

**DEVELOPMENT OF A 2-D BLACK-OIL RESERVOIR SIMULATOR USING
A UNIQUE GRID-BLOCK SYSTEM**

A Thesis

by

EMELINE E. CHONG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2004

Major Subject: Petroleum Engineering

**DEVELOPMENT OF A 2-D BLACK-OIL RESERVOIR SIMULATOR USING
A UNIQUE GRID-BLOCK SYSTEM**

A Thesis

by

EMELINE E. CHONG

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

David S. Schechter
(Co-Chair of Committee)

Duane A. McVay
(Co-Chair of Committee)

Wayne M. Ahr
(Member)

Stephen A. Holditch
(Head of Department)

December 2004

Major Subject: Petroleum Engineering

ABSTRACT

Development of a 2-D Black-Oil Reservoir Simulator Using
a Unique Grid-Block System.

(December 2004)

Emeline E. Chong, B.S., Malaysia University of Technology

Co-Chairs of Advisory Committee: Dr. David. S. Schechter
Dr. Duane A. McVay

The grid orientation effect is a long-standing problem plaguing reservoir simulators that employ finite difference schemes. A rotation of the computational grids yields a substantially different solution under certain circumstances. For example, in a five-spot pattern, the predicted recovery, water cut performance and the locations of the fronts depend on the type of grid system used. A Cartesian grid with one axis parallel to the line joining an injector and producer gives a solution significantly different from a grid that has the axes oriented at 45° to this line.

This study develops a unique grid-block assignment where rectangular grid blocks are interspersed with octagonal grid blocks. This grid block system is called the Hybrid Grid Block (HGB) system. The objective of this study is to evaluate the grid orientation effect of the HGB grid to see whether it is an improvement over the conventional Cartesian grid system.

In HGB, flow can progress in four directions in the octagonal grid blocks and two in the square grid blocks. The increase in the number of flow directions in the octagonal grid blocks is expected to reduce the grid orientation effect in the model. Hence, this study also evaluates the grid orientation effect of the HGB and compares it with the Cartesian grid system.

To test the viability of HGB, a general purpose finite difference IMPES-formulated two-dimensional black oil simulator was developed in this study, while retaining the familiar finite-difference discretization of the flow equations. Several

simulation cases were conducted to compare HGB and conventional grid block systems. Comparisons with commercial simulator are also made.

Despite the fact that the reservoir is isotropic and homogeneous, grid orientation effect was still observed when rectangular Cartesian grid models are run at mobility ratio, $M = 1.0$. Grid refinement can help to reduce the grid orientation effect in rectangular Cartesian grid models when there are favorable mobility ratios, i.e. $M = 1.0$ or less.

However, at an unfavorable mobility ratio of $M = 10.0$, it is found that neither parallel nor diagonal orientation can be used reliably for the displacement problems run in this study. This is because as the number of grid blocks is increased, the performance of diagonal and parallel models actually diverges for the grid spacings investigated here.

On the other hand, HGB grid is able to reduce the grid orientation effect even for unfavorable mobility ratio displacement problems (up to $M = 50.0$), with maximum relative difference in pore volume recovered of 6% between parallel and diagonal HGB grid models for all the cases run in this study.

Comparisons between the conventional Cartesian and HGB grid show that the HGB grid is more effective in reducing the grid orientation effect than the Cartesian grid. The HGB grid performs better by consistently giving a smaller relative difference between HGB parallel grid and HGB diagonal grid in pore volume recovered (6.0, 4.5, 3.3, and 2.2%) compared to the relative difference between Cartesian parallel grid and Cartesian diagonal grid in pore volume recovered (17.0, 13.0, 9.3, 7.9%) at similar averaged area per grid block for all the four comparison cases studied.

DEDICATION

To all my family members for their love, care and support throughout the years

ACKNOWLEDGMENTS

First of all, a heartfelt thanks to my advisor, Dr. David S. Schechter for giving me all the opportunities to study this research topic as well as his guidance, support and understanding throughout my time here as a graduate student.

I would also like to thank Dr. Duane A. McVay, Dr. Wayne M. Ahr, and Dr. Thomas A. Blasingame for taking the time to serve on my committee and for their helpful suggestions.

Thanks also to Dr. Erwin Putra for his advice and valuable suggestions. In addition, I am very grateful for the kind assistance and suggestions by Mr. Zuher Syihab during the course of my work.

Special thanks to fellow graduate students at the Harold Vance Department of Petroleum Engineering, especially members of the Naturally Fractured Reservoirs Group and all my friends here at Texas A&M University for enriching my life in countless ways.

This work was supported by the Department of Energy. This support is gratefully acknowledged.

Finally, I would like to thank my entire family for their prayers, love, support and encouragement – without whom the completion of this project would not have been possible.

TABLE OF CONTENTS

	Page
ABSTRACT.....	iii
DEDICATION.....	v
ACKNOWLEDGMENTS	vi
TABLE OF CONTENTS.....	vii
LIST OF FIGURES	ix
LIST OF TABLES.....	xiii
 CHAPTER	
I INTRODUCTION	1
1.1 Overview of Gridding Techniques.....	1
1.2 Literature Review on Grid Orientation Effect	2
1.3 Grid Orientation Effect	5
1.4 Research Methodology	6
II FINITE DIFFERENCE FORMULATION: IMPES PROCEDURE.....	7
2.1 Conservation of Mass Equations	7
2.2 IMPES Method	10
2.3 IMPES Flow Equations for Three-Phase Flow.....	11
III PROGRAM CHARACTERISTICS AND PROPERTIES	20
3.1 Algorithm of VB Code.....	20
3.1.1 Initialization Data.....	21
3.1.2 Averaging of Flow Equation Terms	21
3.1.3 Boundary Conditions	23
3.1.4 Well Model	24
3.1.4.1 Peaceman’s Well Model	25
3.1.4.2 Well Constraints.....	27
3.1.5 Time Step Control.....	29
3.1.6 Solution Method – Linear Solver.....	29
3.2 Spatial Discretization of Cartesian Grid System	31
3.3 Implementation of Hybrid Grid-Block (HGB) System	32
3.3.1 Grid Block Generation.....	33
3.3.2 Transmissibility Calculations.....	34
3.3.3 Grid Numbering and Structure of Matrix Forms	36

CHAPTER	Page
3.3.4 Palagi's Well Model	39
IV GRID ORIENTATION EFFECT	42
V PROGRAM VALIDATION AND PERFORMANCE OF HGB MODEL	53
5.1 Program Validation.....	53
5.2 Use of HGB Grid to Reduce Grid Orientation Effect.....	59
5.2.1 HGB Sensitivity	65
VI CONCLUSIONS.....	71
REFERENCES	74
APPENDIX A.....	79
APPENDIX B	91
VITA.....	139

LIST OF FIGURES

FIGURE		Page
1.1	Flow paths for parallel and diagonal flow in a Cartesian grid.....	5
2.1	1-D model with three uniform grid blocks	7
2.2	Flow chart showing how IMPES can be implemented in a computer program.....	19
3.1	Flow chart of the Sim2D code	22
3.2	2-D flow domain with a well	24
3.3	Algorithm for the time step cutback loop	30
3.4	1-D, block-centered, finite difference grid	32
3.5	Grid numbering for the Cartesian grid system.....	32
3.6	HGB grid model.....	33
3.7	Example of transmissibility calculations in HGB.....	34
3.8	Calculations in eight directions for a central octagonal block.....	35
3.9	Ordering #1: 2-D grid block ordering.....	37
3.10	Locations of matrix elements in Ordering #1	37
3.11	Ordering #2: 2-D grid block ordering.....	38
3.12	Locations of matrix elements in Ordering #2	38
3.13	Ordering #3: Re-ordering of grid blocks to reduce band width	39
3.14	Locations of matrix elements in Ordering #3	40
3.15	Well model for a polygon	41

FIGURE	Page
4.1 Parallel and diagonal orientation for simulations of waterflooding in five-spot symmetry elements.....	42
4.2 Porosity modifications.....	44
4.3 Permeability modifications.....	45
4.4 Well model modifications.....	45
4.5 Predicted performance at $M=0.5$ for parallel (8x8) and diagonal (6x6) grid blocks	47
4.6 Predicted performance at $M=0.5$ for different number of diagonal grid blocks.....	47
4.7 Predicted performance at $M=0.5$ for different number of parallel grid blocks.....	48
4.8 Predicted performance at $M=0.5$ for parallel (29x29) and diagonal (21x21) grid blocks	48
4.9 Predicted performance at $M=1.0$ for different number of diagonal grid blocks.....	49
4.10 Predicted performance at $M=1.0$ for different number of parallel grid blocks.....	49
4.11 Predicted performance at $M = 1.0$ for parallel (57x57) and diagonal (41x41) grid blocks	50
4.12 Predicted performance at $M = 10.0$ for different number of diagonal grid blocks.....	51
4.13 Predicted performance at $M = 10.0$ for different number of parallel grid blocks	51
4.14 Saturation distribution map for (a) diagonal model, and (b) parallel model at $PV_{inj}=1.0$ for $M = 10.0$	52
5.1 Relative permeability curve.....	54
5.2 Comparison of Sim2D oil and water rates and watercut with ECL100 showing good agreement between the two simulators.....	55

FIGURE	Page
5.3 Comparison of Sim2D well bottomhole pressure at producer with ECL™ 100.....	56
5.4 Comparison of Sim2D well block pressure at producer with ECL™ 100.....	56
5.5 Comparison of Sim2D well block oil saturation at producer with ECL™ 100.....	57
5.6 Comparison of Sim2D well bottomhole pressure at injector with ECL™ 100.....	57
5.7 Comparison of Sim2D well block pressure at injector with ECL™ 100.....	58
5.8 Comparison of Sim2D well block oil saturation at injector with ECL™ 100.....	58
5.9 (a) Parallel and (b) diagonal grid orientation in HGB grid.....	61
5.10 Porosity, permeability and well model modifications	61
5.11 Influence of mobility ratios on the predicted performance of HGB grid	62
5.12 Saturation distribution map for parallel HGB grid as shown in Fig. 5.9(a) at various mobility ratios	63
5.13 Saturation distribution map for diagonal HGB grid as shown in Fig. 5.9(b) at various mobility ratios.....	64
5.14 Comparison between HGB grid (50 and 98 grid blocks) and Cartesian grid (49 and 100 grid blocks) at $M = 0.5$	68
5.15 Comparison between HGB grid (98 and 200 grid blocks) and Cartesian grid (100 and 196 grid blocks) at $M = 0.5$	68
5.16 Comparison between HGB grid (200 and 392 grid blocks) and Cartesian grid (196 and 400 grid blocks) at $M = 0.5$	69
5.17 Comparison between HGB grid (392 and 800 grid blocks) and Cartesian grid (400 and 784 grid blocks) at M	

	Page
$= 0.5$	69
5.18 Effect of grid spacing on parallel HGB for $M = 0.5$	70
5.19 Effect of grid spacing on diagonal HGB for $M = 0.5$	70

LIST OF TABLES

TABLE		Page
3.1	Averaging of parameters.....	23
4.1	Data used for five-spot pattern simulations	43
4.2	Grid sizes used in Cartesian grid models.....	43
5.1	Reservoir data	54
5.2	PVT data	55
5.3	Data used for HGB pattern simulations.....	59
5.4	2-Phase PVT data (for $M=0.5$).....	60
5.5	Relative difference between parallel and diagonal models of various mobility ratios for HGB grid models in Fig. 5.10.....	62
5.6	Averaged area per grid block for the HGB grid	66
5.7	Averaged area per grid block for the Cartesian grid.....	66
5.8	Relative difference between parallel and diagonal grid for both HGB and Cartesian grid at $M = 0.5$	67

CHAPTER I

INTRODUCTION

1.1 Overview of Gridding Techniques

Most commonly used grids are constructed by aligning the grid block along orthogonal coordinate directions, and then distorting the grid, to fit major reservoir features. It is generally believed that heterogeneous reservoirs could also be represented if grids are made sufficiently small. Even though the Cartesian grids have been widely used, it is not always suitable for the simulation of complex reservoirs. Some shortcomings of Cartesian grids include its inflexibility in the description of faults, pinch outs and discontinuities in reservoirs, and the influence of grid orientation on the results.

In principle, if extremely fine grids could be created it would be possible to represent heterogeneous reservoirs easily. However, the number of grids in a model is practically limited by computer capacity and CPU time. In order to solve this problem, the concept of local grid refinement has been introduced. Local grid refinement involves using a fine grid inside a coarse-based grid. This is usually done for regions with large pressure changes near the wellbore, in areas of wide variation in saturation, in regions of interest which require finer resolution, and in highly heterogeneous regions. This might reduce the computation time but it should yield results which are very similar to a fine-based grid in accuracy. Nevertheless, the banded structure of the matrix is lost so matrix-solving procedure may be less efficient. For example, to model radial flow near a well, hybrid local grid refinement was proposed by Pedrosa and Aziz¹. Orthogonal curvilinear

This thesis follows the style and format of the journal of *SPE Reservoir Evaluation and Engineering*.

grids are used in the well region and Cartesian grids are used in the rest of the reservoir. Different types of locally refined grids have been presented throughout the literature.

Reservoir simulations are normally being performed on rectangular Cartesian grid, radial grid was developed later to simulate flow near the wellbore. Local grid refinement was developed to achieve better accuracy in high flow regions¹⁻². Development of corner-point geometry grid³⁻⁴ enables the use of non-rectangular grid blocks. This provides the ability to model faults and other complex geological features. Until then, all grids were structured, where the neighbors of a grid block could be easily identified from their i,j,k indices.

However, in the last decade, unstructured grids⁵⁻¹⁰ were introduced. In unstructured grids, the connections between grid blocks are flexible, and a connection list is used to keep track of the connected grid blocks. More and more reservoir simulators have flexible grid capabilities already available or in development. More studies should be done to determine whether these techniques are reliable and accurate, and whether they can allow a significant computer time saving during a reservoir simulation run.

1.2 Literature Review on Grid Orientation Effect

Several methods have been proposed to reduce the grid orientation effect throughout the years. The literature can be divided into several major groups in terms of the approach taken to reduce the grid orientation effect.

Grid orientation effect in reservoir simulation caused by conventional rectangular 5-point discretization scheme was reported by Todd et al.¹¹ The orientation of the grid relative to the lines of flow influenced results from the scheme involving five-point differencing and single-point upstream weighting. They attributed the problem to single-point upstream weighting. They proposed the use of two-point upstream mobility weighting in replace of the generally used single-point approximation. They reported a reduction of both numerical dispersion of flood fronts and the sensitivity of predicted areal displacement performance to grid orientation. Holloway et al.¹² presented an

approach to reduce the grid orientation effect by modifying phase transmissibilities and the two-point upstream weighting method proposed by Todd et al.¹¹ that permitted diagonal flow, but their modifications only resulted in marginal improvement over the original two-point weighting. Meanwhile, a generalization of upstream weighting was proposed by Frauenthal et al.¹³, which involves using a weighting parameter between the two mobilities instead of the simple single-point weighting. The main attraction of these techniques is that they can be easily implemented into existing computer codes and do not add significantly to computational time. However, based on the studies done by Vinsome and Au¹⁴, they concluded that in an extreme case of unfavorable mobility ratio, the upstream formulation predicts a pressure drop across a shock front that is much smaller than it is supposed to be, and vice versa in the case of favorable mobility ratio.

