
RATES
        2         0         0
IN WE
        1         1         1         1         3         0.0       0.0  -100000.       0.
        20.6      0.0
PR WE
        10        10        3         1    -11  0.        .0        0.0       .0
        20.60     1000.0
        0         330       0         0         0         0         0         0         0
        1.        .01       5.        0.        0.        0.
        0         1         1         1         1         1         1         1         0
        1.        .01       5.        0.        0.        0.
        0         360       0         0         0         0         0         0         0
        1.        .01       5.        0.        0.        0.
        0         1         1         1         1         1         1         1         0
        1.        .01       5.        0.        0.        0.
        0         970       0         0         0         0         0         0         0
```

1.	.01	5.	0.	0.	0.			
0	1	1	1	1	1	1	1	0
1.	.01	5.	0.	0.	0.			
0	360	0	0	0	0	0	0	0
1.	.01	10.	0.	0.	0.			
0	1	1	1	1	1	1	1	0
1.	.01	3.	0.	0.	0.			
0	730	0	0	0	0	0	0	0
1.	.01	10.	0.	0.	0.			
0	1	1	1	1	1	1	1	0
1.	.01	3.	0.	0.	0.			
0	1830	0	0	0	0	0	0	0
1.	.01	10.	0.	0.	0.			
0	1	1	1	1	1	1	1	0
1.	.01	3.	0.	0	0.			

VITA

Husam M. Yaghi was born in Amman, Jordan, on January 1st, 1961. After graduating from a top scientific stream high school (King Hussein College) in 1979, he pursued educational and vocational interests in the United States of America. In 1984, he received a bachelor of science degree in Electrical Engineering and in 1986 a master of science degree in Computer Science from Southern University. He worked on contracts for IBM, Bell Communications Research, U.S. Department of Defense, among others. He served for ten years as Assistant Professor in the Electrical Engineering Department at Southern University. He served as Director of the Engineering Computing Services, Campus Manager for Telecommunications, and a member of numerous computer and telecommunications task forces locally and nationally. Currently, he serves as Chief Trainer for telecommunications at Vinnell Arabia and Manager of the Networks Department at the Saudi Arabian National Guard. Yaghi has over twenty-five publications in his areas of expertise. He is better known among friends and collegues as "Sam". Currently, he is a candidate for the doctor of philosophy degree in Computer Science.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Husam M. Yaghi

Major Field: Computer Science

Title of Dissertation: Petroleum Reservoir Simulation using 3-D Finite Element Method with Parallel Implementation

Approved:

John M. Tyler
Major Professor and Chairman

Lynn W. Jelinski
Dean of the Graduate School

EXAMINING COMMITTEE:

J. Liu

Adam T Bourgoyne Jr

R. F. O'Connell

Date of Examination:

May 10, 1999