The second group of the literature developed around the method of using a nine-point finite difference discretization scheme, which was initially proposed by Yanosik and McCracken¹⁵. This scheme is based on adding diagonal transmissibilities in the areal (X-Y) direction in order to reduce grid orientation effects when the flow is not aligned with the grid. They introduced a “weighting factor”, which were four and one for the diagonal and parallel grids respectively. Various forms of nine-point schemes were also introduced by subsequent authors¹⁵⁻²¹. Ko and Au¹⁶ concluded that the nine-point scheme proposed by Yanosik and McCracken could not solve the problem of grid orientation for all mobility ratios since the weighting factor used in this method is a function of mobility ratio itself. In addition, as the nine-point scheme is a weighted-interpolation between the two five-point grids with a common center point and its diagonal transmissibilities, it hence lacks physical justification.

In single-point upstream mobility weighting, the mobility term is discretized using first order scheme. It is generally believed that the grid orientation effect is partly caused by numerical dispersion in low order techniques such as this. Also, truncation error manifests itself as a numerical dispersion which will cause smearing of the flood front. Coarser grids will have larger truncation errors and more dispersion. On the contrary, finer grids will have smaller truncation error and less dispersion. However, as stated by

Brand et al.²², "...in general the GOE (Grid Orientation Effect) cannot be overcome with grid refinement....When the grid is refined, the solutions still depend on the size and orientation of the underlying grid, as long as numerical diffusion dominates over physical dispersion and diffusion."

The third group concerns mainly with the numerical implications²³⁻²⁶ of the finite difference solutions - using a higher-order finite difference methods, or generally known as the high-order techniques (HOT). For example, Chen et al.²³, Pinto and Correa²⁴ and Wolcott et al.²⁵ proposed using the Total Variation Diminishing (TVD) methods. Wolcott et al.²⁵ used a combination of nine-point scheme and the third order Taylor's series expansion TVD scheme. The authors reported that this method was able to reduce numerical dispersion and produce sharper saturation fronts.

A type of uniform triangular grids was also introduced in the early 1980s²⁷. This method requires the use of the point-distributed grid system and the grid generation is more complicated than the conventional grid system. The advantage of this grid is that the grid boundaries are not aligned in one particular direction or the other. On the other hand, Pruess and Bodvarsson²⁶ proposed the use of a seven-point discretization scheme, which is essentially a structured and uniform hexagonal grid-block model. They investigated steam injection problems with relatively coarse grids and concluded that the hexagonal grid can reduce the grid orientation effect. The use of hexagonal grid was further supported by Heinemann et al.²⁸ in their PEBI (Perpendicular-Bisector) grid model. They also reported the unrealistic saturation front produced by the hexagonal grid and by the Cartesian grid with the nine-point formulation for $M = 50$.

The current trend includes the development of flexible gridding to alleviate the problem associated with grid orientation effect resulted from using rectangular Cartesian grid. Even so, the generation and construction of unstructured grids are not as simple as Cartesian grids. For example, the construction of an unstructured grid for a reservoir is feasible only if it is done by a numerical grid generation procedure.

1.3 Grid Orientation Effect

Finite difference solutions of 2D frontal displacement problems can be strongly influenced by the orientation of the underlying grid. In multidimensional models, numerical dispersion leads to a phenomenon where calculated performance is influenced by the orientation of the grid relative to the locations of injection and production wells. This is called the grid orientation effect. The grid orientation effect has been found to be particularly pronounced in simulations where the displacing phase is much more mobile than the displaced phase.

Fig. 1.1 illustrates the problem. It is a sketch of part of the Cartesian grid system of a model for simulating water flooding in an oil reservoir. This part of the model contains one production well and two injection wells. In the simulator, water from Well A will move in a direct path to the producer. However, water from Well B will follow a zig-zag path to the producer. Not only is the flow path from Well B longer, but water from Well B will sweep the reservoir “more efficiently” than water from Well A. However, if the grid is rotated 45°, the performances calculated for the two wells would be reversed.

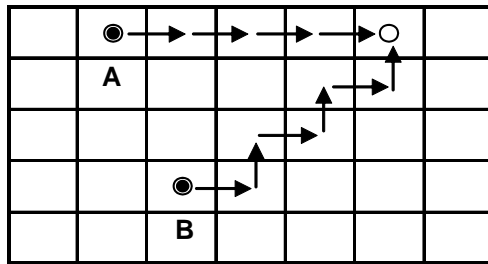


Fig. 1.1 - Flow paths for parallel and diagonal flow in a Cartesian grid (after Mattax and Dalton)¹¹

To complicate matters, grid orientation may distort and affect the accuracy of calculated pressures and saturations²⁹. Thus, the grid orientation effect has become one of the important factors in evaluating different types of grid.

In general, neither parallel nor diagonal orientation can be used reliably for displacements at highly unfavorable mobility ratios. Numerous attempts to eliminate the grid orientation effect in finite difference simulators have been made, and the latest methods being attempted is the use of flexible or unstructured gridding methods.

1.4 Research Methodology

This study presents a novel approach to reduce the effect of grid orientation on computed numerical results in finite difference reservoir simulation. This method involves using a unique grid-block assignment where rectangular grid blocks are interspersed with octagonal grid blocks. The boundaries are then populated with triangular grid blocks. Thus, the entire domain will consist of different structured grid block systems called Hybrid Grid Block (HGB) system. In HGB, flow can progress to four different directions in the octagonal grid blocks and two in the rectangles. This increase in flow directions is expected to reduce the grid orientation effect in the model. As a structured grid system, HGB retains the familiar finite-difference discretization of the flow equations.

To test the viability of this grid system, a general purpose IMPES (Implicit Pressure Explicit Saturation)-formulated 2-D black oil simulator with HGB system was developed using the Visual BASIC programming language. The simulator developed is named Sim2D. Furthermore, comparative evaluations are made by comparing several simulation cases between HGB and conventional grid block systems. This innovative grid block assignment will help to reduce the grid orientation effect.

Chapter II consists of the derivation of material balance equations, and the final IMPES flow equation that is applied in the coding. In Chapter III, the structure and algorithm of the reservoir simulator will be discussed. Chapter IV discusses and analyzes the results of the developed simulator. This will be followed by the conclusions of this study.

CHAPTER II

FINITE DIFFERENCE FORMULATION: IMPES PROCEDURE

2.1 Conservation of Mass Equations

The basic mass conservation laws of reservoir simulation are the conservation of mass, energy and momentum. Mass balance in a grid block is achieved by equating the accumulation of mass in the block with the difference between the mass leaving the block and the mass entering the block. Many derivations of the oil, water, and gas fluid flow equations exist abundantly in the literature³⁰⁻³³. Therefore, only a brief discussion will be presented here.

Considering the grid block i in a 1-D model with three uniform grid blocks size in **Fig. 2.1** below:

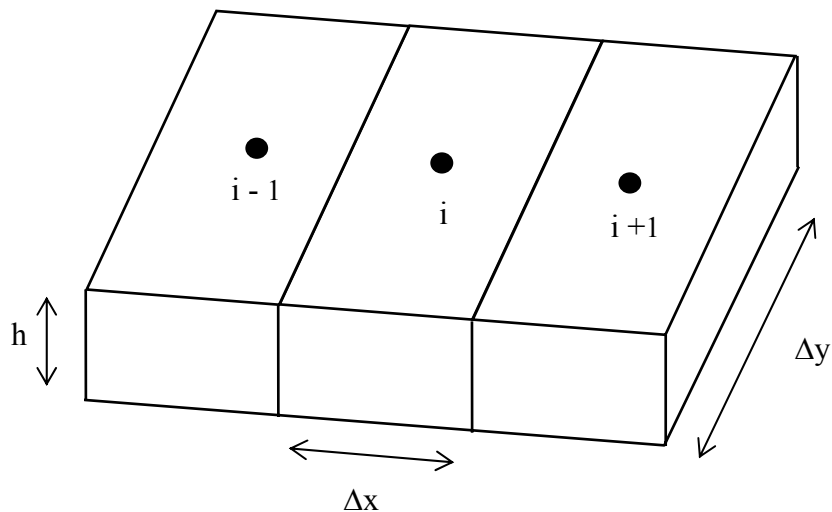


Fig. 2.1 – 1-D model with three uniform grid blocks

Net flow rate in (scf/D) = Rate of Accumulation (scf/D),.....(2.1)

The pore volume of grid block i is:

$$V_p = \phi \Delta x \Delta y h ,(2.2)$$

The oil in place (OIP) can be calculated as:

$$OIP = \frac{V_p S_o}{B_o} ,(2.3)$$

Net flow rate in = $q_{i-1} + q_{i+1}$,.....(2.4)

Rate of accumulation of oil during the time step:

$$= \frac{1}{\Delta t} \left[\left(\frac{V_p S_o}{B_o} \right)^{n+1} - \left(\frac{V_p S_o}{B_o} \right)^n \right] ,(2.5)$$

Our material balance equation can now be stated as:

$$q_{i-1} + q_{i+1} = \frac{1}{\Delta t} \left[\left(\frac{V_p S_o}{B_o} \right)^{n+1} - \left(\frac{V_p S_o}{B_o} \right)^n \right] ,(2.6)$$

Using Darcy's Law to determine flow rate between adjacent grid blocks, flow from the left, from grid block $i-1$ to i :

$$q_{i-1} = \frac{u_{i-1} A}{B_o} ,(2.7)$$

Using field units, we can rewrite q_{i-1} as:

$$q_{i-1} = \left(\frac{0.00633 K K_{ro}}{\mu_o B_o} \right) \left(\frac{P_{i-1} - P_i}{\Delta x} \right) (\Delta y t) ,(2.8)$$

To simplify the notations, we can rewrite this equation into 3 parts:

$$q_{i-1} = \left(\frac{0.00633kh\Delta y}{\Delta x} \right) \left(\frac{kr}{B_o} \right) (P_{i-1} - P_i), \dots \dots \dots (2.9)$$

The first term is called the “transmissibility”:

$$T_{i-1/2} = \frac{0.00633kh\Delta y}{\Delta x}, \dots \dots \dots (2.10)$$

The subscript $i-1/2$ denoted that this term applies between grid block i and $i-1$. It is a directional notation and can be replaced with N(orth), S(outh), E(ast) or W(est), or any other notations, as long as it is consistent.

The second term is called “mobility”. It is dependent on the phase of interest and its value changes with time. Mobility is defined as:

$$\left(\frac{kr}{B\mu} \right)_{\lambda i-1/2}, \dots \dots \dots (2.11)$$

where, λ is the phase of interest – in this case, oil.

Our material balance equation now can be written as:

$$\left(\frac{kr}{B\mu} \right)_{oi+1/2} T_{i+1/2} (P_{i+1} - P_i) + \left(\frac{kr}{B\mu} \right)_{oi-1/2} T_{i-1/2} (P_{i-1} - P_i) = \frac{1}{\Delta t} \left[\left(\frac{V_p S_o}{B_o} \right)^{n+1} - \left(\frac{V_p S_o}{B_o} \right)^n \right], \dots \dots \dots (2.12)$$

To further simplify this equation, we can group another term, called the “oil symmetrical flow coefficient” as follows:

$$a_{oi-1/2} = \left(\frac{kr}{B\mu} \right)_{oi-1/2} T_{i-1/2}, \dots \dots \dots (2.13)$$

The 1-D finite difference equation is:

$$a_{oi+1/2}(P_{i+1} - P_i) + a_{oi-1/2}(P_{i-1} - P_i) = \frac{1}{\Delta t} \left[\left(\frac{V_p S_o}{B_o} \right)^{n+1} - \left(\frac{V_p S_o}{B_o} \right)^n \right], \dots \dots \dots (2.14)$$

Similar derivations can be applied to a 2-D or 3-D model.

2.2 IMPES Method

A simple procedure to solve the three-phase reservoir simulation problems is called the IMPES Method. IMPES stands for “Implicit Pressure Explicit Saturation”. Contrary to the fully implicit method where the main variables are calculated at the same time (ie. all primary variables at the new time level are determined simultaneously), the IMPES method solves for pressure at the new time level using saturations at the old time level, then uses the pressures at the new time level to explicitly calculate saturations at the new time level. However, IMPES becomes unstable for large time steps. Using Neumann stability analysis, the explicit formulation has the following stability requirement³⁴:

$$\Delta t \leq \frac{1}{2} \left(\frac{\phi \mu c}{k} \right) \Delta x^2, \dots \dots \dots (2.15)$$

where,

- Δt = incremental time step
- ϕ = porosity
- μ = viscosity
- c = compressibility
- k = absolute permeability
- Δx = grid size

This requirement has the consequence that the time step is limited by both the grid size and properties of the rock and fluid. IMPES is widely used for field scale reservoir simulation as it is simple to implement. It can also be fast and accurate for many reservoir problems as long as the time steps are kept small.

2.3 IMPES Flow Equations for Three-Phase Flow

The three finite difference equations for oil, water and gas for a three-phase system will be presented below. For simplicity, we shall assume that porosity is constant and not a function of pressure. Here, the solution gas, capillary pressure, and gravity will be considered. However, in the development of the simulator Sim2D, these three terms will be ignored.

Starting from the three finite difference equations for oil, water and gas respectively:

Oil :

$$\Delta a_o \Delta P_o - \Delta a_o \Delta \left(\frac{\rho_o z}{144} \right) = \frac{1}{\Delta t} \left[\left(\frac{V_p S_o}{B_o} \right)^{n+1} - \left(\frac{V_p S_o}{B_o} \right)^n \right] + q_{osc}, \dots \dots \dots (2.16)$$

Water :

$$\Delta a_w \Delta P_w - \Delta a_w \Delta \left(\frac{\rho_w z}{144} \right) = \frac{1}{\Delta t} \left[\left(\frac{V_p S_w}{B_w} \right)^{n+1} - \left(\frac{V_p S_w}{B_w} \right)^n \right] + q_{wsc}, \dots \dots \dots (2.17)$$

Gas :

$$\Delta a_g \Delta P_g - \Delta a_g \Delta \left(\frac{\rho_g z}{144} \right) + \Delta R_{so} a_o \Delta P_o - \Delta R_{so} a_o \Delta \left(\frac{\rho_o z}{144} \right) + \Delta R_{sw} a_w \Delta P_w -$$

$$\Delta R_{sw} a_w \left(\frac{\rho_w z}{144} \right) =$$

$$\frac{1}{\Delta t} \left[\left(\frac{V_p S_g}{B_g} \right)^{n+1} + R_{so}^{n+1} \left(\frac{V_p S_o}{B_o} \right)^{n+1} + R_{sw}^{n+1} \left(\frac{V_p S_w}{B_w} \right)^{n+1} - \left(\frac{V_p S_g}{B_g} \right)^n - \dots \right]$$

$$\left[\dots R_{so}^n \left(\frac{V_p S_o}{B_o} \right)^n - R_{sw}^n \left(\frac{V_p S_w}{B_w} \right)^n \right] + q_{gsc}, \dots \dots \dots (2.18)$$

Eqs. (2.16), (2.17) and (2.18) contain gravity terms. Now, we are going to add the capillary terms:

$$P_{cow} = P_o - P_w \quad \therefore P_w = P_o - P_{cow}, \dots \dots \dots (2.19)$$

$$P_{cog} = P_g - P_o \quad \therefore P_g = P_{cog} + P_o, \dots \dots \dots (2.20)$$

Rewrite Equations (2.16), (2.17) and (2.18) by substituting P_w and P_g as we want our pressure terms to be with respect to P_o :-

Oil :

$$\Delta a_o \Delta P_o - \Delta a_o \Delta \left(\frac{\rho_o z}{144} \right) = \frac{1}{\Delta t} \left[\left(\frac{V_p S_o}{B_o} \right)^{n+1} - \left(\frac{V_p S_o}{B_o} \right)^n \right] + q_{osc}, \dots \dots \dots (2.21)$$

Water :

$$\Delta a_w \Delta P_o - \Delta a_w \Delta P_{cow} - \Delta a_w \Delta \left(\frac{\rho_w z}{144} \right) = \frac{1}{\Delta t} \left[\left(\frac{V_p S_w}{B_w} \right)^{n+1} - \left(\frac{V_p S_w}{B_w} \right)^n \right] + q_{wsc}, \dots \dots (2.22)$$

Gas :

$$\Delta a_g \Delta P_{cog} - \Delta a_g \Delta P_o - \Delta a_g \Delta \left(\frac{\rho_g z}{144} \right) + \Delta R_{so} a_o \Delta P_o - \Delta R_{so} a_o \Delta \left(\frac{\rho_o z}{144} \right) + \Delta R_{sw} a_w \Delta P_o -$$

$$- \Delta R_{sw} a_w \Delta P_{cow} - \Delta R_{sw} a_w \Delta \left(\frac{\rho_w z}{144} \right)$$

$$\frac{1}{\Delta t} \left[\left(\frac{V_p S_g}{B_g} \right)^{n+1} + R_{so}^{n+1} \left(\frac{V_p S_o}{B_o} \right)^{n+1} + R_{sw}^{n+1} \left(\frac{V_p S_w}{B_w} \right)^{n+1} - \left(\frac{V_p S_g}{B_g} \right)^n - R_{so}^n \left(\frac{V_p S_o}{B_o} \right)^n - \dots \right] \\ \left[\dots R_{sw}^n \left(\frac{V_p S_w}{B_w} \right)^n \right] + q_{gsc}, \dots \dots \dots (2.23)$$

Expanding and rearranging the gas-phase terms only and putting all the known terms in the LHS (left-hand-side of equation) and the unknowns in RHS (right-hand-side of equation):

$$\Delta t \left[\Delta a_g \Delta P_{cog} - \Delta a_g \Delta P_o - \Delta a_g \Delta \left(\frac{\rho_g z}{144} \right) + \Delta R_{so} a_o \Delta P_o - \Delta R_{so} a_o \Delta \left(\frac{\rho_o z}{144} \right) + \Delta R_{sw} a_w \Delta P_o \right] \\ - R_{sw} a_w \Delta P_{cow} - \Delta R_{sw} a_w \Delta \left(\frac{\rho_w z}{144} \right) \pm q_{gsc} \\ - R_{so}^{n+1} \left[\Delta t \left(\Delta a_o \Delta P_o - \Delta a_o \Delta \left(\frac{\rho_o z}{144} \right) - q_{osc} \right) + \left(\frac{V_p S_o}{B_o} \right)^n \right] \\ - R_{sw}^{n+1} \left[\Delta t \left(\Delta a_w \Delta P_o - \Delta a_w \Delta P_{cow} - \Delta a_w \Delta \left(\frac{\rho_w z}{144} \right) \pm q_{wsc} \right) + \left(\frac{V_p S_w}{B_w} \right)^n \right] \\ \left(\frac{V_p S_g}{B_g} \right)^n + R_{so}^n \left(\frac{V_p S_o}{B_o} \right)^n + R_{sw}^n \left(\frac{V_p S_w}{B_w} \right)^n = \left(\frac{V_p S_g}{B_g} \right)^{n+1}, \dots \dots \dots (2.24)$$

Since we need another equation for each grid block in order to find a unique algebraic solution, our fourth equation will simply be the volumetric equation saying that the saturations sum up to unity. This assures that all the fluid volumes fit the pore volume.

$$S_o^{n+1} + S_w^{n+1} + S_g^{n+1} = I, \dots \dots \dots (2.25)$$

Rearranging equations to solve explicitly for S_o^{n+1} , S_w^{n+1} , S_g^{n+1} :

$$S_o^{n+1} = \left\{ \Delta t \left[\Delta a_o \Delta P_o - \Delta a_o \Delta \left(\frac{\rho_o z}{144} \right) - q_{o,sc} \right] + \left(\frac{V_p S_o}{B_o} \right)^n \right\} \left(\frac{B_o}{V_p} \right)^{n+1}, \dots \dots \dots (2.26)$$

$$S_w^{n+1} = \left\{ \Delta t \left[\Delta a_w \Delta P_o - \Delta a_w \Delta P_{cow} - \Delta a_w \Delta \left(\frac{\rho_w z}{144} \right) \pm q_{w,sc} \right] + \left(\frac{V_p S_w}{B_w} \right)^n \right\} \left(\frac{B_w}{V_p} \right)^{n+1} \dots \dots \dots (2.27)$$

$$S_g^{n+1} = \left\{ \Delta t \left[(-\Delta a_g + \Delta R_{so} a_o + \Delta R_{sw} a_w - R_{so}^{n+1} \Delta a_o - R_{sw}^{n+1} \Delta a_w) \Delta P_o + \Delta a_g \Delta P_{cog} - \right. \right. \\ \left. \left. (\Delta R_{sw} a_w - R_{sw}^{n+1} \Delta a_w) \Delta P_{cow} - \Delta a_g \Delta \left(\frac{\rho_g z}{144} \right) - (\Delta R_{so} a_o - R_{so}^{n+1} \Delta a_o) \Delta \left(\frac{\rho_o z}{144} \right) - \right. \right. \\ \left. \left. (\Delta R_{sw} a_w - R_{sw}^{n+1} \Delta a_w) \Delta \left(\frac{\rho_w z}{144} \right) \pm q_{g,sc} + R_{so} q_{o,sc} \pm R_{sw}^{n+1} q_{w,sc} \right] + \left(\frac{V_p S_g}{B_g} \right)^n + \right. \\ \left. (R_{so}^n - R_{so}^{n+1}) \left(\frac{V_p S_o}{B_o} \right)^n + (R_{sw}^n - R_{sw}^{n+1}) \left(\frac{V_p S_w}{B_w} \right)^n \right\} \left(\frac{B_g}{V_p} \right)^{n+1}, \dots \dots \dots (2.28)$$

We note that all P^{n+1} are known from the previous step, so the right-hand side will be easy to evaluate. These are the explicit calculations with only one unknown. They may be solved in any order.

We can now derive the pressure equation that we need. We start by simply adding the saturation equations, noting that the summation of the saturations is equal to unity.

$$1 = \left\{ \Delta t \left[\Delta a_o \Delta P_o - \Delta a_o \Delta \left(\frac{\rho_o z}{144} \right) - q_{o,sc} \right] + \left(\frac{V_p S_o}{B_o} \right)^n \right\} \left(\frac{B_o}{V_p} \right)^{n+1} + \\ \left\{ \Delta t \left[\Delta a_w \Delta P_o - \Delta a_w \Delta P_{cow} - \Delta a_w \Delta \left(\frac{\rho_w z}{144} \right) \pm q_{w,sc} \right] + \left(\frac{V_p S_w}{B_w} \right)^n \right\} \left(\frac{B_w}{V_p} \right)^{n+1} +$$

$$\begin{aligned}
& \left\{ \Delta t \left[\left(-\Delta a_g + \Delta R_{so} a_o + \Delta R_{sw} a_w - R_{so}^{n+1} \Delta a_o - R_{sw}^{n+1} \Delta a_w \right) \Delta P_o + \Delta a_g \Delta P_{cog} - \right. \right. \\
& \left. \left(\Delta R_{sw} a_w - R_{sw}^{n+1} \Delta a_w \right) \Delta P_{cow} - \Delta a_g \Delta \left(\frac{\rho_g z}{144} \right) - \left(\Delta R_{so} a_o - R_{so}^{n+1} \Delta a_o \right) \Delta \left(\frac{\rho_o z}{144} \right) - \right. \\
& \left. \left(\Delta R_{sw} a_w - R_{sw}^{n+1} \Delta a_w \right) \Delta \left(\frac{\rho_w z}{144} \right) \pm q_{gsc} + R_{so} q_{osc} \pm R_{sw}^{n+1} q_{wsc} \right] + \left(\frac{V_p S_g}{B_g} \right)^n + \\
& \left. \left(R_{so}^n - R_{so}^{n+1} \right) \left(\frac{V_p S_o}{B_o} \right)^n + \left(R_{sw}^n - R_{sw}^{n+1} \right) \left(\frac{V_p S_w}{B_w} \right)^n \right\} \left(\frac{B_g}{V_p} \right)^{n+1}, \dots \dots \dots (2.29)
\end{aligned}$$

We note that this equation has now eliminated the unknown saturations since they have summed to unity. We now have an equation with only unknown pressures, P^{n+1} . We now will manipulate this equation to be in a more convenient form for the pressure equation. We multiply by $V_p^{n+1}/\Delta t$ and put the flow terms back on the left-hand side. This results in:

$$\begin{aligned}
& \left\{ \Delta t \left[B_o^{n+1} \Delta a_o \Delta P_o - B_o^{n+1} \Delta a_o \Delta \left(\frac{\rho_o z}{144} \right) - q_{osc} B_o^{n+1} \right] + \right. \\
& \left. \Delta t \left[B_w^{n+1} \Delta a_w \Delta P_o - B_w^{n+1} \Delta a_w \Delta P_{cow} - \Delta a_w \Delta \left(\frac{\rho_w z}{144} \right) \pm q_{wsc} B_w^{n+1} \right] + \right. \\
& \left. \Delta t \left[\left(-B_g^{n+1} \Delta a_g + B_g^{n+1} \Delta R_{so} a_o + B_g^{n+1} \Delta R_{sw} a_w - R_{so}^{n+1} B_g^{n+1} \Delta a_o - R_{sw}^{n+1} B_g^{n+1} \Delta a_w \right) \Delta P_o \right. \right. \\
& \left. \left. + B_g^{n+1} \Delta a_g \Delta P_{cog} - \left(B_g^{n+1} \Delta R_{sw} a_w - R_{sw}^{n+1} B_g^{n+1} \Delta a_w \right) \Delta P_{cow} - B_g^{n+1} \Delta a_g \Delta \left(\frac{\rho_g z}{144} \right) - \right. \right. \\
& \left. \left. \left(B_g^{n+1} \Delta R_{so} a_o - R_{so}^{n+1} B_g^{n+1} \Delta a_o \right) \Delta \left(\frac{\rho_o z}{144} \right) - \left(B_g^{n+1} \Delta R_{sw} a_w - R_{sw}^{n+1} B_g^{n+1} \Delta a_w \right) \Delta \left(\frac{\rho_w z}{144} \right) - \right. \right. \\
& \left. \left. B_g^{n+1} \left(\pm q_{gsc} - R_{so}^{n+1} q_{osc} \pm R_{sw}^{n+1} q_{wsc} \right) \right. \right. \\
& \left. = V_p^{n+1} - \left(\frac{V_p S_o}{B_o} \right)^n B_o^{n+1} - \left(\frac{V_p S_w}{B_w} \right)^n B_w^{n+1} - \left(\frac{V_p S_g}{B_g} \right)^n B_g^{n+1} \right.
\end{aligned}$$

$$-\left(R_{so}^n - R_{so}^{n+1}\right)\left(\frac{V_p S_o}{B_o}\right)^n B_g^{n+1} - \left(R_{sw}^n - R_{sw}^{n+1}\right)\left(\frac{V_p S_w}{B_w}\right)^n B_g^{n+1}, \dots \dots \dots (2.30)$$

We have several values on the right-hand side which depend on the new pressure, P^{n+1} . We want to replace these values with "chord slopes", so we can solve directly for P^{n+1} with coefficients that are "almost constant".

$$V_p^{n+1} = V_p^n + \frac{V_p^{n+1} - V_p^n}{P^{n+1} - P^n} (P^{n+1} - P^n), \dots \dots \dots (2.31)$$

$$B_o^{n+1} = B_o^n \left[1 - c_o (P^{n+1} - P^n) + \frac{B_g^{n+1}}{B_o^n} (R_{so}^{n+1} - R_{so}^n) \right], \dots \dots \dots (2.32)$$

$$B_w^{n+1} = B_w^n \left[1 - c_w (P^{n+1} - P^n) + \frac{B_g^{n+1}}{B_w^n} (R_{sw}^{n+1} - R_{sw}^n) \right], \dots \dots \dots (2.33)$$

$$B_g^{n+1} = B_g^n [1 - c_g (P^{n+1} - P^n)], \dots \dots \dots (2.34)$$

These relationships are now substituted in the right-hand side of our pressure equation:

$$\begin{aligned} RHS = & V_p^n \left[1 + c_f (P^{n+1} - P^n) \right] - V_p^n S_o^n \left[1 - c_o (P^{n+1} - P^n) + \frac{B_g^{n+1}}{B_o^n} (R_{so}^{n+1} - R_{so}^n) \right] - \\ & V_p^n S_w^n \left[1 - c_w (P^{n+1} - P^n) + \frac{B_g^{n+1}}{B_w^n} (R_{sw}^{n+1} - R_{sw}^n) \right] - \\ & V_p^n S_g^n \left[1 - c_g (P^{n+1} - P^n) \right] - \left(R_{so}^{n+1} - R_{so}^n \right) \left(\frac{V_p S_o}{B_o} \right)^n B_g^{n+1} - \\ & \left(R_{sw}^{n+1} - R_{sw}^n \right) \left(\frac{V_p S_w}{B_w} \right)^n B_g^{n+1}, \dots \dots \dots (2.35) \end{aligned}$$

Let:

$$c_t = c_f + c_o S_o^n + c_w S_w^n + c_g S_g^n, \dots \dots \dots (2.36)$$

$$RHS = V_p^n \left[(c_f + c_o S_o^n + c_w S_w^n + c_g S_g^n) (P^{n+1} - P^n) \right], \dots \dots \dots (2.37)$$

$$RHS = V_p^n c_t (P^{n+1} - P^n), \dots \dots \dots (2.38)$$

Now, let us rearrange our equation again, moving the production terms from the LHS to RHS:

$$\begin{aligned} & (B_o^{n+1} - R_{so}^{n+1} B_g^{n+1}) \Delta a_o P_o - (B_o^{n+1} - R_{so}^{n+1} B_g^{n+1}) \Delta a_o \Delta \left(\frac{\rho_o z}{144} \right) + \\ & (B_w^{n+1} - R_{sw}^{n+1} B_g^{n+1}) \Delta a_w P_o - (B_w^{n+1} - R_{sw}^{n+1} B_g^{n+1}) \Delta a_w \Delta P_{cow} - \\ & (B_w^{n+1} - R_{sw}^{n+1} B_g^{n+1}) \Delta a_w \Delta \left(\frac{\rho_w z}{144} \right) + \\ & B_g \left[-\Delta a_g \Delta P_o + \Delta R_{so} a_o \Delta P_o + \Delta R_{sw} a_w \Delta P_o + \Delta a_g \Delta P_{cog} - \Delta a_g \Delta \left(\frac{\rho_g z}{144} \right) - \right. \\ & \left. \Delta R_{so} a_o \Delta \left(\frac{\rho_o z}{144} \right) - \Delta R_{sw} a_w \Delta \left(\frac{\rho_w z}{144} \right) - \Delta R_{sw} a_w \Delta P_{cow} \right] \\ & = \frac{V_p^n C_t}{\Delta t} (P_o^{n+1} - P_o^n) + q_{osc} B_o^{n+1} \pm q_{wsc} B_w^{n+1} + B_g^{n+1} (\pm q_{gsc} - R_{so}^{n+1} q_{osc} \pm R_{sw}^{n+1} q_{wsc}) \\ & \dots \dots \dots (2.39) \end{aligned}$$

Let us simplify our equation by using these definitions:

$$CGOT = -\Delta a_o \Delta \left(\frac{\rho_o z}{144} \right), \dots \dots \dots (2.40)$$

$$CGWT = -\Delta a_w \Delta \left(P_{cow} + \frac{\rho_w z}{144} \right), \dots \dots \dots (2.41)$$

$$CGgT = \Delta a_g \Delta \left(P_{cog} - \frac{\rho_g z}{144} \right) - \Delta R_{so} a_o \Delta \left(\frac{\rho_o z}{144} \right) - \Delta R_{sw} a_w \Delta \left(P_{cow} + \frac{\rho_w z}{144} \right), \dots (2.42)$$

$$q_t = q_{osc} B_o^{n+1} \pm q_{wsc} B_w^{n+1} + B_g^{n+1} (\pm q_{gsc} - R_{so}^{n+1} q_{osc} \pm R_{sw}^{n+1} q_{wsc}), \dots (2.43)$$

This gives us the final form of the pressure equation which we can finally state as:

$$\begin{aligned} & (B_o^{n+1} - R_{so}^{n+1} B_g^{n+1}) \Delta a_o \Delta P_o + (B_w^{n+1} - R_{sw}^{n+1} B_g^{n+1}) \Delta a_o \Delta P_o + \\ & B_g^{n+1} (\Delta a_g \Delta P_o + \Delta R_{so} a_o \Delta P_o + \Delta R_{sw} a_w \Delta P_o) \\ & = \frac{V_p^n C_t}{\Delta t} (P_o^{n+1} - P_o^n) + q_t - (B_o^{n+1} - R_{so}^{n+1} B_g^{n+1}) CGOT - \\ & (B_w^{n+1} - R_{sw}^{n+1} B_g^{n+1}) CGWT - B_g^{n+1} CGgT, \dots (2.44) \end{aligned}$$

Fig. 2.2 shows a flow chart how IMPES can be implemented in a computer program.

Simulation of a black oil reservoir requires solving a system of partial differential equations. The partial differential equations are approximated by algebraic equations known as finite difference equations. The finite difference equations are obtained by replacing derivatives with approximations derived from truncated Taylor series expansions.

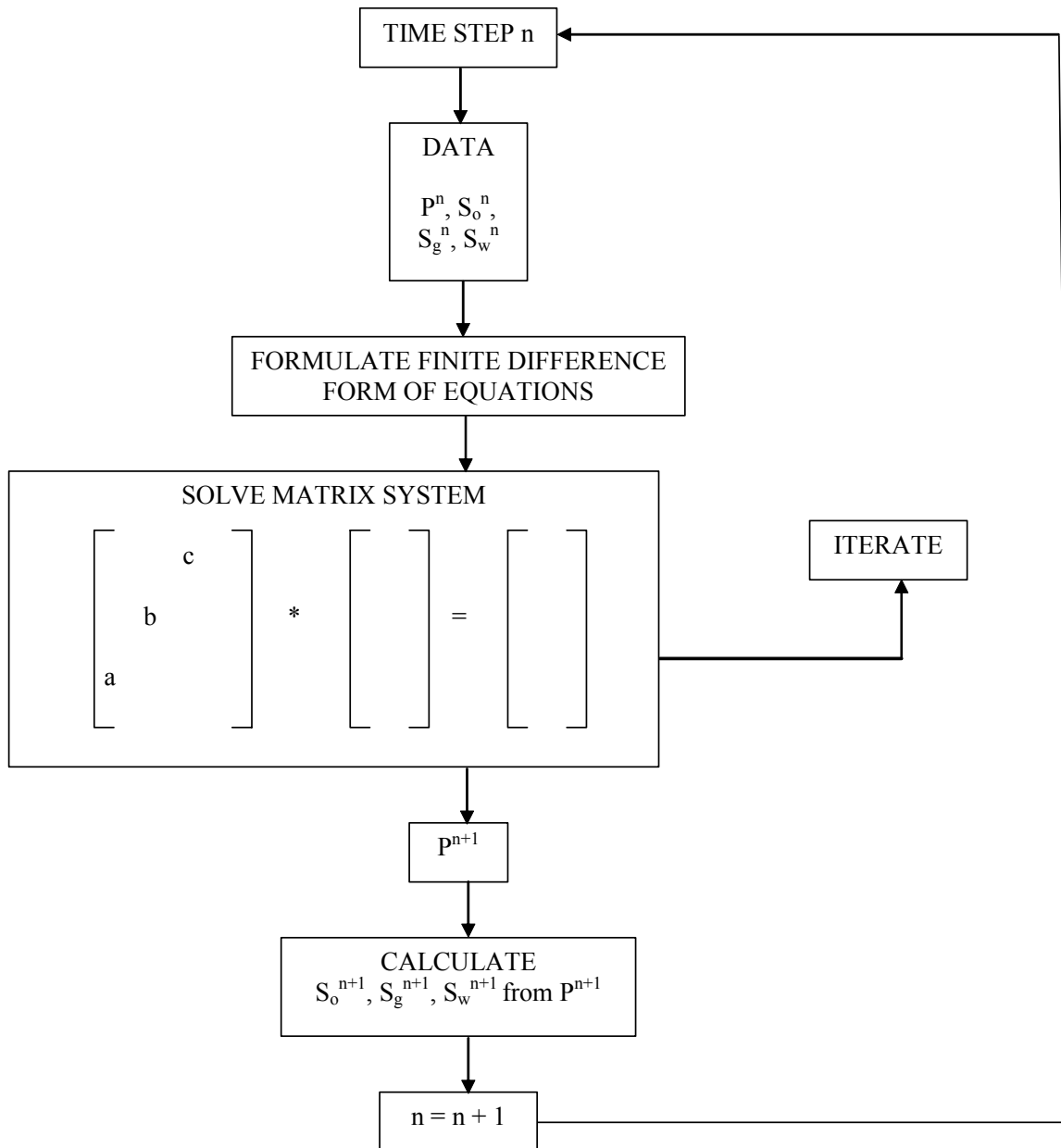


Fig. 2.2 - Flow chart showing how IMPES can be implemented in a computer program

CHAPTER III

PROGRAM CHARACTERISTICS AND PROPERTIES

In this chapter, the framework of Sim2D is discussed, as well as its main attributes and properties. Sim2D is developed using the Visual Basic 6.0 programming language and it is then compiled into an executable file with simple interface to make it an easier to use program. In addition, it is equipped with data control/access and file system.

The Sim2D program simulates isothermal, Darcy's flow in two dimensions. It assumes reservoir fluids can be described by three fluid phases (oil, gas, and water) of constant composition with physical properties that depend on pressure only. Sim2D is designed to be an easy-to-use program which would be suited to simulate primary depletion and basic secondary recovery operations (such as water flooding) in a black-oil reservoir. Sim2D is a finite-difference, implicit pressure-explicit saturation (IMPES) numerical simulator. It contains an iterative solution technique (Bi-Conjugate Gradient) for solving systems of algebraic equations. The well model in Sim2D allows specification of rate or pressure constraints on well performance. Several user-controlled output options are also available.

On top of that, Sim2D provides two types of grid systems, namely the conventional Cartesian grid as well as the proposed grid called the Hybrid Grid Block System (HGB).

3.1 Algorithm of VB Code

Sim2D is developed using the IMPES formulation. The code consists of different subroutines. They are contained by a main subroutine that controls the order of the run and loops the required subroutines over each time step until the last time step is reached. Some basic tasks such as interpolations and averaging are executed by functions instead of subroutines. **Fig. 3.1** shows the algorithm that is employed by Sim2D.

3.1.1 Initialization Data

Initialization data describes the reservoir model grid dimensions and geometry, the distribution of porosity and permeability, relative permeability, fluid PVT data, initial pressure and saturation distributions within the reservoir, specification of the solution method to be used, and run control parameters. After the input data has been read, the required memory is allocated for each variable.

To complete the mathematical description of a reservoir, it is necessary to specify the initial conditions. For the initial conditions at $n = 0$, a value is specified for pressure and saturations. Every node is assigned the values of these initial conditions. Pore volumes are calculated for each grid block and the summation of the reservoir pore volume is stored. Parameters such as formation volume factors, viscosities and compressibility, as well as relative permeabilities at initial conditions are then interpolated from the PVT table provided by the user.

3.1.2 Averaging of Flow Equation Terms

Several parameters in the material balance equation need to be averaged and the most common methods used are summarized in **Table 3.1**.

In multiphase system, one or more relative permeabilities must be assigned that will control the flow of the individual phases from one grid block to the next. In the case of Sim2D simulator, upstream permeability is used. Here, mobilities are evaluated at saturations that exist in the block from which the fluid phases are moving. For instance, if the flow is from left to right, the relative permeability from the $i-1$ grid block will be considered the “upstream block” so $(kro)_{i-1}$ is used for $(kro)_w$. Similarly, if flow is from right to left, the $i+1$ grid block will be upstream instead.

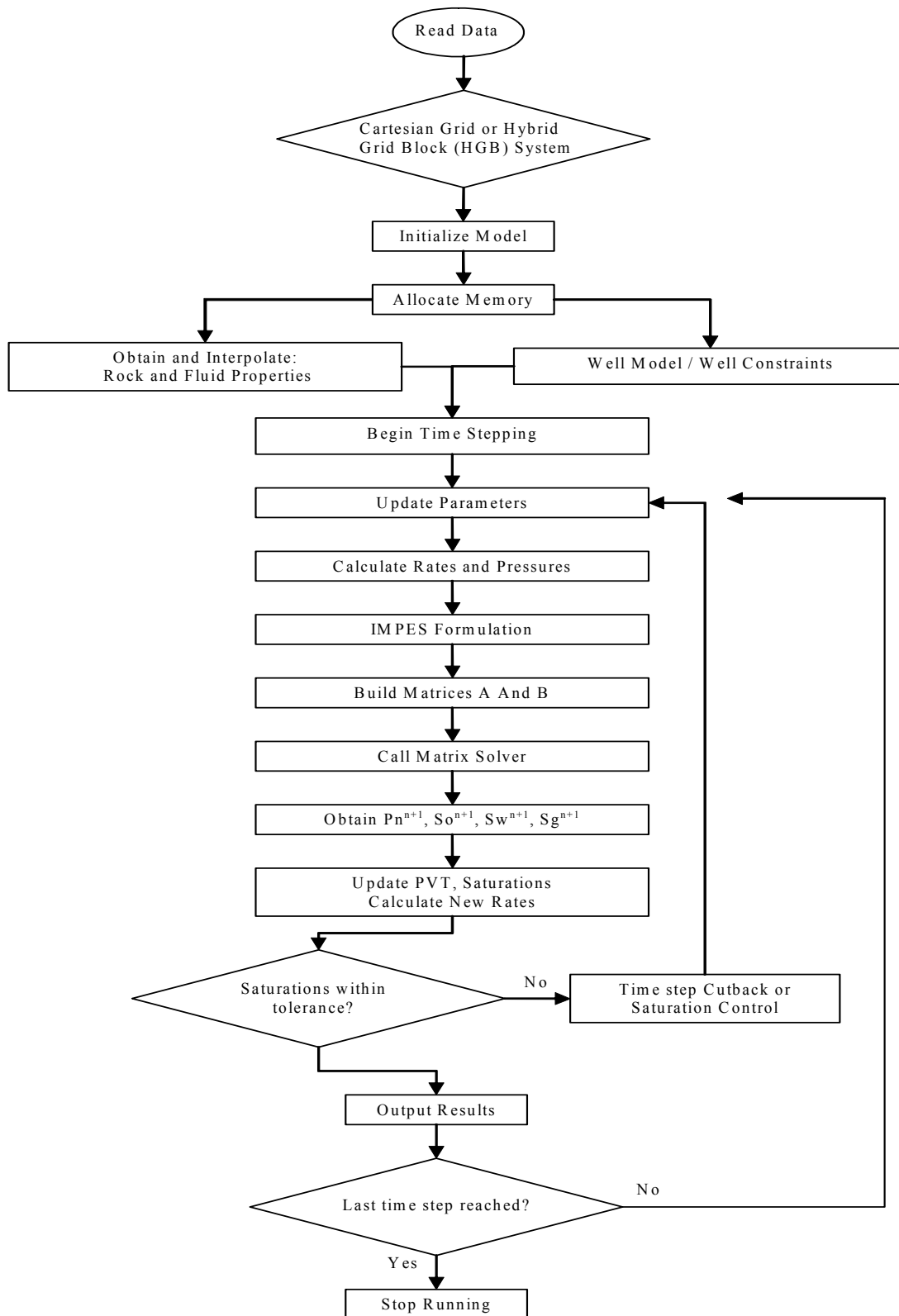


Fig. 3.1 - Flow chart of the Sim2D code

Table 3.1 – Averaging of parameters

Parameter	Method of Averaging
Porosity	Arithmetic Averaging
Viscosity	Arithmetic Averaging
Absolute Permeability	Harmonic Averaging
Relative Permeability	Upstream Weighting
Formation Volume Factor	Arithmetic Averaging

3.1.3 Boundary Conditions

In reservoir simulation problems, initial conditions (initial reservoir pressure) and saturation distributions are required to initialize the model. For example, the initial conditions are obtained by assuming initial capillary and gravity equilibrium. Then, the pressure distribution is obtained by specifying pressure at a given datum and using the fluid pressure gradients to determine pressures at all other depths.

The boundary conditions used in reservoir simulators can be very complicated as the differential equations solved by the simulators require that all boundaries be specified. This includes both internal and external boundaries.

Consider a 2-D flow domain as depicted in **Fig. 3.2**.

The driving force for flow arises from the boundary conditions. Reservoir boundaries are represented physically by faults, pinch outs (porosity, permeability), aquifers, facies change (shales) etc. In the numerical model, these discontinuities are modeled as external boundaries or internal boundaries, depending on the position within the reservoir.

External boundaries are the physical boundaries of the flow domain, while for internal boundaries, either well rates or bottomhole pressures can be specified. If a rate is specified for a well, a Neumann-type boundary condition is generated. Conversely, if the pressure is specified for the wellbore, then a Dirichlet-type boundary condition is obtained.

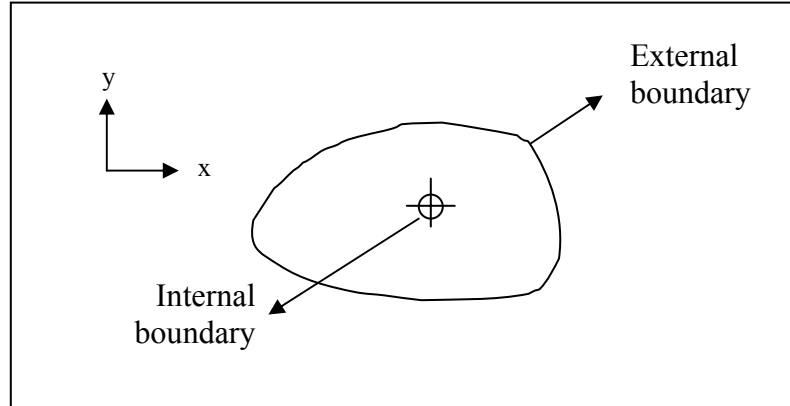


Fig. 3.2 -2-D flow domain with a well

In Sim2D, it assumes that a no-flow outer boundary exists. To model no-flow boundaries, phase transmissibilities across the boundary interfaces are set to zero. This implied that there is no communication or “flux contribution” across the adjacent boundary blocks.

3.1.4 Well Model

A reservoir simulation uses an analytical model to represent flow within a grid block as it enters or leaves a well. This model is called the well model. It is well-known that that pressure of the wellblock is different from the bottomhole well flowing pressure at the well. This is because in general, the grid block dimensions are significantly greater than the wellbore radius. The flow rate in the well is proportional to the difference between the block and well pressures. The coefficient of proportionality is known as the productivity or injectivity index. The geometric part of this term is usually called the well index, and the model used to determine the well index is known as the well model.

Production of fluids by wells is assumed to be similar to flow from a grid block to another grid block. Assuming that Darcy Law holds for flow in a well, writing the Darcy law for radial co-ordinates we have:

$$q = \frac{2\pi k k_r h}{\mu B \ln(r_e / r_w)} (P_i - P_{wf}) \dots\dots\dots(3.1)$$

where,

- k = permeability
- k_r = relative permeability
- h = thickness
- r_e = outer radius (radius of influence)
- r_w = wellbore radius
- P_i = grid block pressure
- P_{wf} = bottomhole well flowing pressure

Since the grid block pressure and all other physical properties are assumed to be centered at the middle of the grid cell, the well is also assumed to be at the center of the grid block.

3.1.4.1 Peaceman's Well Model

The well model presented by Peaceman³⁵ is based on the comparison between numerical and analytical solutions for a repeated five-spot pattern. Peaceman's model also assumes the following:

- (pseudo-)steady state flow
- homogeneous reservoir
- isolated wells
- incompressible flow

Peaceman found that the pressure calculated for a well block is the same as the flowing pressure at an equivalent radius, r_o , where he defined r_o as "the radius at which the steady-state flowing pressure for the actual well is equal to the numerically calculated pressure for the well block". This definition of r_o gives:

$$P_{wf} - P_o = \frac{q\mu}{2\pi} \ln \frac{r_w}{r_o}, \dots\dots\dots(3.2)$$

where,

- P_{wf} = bottomhole well flowing pressure
- P_o = pressure calculated for grid block containing the well
- q = production rate of well
- r_w = wellbore radius

Using uniform square Cartesian grid blocks ($\Delta x = \Delta y$), Peaceman showed that if $\Delta x = \Delta y$ and $k_x = k_y$ then:

$$r_o = 0.2\Delta x, \dots\dots\dots(3.3)$$

In a subsequent paper, Peaceman (1983)³⁶ derived an expression for r_o for an isotropic reservoir with non-square grid blocks:

$$r_o = 0.14(\Delta x^2 + \Delta y^2)^{1/2}, \dots\dots\dots(3.4)$$

For an anisotropic reservoir, Peaceman (1983)³⁶ determined that r_o is given by:

$$r_o = \frac{0.28 \left[\sqrt{k_y/k_x} (\Delta x)^2 + \sqrt{k_x/k_y} (\Delta y)^2 \right]^{1/2}}{(k_y/k_x)^{1/4} + (k_x/k_y)^{1/4}}, \dots\dots\dots(3.5)$$

We can now introduce the term well index, J , which is defined as:

$$J = \frac{2\pi kh}{\ln \frac{r_o}{r_w}}, \dots\dots\dots(3.6)$$

Now Eq. 3.1 can be rearranged and reduced to:

$$q_{\alpha} = J\lambda_{\alpha}(P_i - P_{wf}), \dots\dots\dots(3.7)$$

$$\lambda_{\alpha} = \left(\frac{k_r}{\mu B} \right), \dots\dots\dots(3.8)$$

where,

α = species or phase

λ_{α} = mobility of the species or phase

3.1.4.2 Well Constraints

Producers are operated by the following constraints:

1. Constant flow rate of any one phase

If the rate of any one phase is specified, then the rate of the other phase(s) can be calculated as follows:

$$q_{\alpha} = J\lambda_{\alpha}(P_i - P_{wf}), \dots\dots\dots(3.9)$$

$$P_{wf} = P_i - \frac{q_{\alpha o}}{J\lambda_{\alpha o}}, \dots\dots\dots(3.10)$$

where,

α = unknown phase

αo = known phase

2. Constant bottom hole pressure

If the well bottomhole pressure is specified, then the rate of any phase can be obtained as follows:

$$q_{\alpha} = J\lambda_{\alpha}(P_i - P_{wf}), \dots\dots\dots(3.11)$$

3. Constant liquid rate

If the rate of the liquid phase is specified, then:

$$q_t = B_o q_o + B_w q_w + B_g (q_g - R_s q_o), \dots \dots \dots (3.12)$$

$$\lambda_t = \frac{k_{ro}}{\mu_o} + \frac{k_{rw}}{\mu_w} + \frac{k_{rg}}{\mu_g}, \dots \dots \dots (3.13)$$

Using equation (2), we can rewrite it for the rate of other phase(s):

$$q_o = \frac{q_t \lambda_o}{B_o \lambda_t}, \dots \dots \dots (3.14)$$

Similarly,

$$q_w = \frac{q_t \lambda_w}{B_w \lambda_t}, \dots \dots \dots (3.15)$$

$$q_g = \frac{q_t \lambda_g}{B_g \lambda_t} + R_s q_o, \dots \dots \dots (3.16)$$

Injectors are usually operated at two constraints – either constant injection rate or constant injection pressure. However, in Sim2D, only the constant injection rate constraint is implemented.

4. Constant Injection Rate

If the injection rate of any one phase is specified, then the flowing bottomhole pressure is computed as follows:

$$P_{wf} = P_i + \frac{q_\alpha}{J \lambda_\alpha}, \dots \dots \dots (3.17)$$

5. Constant Injection Pressure

If the injection pressure is specified, then the rate of the injected phase can be obtained as follows:

$$q_\alpha = J \lambda_\alpha (P_{wf} - P_i), \dots \dots \dots (3.18)$$

3.1.5 Time Step Control

In order to ensure that the IMPES formulation used will give accurate solutions, there is a need for a “time step cutback” procedure. The simulator that has been developed has the capability to set time steps automatically by calculating the values of saturation change and adjusting time steps until its changes meet specified tolerance criteria. This tolerance is specified by the user. After the simulator takes a time step, it tests against this tolerance. If it is not met, the calculations made with the time step are discarded and a smaller time step is selected. A special counter called “ncut” is also used so that the user can specify the number of maximum time step cut allowable before it proceeds to the next time step. The algorithm is shown in **Fig. 3.3**.

Of course, sensitivity runs should be performed to determine an acceptable time step (to find a compromise between accuracy of solutions and simulation time required). Also, this simulator allows the user to input the frequency of output desired.

3.1.6 Solution Method – Linear Solver

The finite difference form of the pressure equation leads to a system of linear equation for the i - j unknowns $P_{i,j}^{n+1}$. Here, $P_{i,j}^{n+1}$ denotes the pressure at grid block (i,j) at the new $(n+1)$ time level. Such a system of equations may be written as:

$$\begin{aligned}
 a_{1,1}P_1 + a_{1,2}P_2 + \dots + a_{1,N}P_N &= B_1 \\
 a_{2,1}P_1 + a_{2,2}P_2 + \dots + a_{2,N}P_N &= B_2 \\
 \vdots & \\
 a_{N,1}P_1 + a_{N,2}P_2 + \dots + a_{N,N}P_N &= B_N
 \end{aligned}
 \tag{3.19}$$

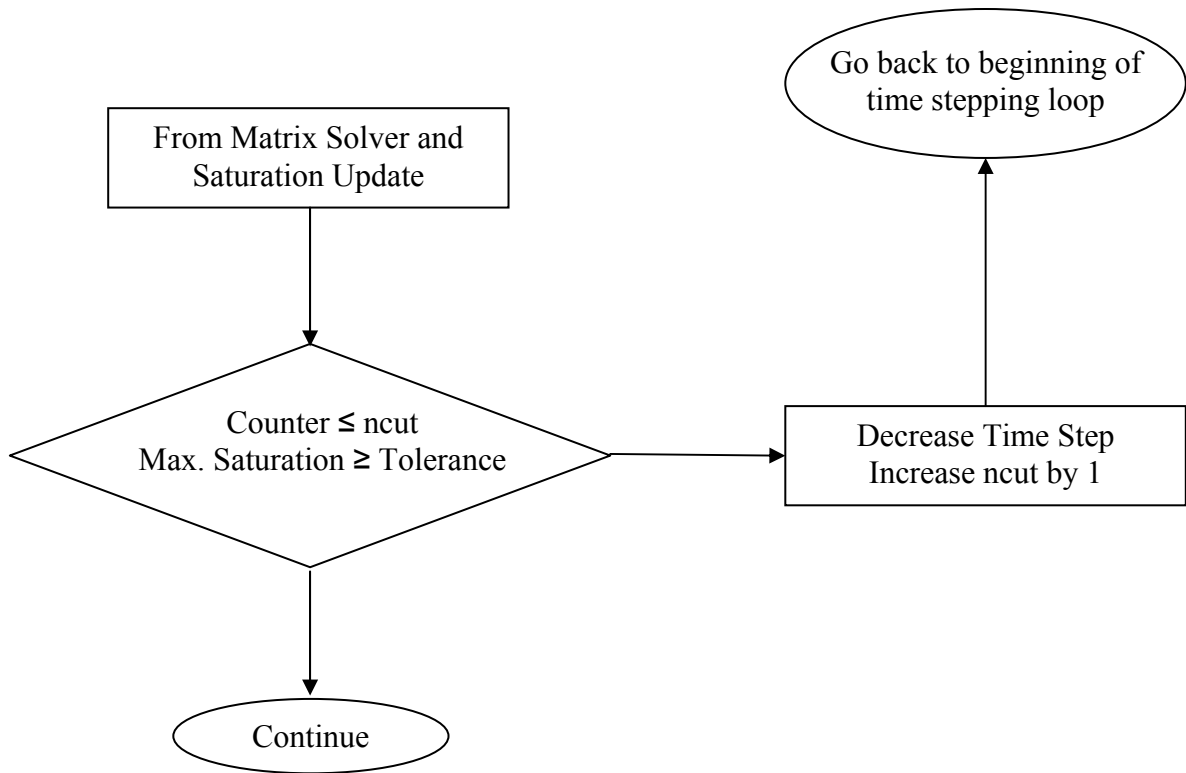


Fig. 3.3 – Algorithm for the time step cutback loop

Alternatively, the same set of equations may be expressed in a more compact form using matrix notation as $\mathbf{AP} = \mathbf{B}$, where \mathbf{A} is the co-efficient matrix, and \mathbf{P} and \mathbf{B} are column vectors as given below.

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & & & \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_N \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix}, \dots \dots \dots (3.20)$$

Various methods exist for solving such a system of linear equations, but generally these methods fall into one of two groups - direct methods, or iterative methods. In Sim2D, the method of preconditioned and stabilized biconjugate-gradient is

used to solve the system of linear equations, which is a type of iterative solver. In the case of the “diagonal grid ordering” for HGB, this method is especially attractive - the co-efficient matrix has a sparse structure, containing a large number of non-zero entries. Because only non-zero elements are used in the iterative methods, these methods require relatively little storage memory. However, the discussion of the matrix solver in details is beyond the scope of this thesis. Even so, regardless of the matrix solver used, all methods should yield the same results and should be accurate within the specified tolerance.

3.2 Spatial Discretization of Cartesian Grid System

Sim2D uses the block-centered finite difference grid. For the Cartesian coordinate system, the gridpoints are defined as the centers of these grid blocks. For a 1-D model, for flow in the x-direction, a block-centered grid system can be constructed as in **Fig. 3.4**. In this figure, a grid system consisting of nx gridblocks is superimposed over a reservoir. These grid blocks have predetermined dimensions of Δx_i that are not necessarily equal. Once the grid blocks are defined, the grid points are placed at the center of the blocks. The boundaries of the i^{th} grid block are designated $x_{i-1/2}$ and $x_{i+1/2}$, whereas the block center is named x_i . These grid block properties are related through the following equations:

$$x_i = (x_{i-1/2} + x_{i+1/2}) / 2, \dots \dots \dots (3.21)$$

$$\Delta x_i = x_{i+1/2} - x_{i-1/2}, \dots \dots \dots (3.22)$$

Fig. 3.4 illustrates the terms in these equations.

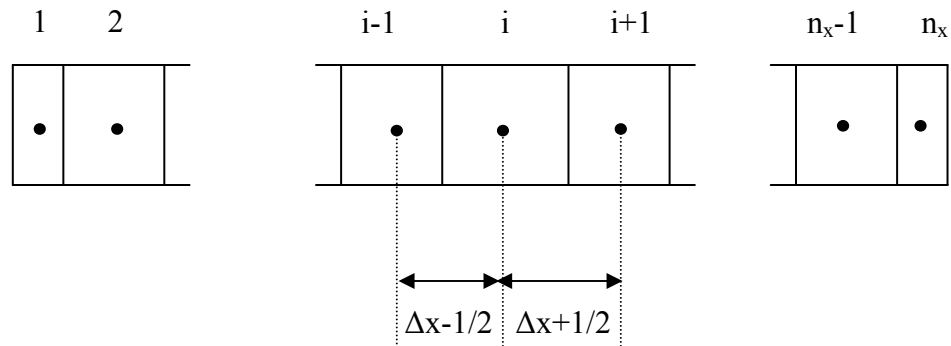


Fig. 3.4 – 1-D, block-centered, finite difference grid

The 2-D Cartesian grid is numbered using a single index system, as shown in **Fig. 3.5**.

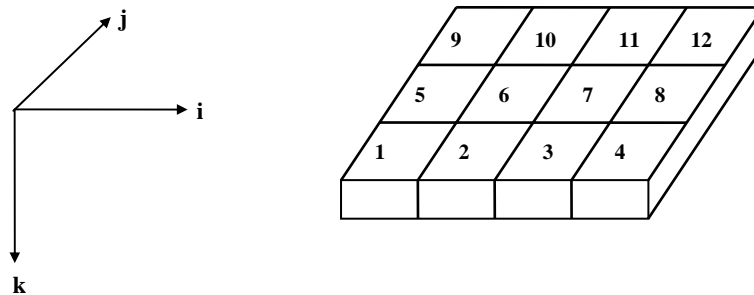


Fig. 3.5 – Grid numbering for the Cartesian grid system

3.3 Implementation of Hybrid Grid-Block (HGB) System

In employing the HGB grid system, several modifications were made in order for this system to be integrated into Sim2D. Changes occur in the areas of calculating intergrid-block transmissibilities, the grid block numbering as well as the well model. These topics will be dealt with in the following discussions:

3.3.1 Grid Block Generation

This method involves using a unique grid-block assignment where rectangular grid blocks are interspersed with octagonal grid blocks. The boundaries are then populated with triangular grid blocks. Thus, the entire domain will consist of a “structured” grid block system. This arrangement is shown in **Fig. 3.6**, with the black dots representing the center of each grid block. The fluid will flow to four directions in each of the octagons and two directions in each of the rectangles and triangles.

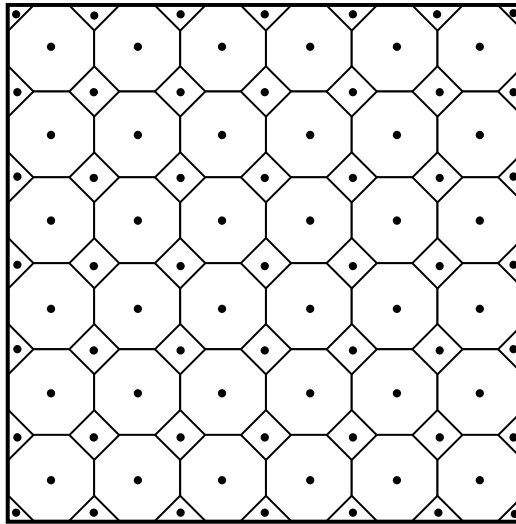


Fig. 3.6 - HGB grid model

The basic elements are generated from input data, including the number of grid blocks and the interval length in each direction. Based on this predefined information, parameters such as the total grid blocks, the internal length between each grid block and the number of flow directions can be calculated.

3.3.2 Transmissibility Calculations

Since HGB assumes a block-centered geometry, transmissibility calculations are based upon the distances between the centers of each grid block. Using **Fig. 3.7** as an example, Cell #1 is connected to Cell #3 “through” cross-sectional area A_{13} . Its transmissibility can be calculated as shown in Eq. 3.23.

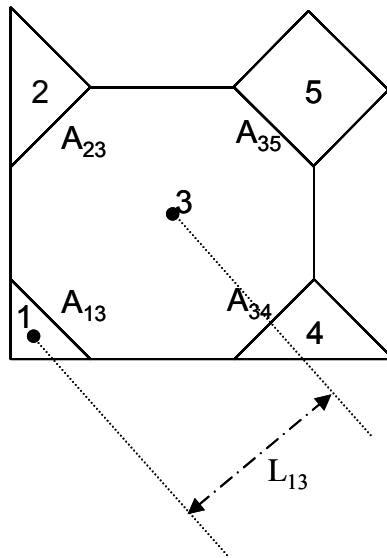


Fig. 3.7 – Example of transmissibility calculations in HGB

$$T_{13} = \frac{0.00633kA_{13}}{\Delta L_{13}}, \dots\dots\dots(3.23)$$

where,

- T = transmissibility
- A = cross sectional area
- L = distance between the centers of two neighboring grid blocks perpendicular to the cross-sectional area

From earlier sections, we have derived the general oil material balance equation as:

$$\Delta a_o \Delta P_o - \Delta a_o \Delta \left(\frac{\rho_o z}{144} \right) = \frac{1}{\Delta t} \left[\left(\frac{V_p S_o}{B_o} \right)^{n+1} - \left(\frac{V_p S_o}{B_o} \right)^n \right] + q_{o.sc} \dots\dots\dots(2.16)$$

To illustrate how the left-hand side of this equation would look like for a 2-D HGB model, let's assume that we have named each grid block as shown in **Fig. 3.8**.

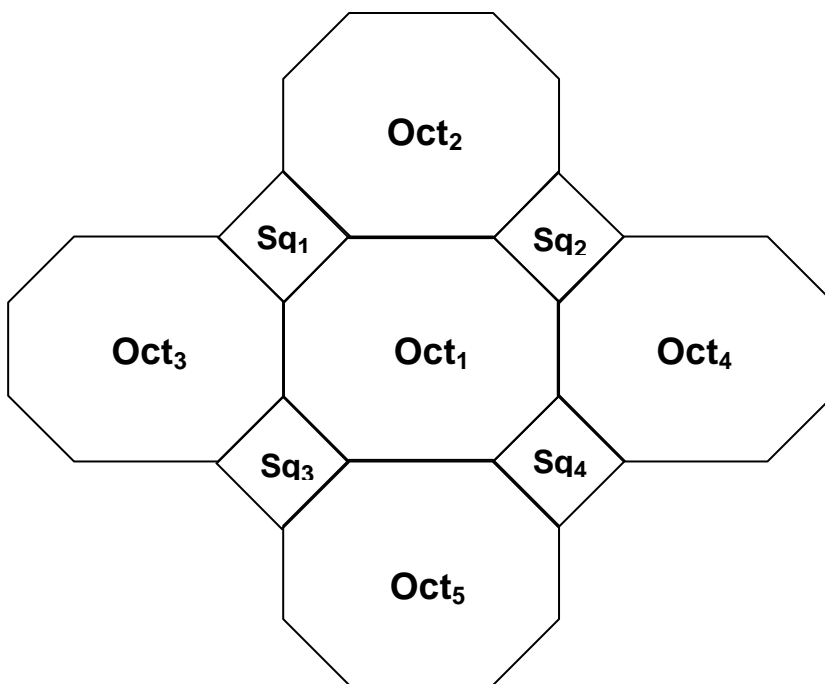


Fig. 3.8 – Calculations in eight directions for a central octagonal block

Taking *Oct₁* as the central grid block where it is surrounded by eight other grid blocks, this equation relative to *Oct₁* can be written and simplified as:

$$\Delta a_o \Delta P = a_{oN}(P_{oct2} - P_{oct1}) + a_{oW}(P_{oct3} - P_{oct1}) + a_{oE}(P_{oct4} - P_{oct1}) + a_{oS}(P_{oct5} - P_{oct1}) + a_{oNW}(P_{sq1} - P_{oct1}) + a_{oNE}(P_{sq2} - P_{oct1}) + a_{oSW}(P_{sq3} - P_{oct1}) + a_{oSE}(P_{sq4} - P_{oct1}), \dots (3.24)$$

where,

N	=	North
E	=	East
S	=	South
W	=	West
NW	=	Northwest
SW	=	Southwest
NE	=	Northeast
SE	=	Southeast

3.3.3 Grid Numbering and Structure of Matrix Forms

The structure of the coefficient matrix depends on the dimensions of the problem and the ordering of the grid blocks. The objective of using different grid block-ordering schemes is to reduce the computational work involved in solving a system of finite difference equations. Numbering system for the 2-D grid and the corresponding non-zero coefficient in the matrix equation for $AP = B$. Therefore, we should order the points in such a way that the band width is the minimum possible.

Using the HGB model, several numbering systems were tested. To illustrate the importance of grid numbering and its effect on the matrix coefficient band width, three different schemes of grid ordering will be presented.

Considering a simple 3x2 case, the following figure shows an example grid ordering and the corresponding matrix structures.

Grid Ordering #1:

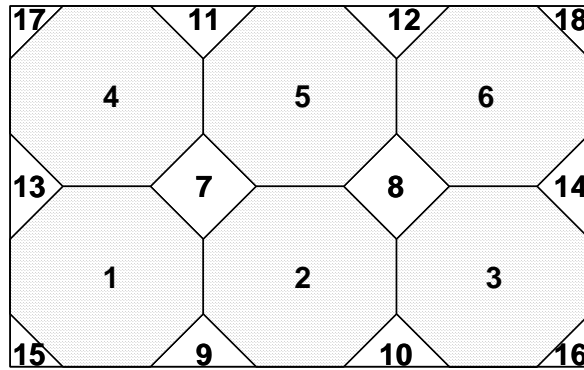


Fig. 3.9 - Ordering #1: 2-D grid block ordering

For the two-dimensional problem shown in **Fig. 3.9** above, the matrix A would have the form as shown in **Fig. 3.10**. We can see that matrix formed is sparse, irregular and the band width is large.

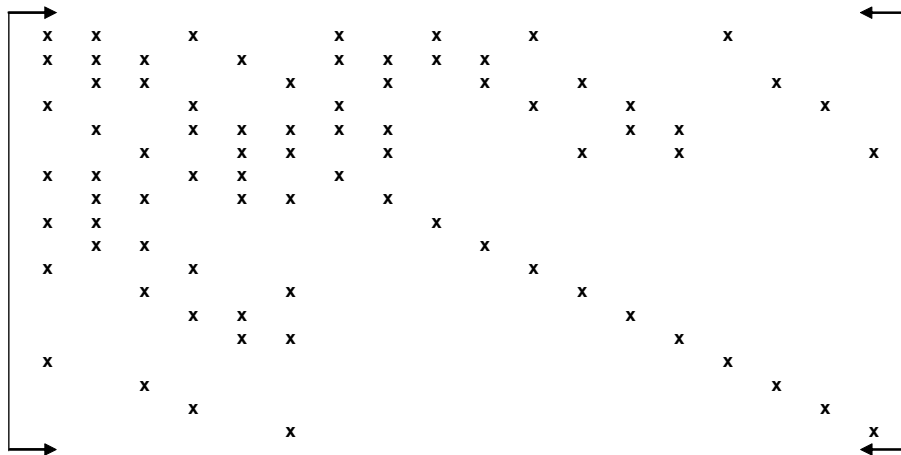


Fig. 3.10 - Locations of matrix elements in Ordering #1

Grid Ordering #2:

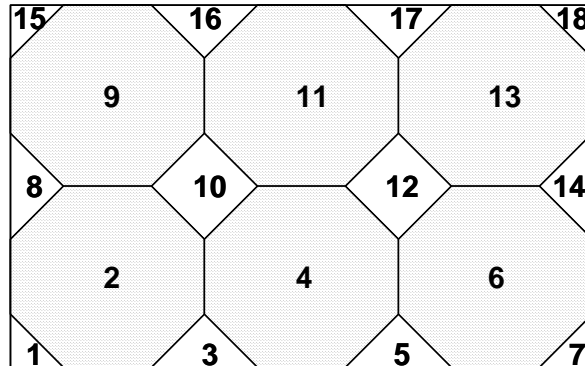


Fig. 3.11 - Ordering #2: 2-D gridblock ordering

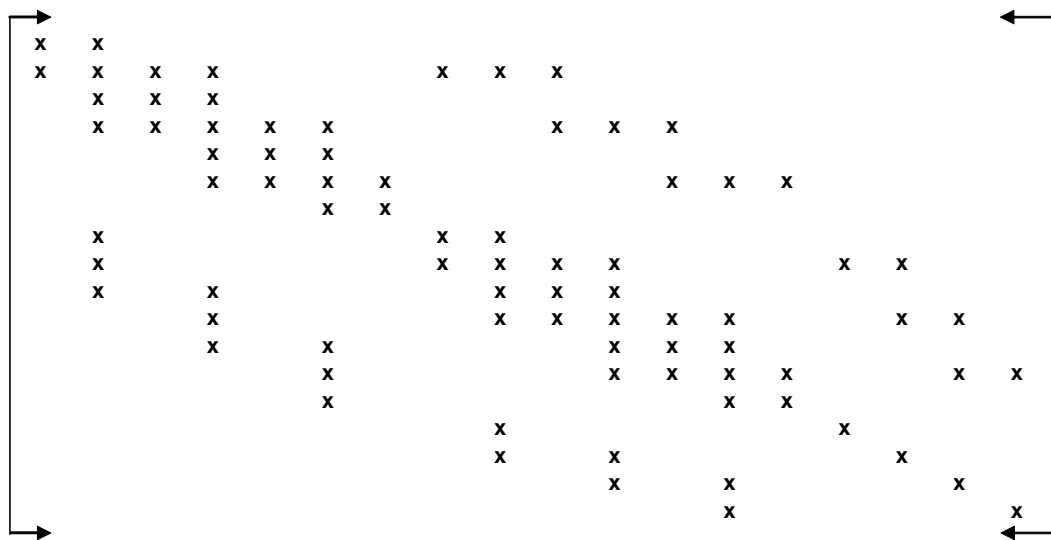


Fig. 3.12 - Locations of matrix elements in Ordering #2

For the next numbering shown in **Fig. 3.11** above, the matrix A would have the form as shown in **Fig. 3.12**. The matrix formed is more regular than Grid Numbering #1, but it is still quite sparse.

Fig. 3.13 shows a type of “diagonal ordering” where the cells are numbered consecutively along the diagonals starting with the shortest direction, as shown by the direction of the arrows. This method groups the cells by “diagonal count”, and increases as we move from the lower left through the grid to the upper right. The band width in **Fig. 3.14** is less than **Figs. 3.10** and **3.12** which gives us a computing advantage as it requires less arithmetic to solve the matrix equation.

Grid Ordering #3:

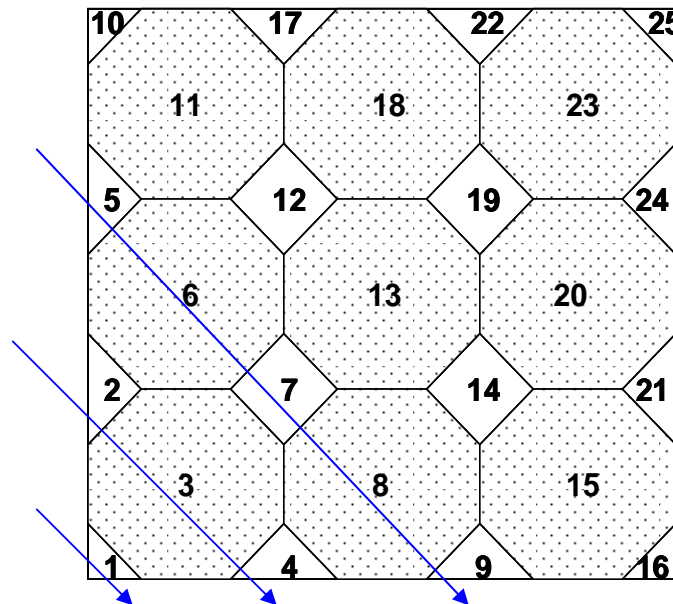


Fig. 3.13 – Ordering #3: Re-ordering of grid blocks to reduce band width

3.3.4 Palagi’s Well Model

In using HGB grid model, we need a different well model as Peaceman’s well model is formulated for square grid blocks. Palagi³⁷ presented an analytical well model based on Peaceman’s work which can be applied to grids of any geometry. This model assumes that the pressure at all grid blocks that are neighbors of the well block can be

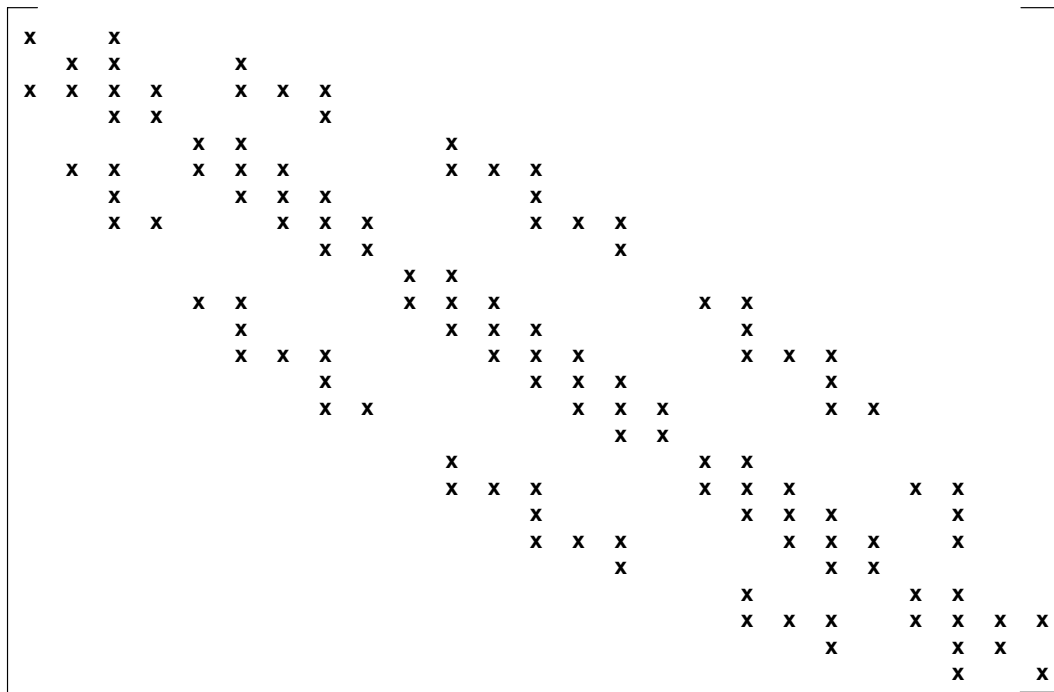


Fig. 3.14 - Locations of matrix elements in Ordering #3

evaluated by the radial flow equation around the well. Also, flow is assumed to be radial around the well block despite of the location of the well. This assumption uses Eq. 3.25 and it is shown in **Fig. 3.15**.

Currently, the wells can only be placed in the square and octagonal grid blocks.

$$r_o = \exp \left(\frac{\sum_j \left(\frac{b}{d} \right)_{ij} - \theta_{ij}}{\sum_j \left(\frac{b}{d} \right)_{ij}} \right) \dots \dots \dots (3.25)$$

where,

- j = grid block that is neighbor of well block i
- b_{ij} = length of side of polygon
- d_{ij} = distance between the centers of grid block i and j

θ_{ij} = angle open to flow ($\theta = 2\pi$ for an internal well, i.e. well located in the center of the block that is opened to flow in all directions)

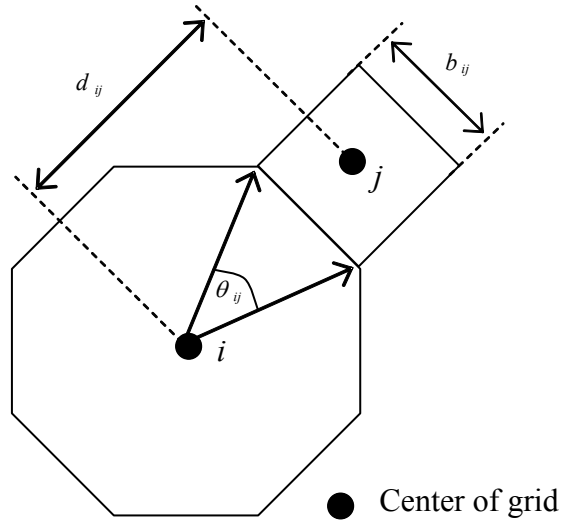


Fig. 3.15 – Well model for a polygon

Palagi derived a special case for Eq. 3.25 when the polygon of interest has equal sides, where:

$$(b/d)_{ij} = \tan(\pi/N), \dots \dots \dots (3.26)$$

Substituting Eq. 3.26 into Eq. 3.25 (with $\theta = 2\pi$) and solving for r_o gives:

$$r_o = d_{ij} \exp\left(\frac{-2\pi}{N \tan(\pi/N)}\right), \dots \dots \dots (3.27)$$

where,

N = number of sides of the polygonal grid block containing the well

CHAPTER IV

GRID ORIENTATION EFFECT

It has been demonstrated by various authors that two-dimensional simulations of immiscible displacements with unfavorable mobility ratio exhibit grid orientation effect. In fact, despite the fact that the reservoir is isotropic and homogeneous with favorable mobility ratio, there can still be an effect of grid orientation.

To examine this effect, we conducted simulations using Eclipse™ 100 (ECL™ 100) of a quarter five-spot waterflood using parallel and diagonal grid systems, as defined and illustrated in **Fig. 4.1**.

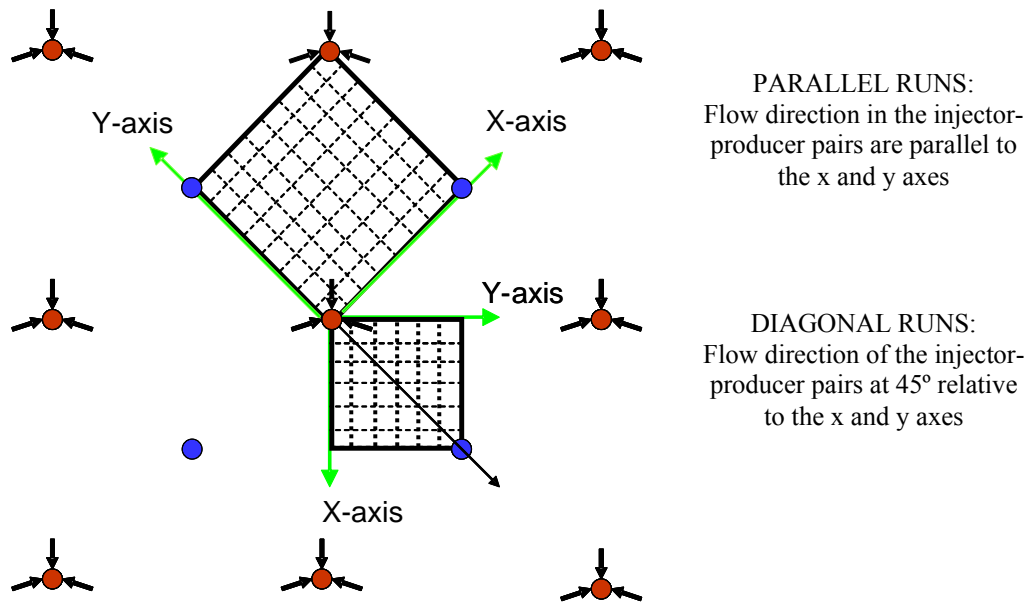


Fig. 4.1 – Parallel and diagonal orientation for simulations of waterflooding in five-spot symmetry elements

Table 4.1 – Data used for five-spot pattern simulations

Rock Permeability, k	= 100 mD
Porosity, ϕ	= 0.20
Net Pay Thickness	= 10 ft
Producer-Injector Distance	\approx 933.4 ft
Production Rate, q_o	= 18 STB/D
Injection Rate, q_w	= 18 STB/D
Initial Pressure	= 5000 psi
Area of Reservoir (Parallel)	= 20 acres
Area of Reservoir (Diagonal)	= 10 acres

Fluid-Rock Properties :

$$k_{rw} = \frac{S_w^2}{\frac{\mu_o}{\mu_w}(1 - S_w^2) - S_w^2} \quad k_{ro} = 1 - k_{rw}$$

Mobility Ratio:

$$M = \frac{k'_{rw} / \mu_w}{k'_{ro} / \mu_o}$$

Table 4.2 - Grid sizes used in Cartesian grid models

Diagonal Grid	Grid Block Size ($\Delta x = \Delta y$)	Parallel Grid	Grid Block Size ($\Delta x = \Delta y$)
6 x 6	132.0 ft	8 x 8	133.34 ft
11 x 11	66.0 ft	15 x 15	66.67 ft
21 x 21	33.0 ft	29 x 29	33.36 ft
41 x 41	16.5 ft	57 x 57	16.67 ft

A parallel grid system is a grid that is oriented parallel to injector-producer pairs. Meanwhile, a diagonal grid system is a grid oriented at 45° between injector and

producer pairs. The distance of a producer to an injector and the size of the grid blocks are the same for both grid systems. Waterflood simulations were performed for oil/water mobility ratios (M) of 0.5, 1.0 and 10. The input data and the grid sizes are shown in **Tables 4.1 and 4.2**. The porosities and permeabilities of the boundary blocks are modified so that a five-spot pattern can be simulated using a block-centered model. The well index was also modified to reflect these changes. The porosity, permeability and well model modifications are shown respectively in the schematic diagrams shown in **Figs. 4.2-4.4**. Essentially, only the area bounded inside the dotted lines in Figs. 4.2-4.4 are modeled.

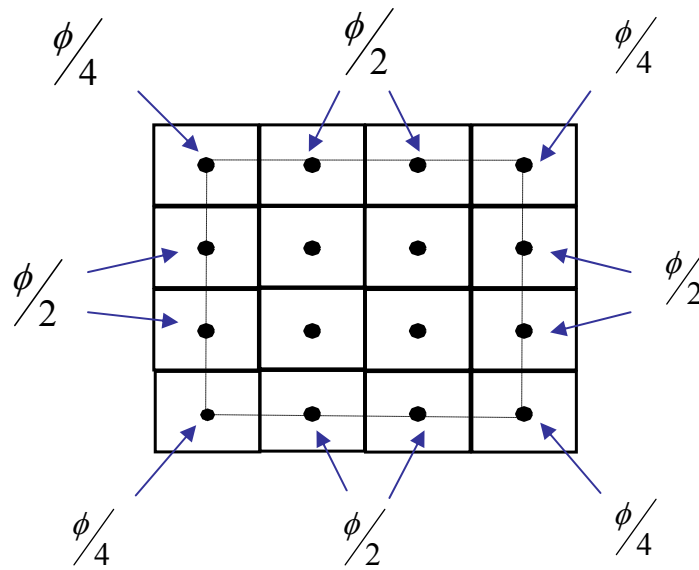


Fig. 4.2 – Porosity modifications

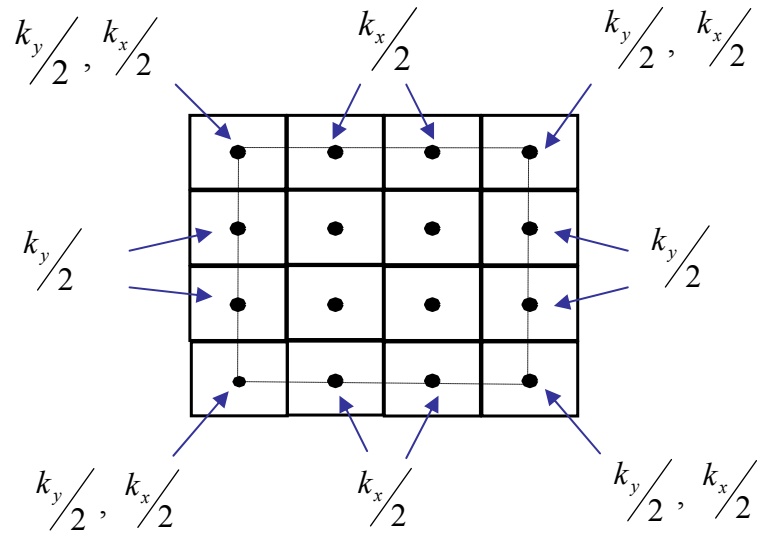


Fig. 4.3 – Permeability modifications

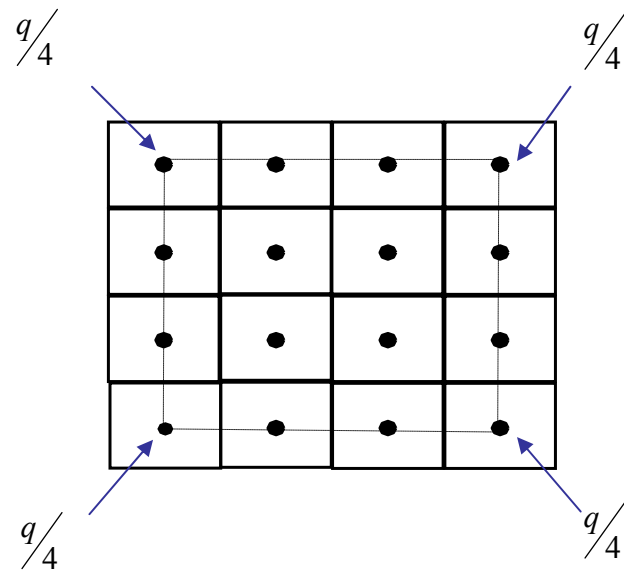


Fig. 4.4 – Well model modifications

Since the distance of injector to producer is the same, we expect to get similar recovery performance from both grid systems. However, when we compare the recovery performance of parallel grid blocks of 8x8 and diagonal of 6x6, the recovery performances from both grid blocks are different as seen in **Fig. 4.5**. This is because rotation of the coordinate axes results in differing amounts of truncation error.²⁵ As pointed out by previous authors, the grid orientation effect can be reduced by increasing the resolution of the grid blocks for cases with favorable mobility ratio ($M \leq 1.0$).¹¹

Thus, we increased the number of grid blocks in diagonal and parallel grid blocks at $M = 0.5$. We found that recovery performance is not very sensitive to the number of grid blocks in the diagonal model (**Fig. 4.6**).

However, as the number of the parallel grid blocks is increased, the recovery performance changes gradually until it converges to a single recovery curve (**Fig. 4.7**). The recovery performances of finer grid blocks in both models (diagonal 21x21 vs. parallel 29x29), were compared. We found that the grid orientation effect was reduced (**Fig. 4.8**) as the difference in the recovery performance curve between the diagonal 21x21 and parallel 29x29 was reduced (**Fig. 4.5**), compared to those results from the parallel 8x8 and diagonal 6x6 grid blocks.

Moreover, as shown in **Fig. 4.9**, the sweep efficiency at $M = 1.0$ decreases gradually as the number of grid blocks are increased. As for the parallel grid, once again, the recovery performance converges at 15x15 number of grid blocks and higher, as shown in **Fig. 4.10**. The results for both the diagonal and parallel grid show a close agreement when the grid block numbers are at 57x57 and 41x41 for the parallel and diagonal grid models, respectively, finer than the grid block sizes required for agreement at $M=0.5$ (**Fig. 4.11**).

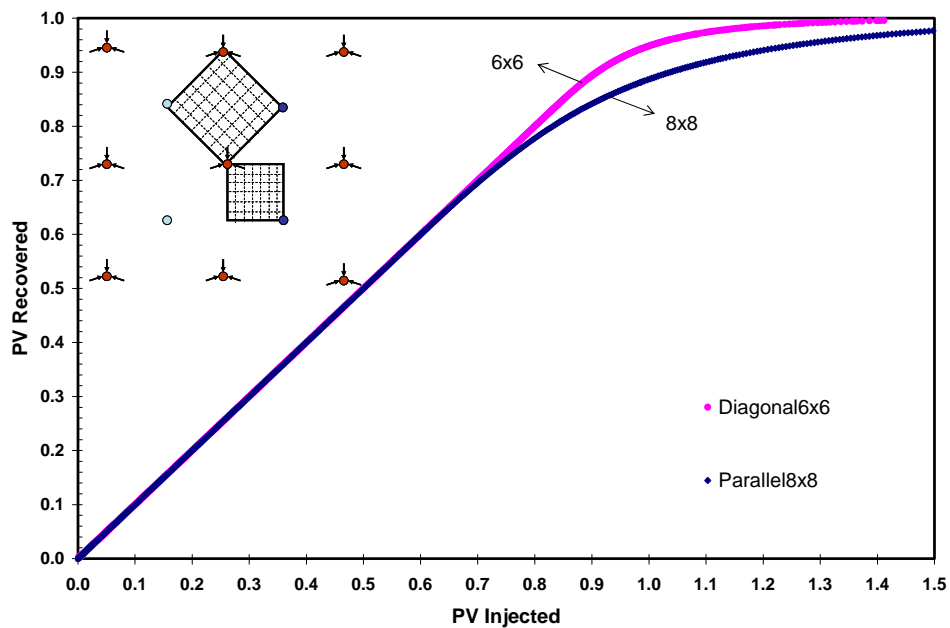


Fig. 4.5 – Predicted performance at $M=0.5$ for parallel (8x8) and diagonal (6x6) grid blocks

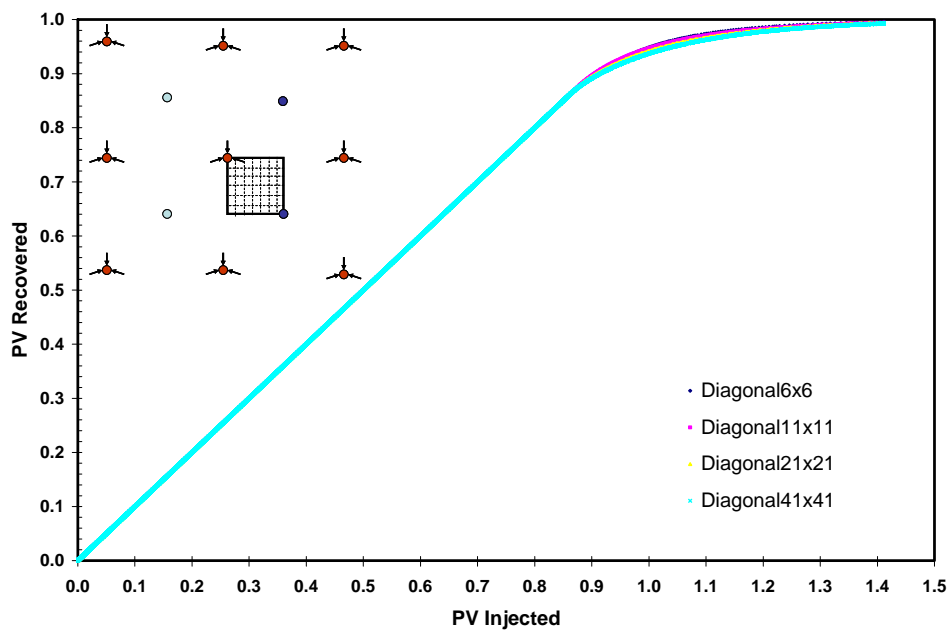


Fig. 4.6 – Predicted performance at $M=0.5$ for different number of diagonal grid blocks

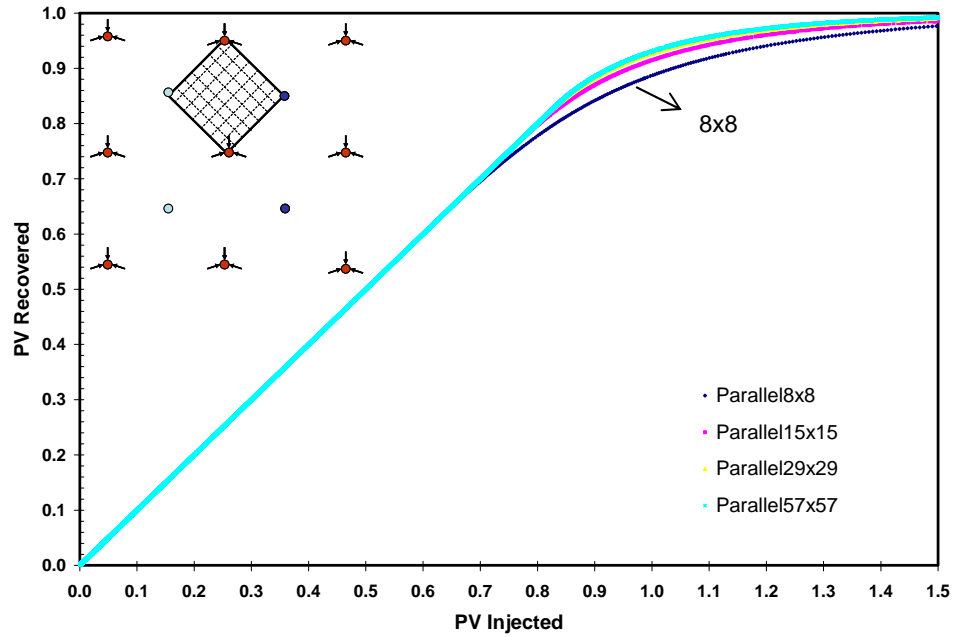


Fig. 4.7 – Predicted performance at $M=0.5$ for different number of parallel grid blocks

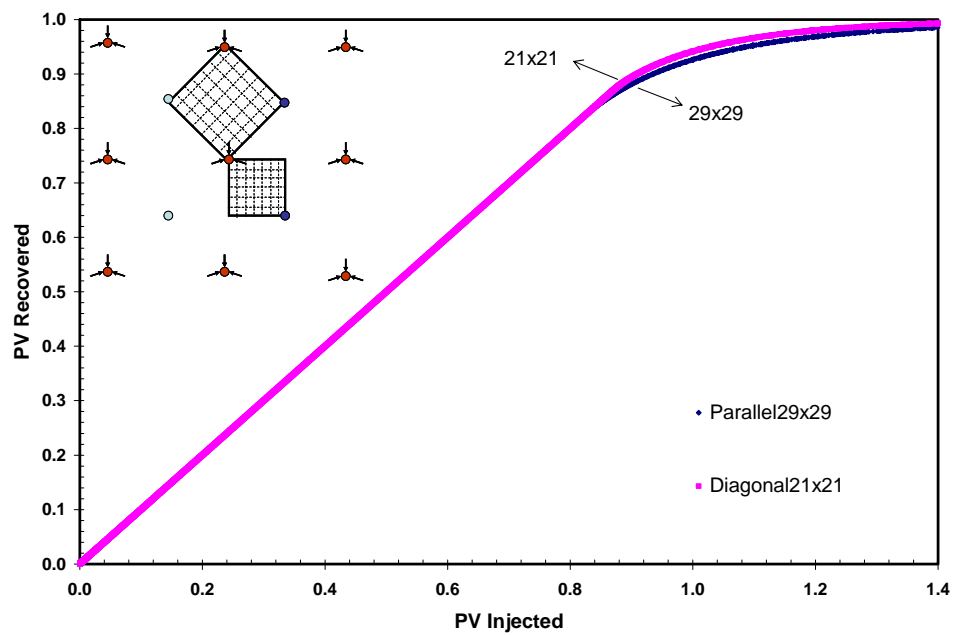


Fig. 4.8 – Predicted performance at $M=0.5$ for parallel (29x29) and diagonal (21x21) grid blocks

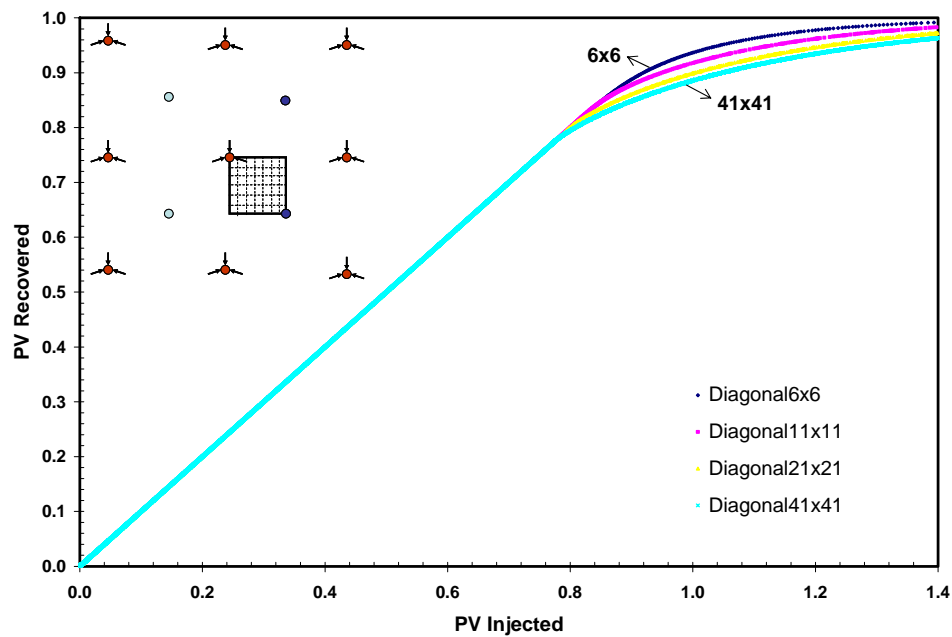


Fig. 4.9 – Predicted performance at $M=1.0$ for different number of diagonal grid blocks

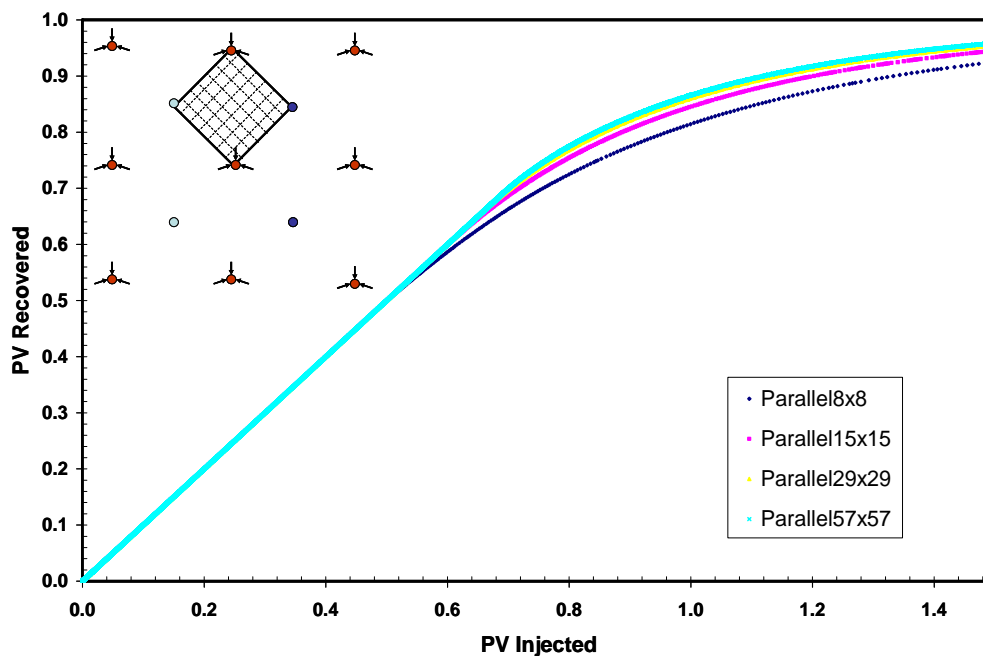


Fig. 4.10 – Predicted performance at $M=1.0$ for different number of parallel grid blocks

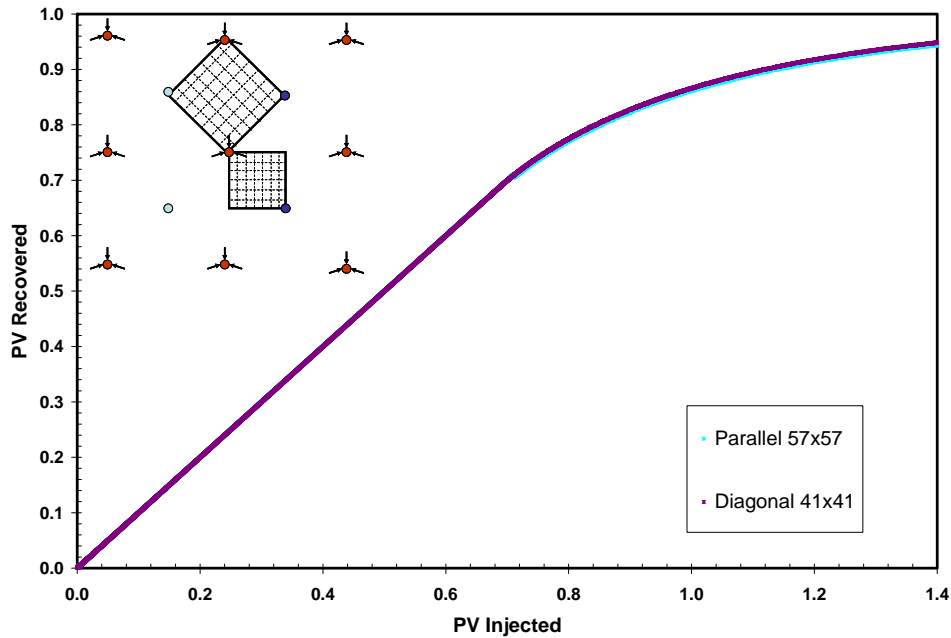


Fig. 4.11 – Predicted performance at $M=1.0$ for parallel (57x57) diagonal (41x41) grid blocks

As the mobility ratio is increased to $M=10.0$, the performance of the diagonal grid does not follow a certain trend (**Fig. 4.12**). On the other hand, for the parallel grid, the solution does not seem to converge to a single curve even when a large number of grid blocks were used, as seen in **Fig. 4.13**. Thus, as the grid spacing is refined, the performance of diagonal and parallel models actually diverges for the grid spacings investigated here.

The saturation map for diagonal grid model shows “viscous fingering” at the saturation front while the parallel model also shows a distorted front (**Fig. 4.14**).

Based on this study, we can conclude that grid refinement can help to minimize the grid orientation effect when we have favorable mobility ratios, i.e. at $M=1.0$ or less. However, at an unfavorable mobility ratio of $M=10.0$ for displacement problems as shown, neither the parallel nor diagonal orientation can be used reliably.

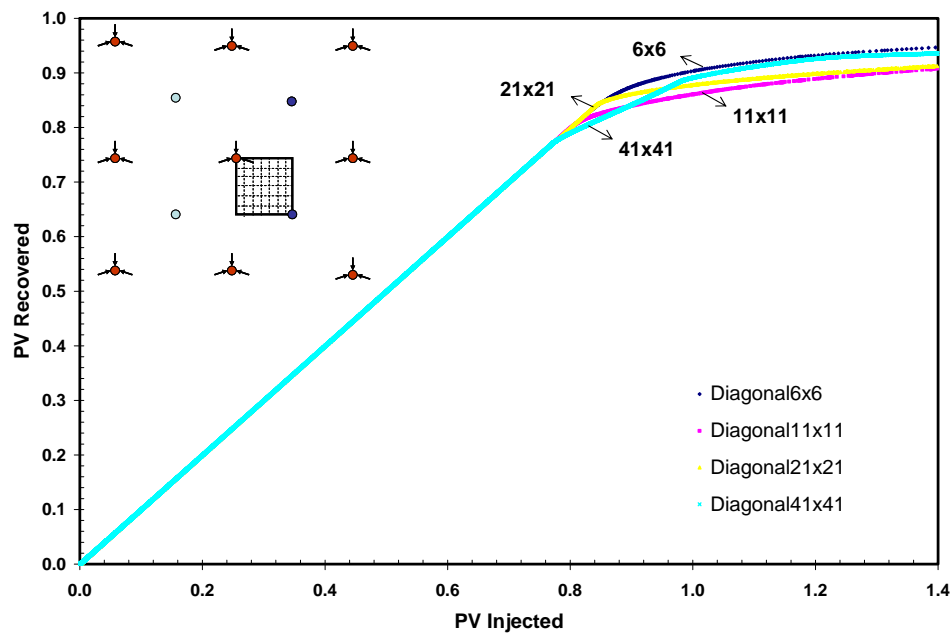


Fig. 4.12 – Predicted performance at $M=10.0$ for different number of diagonal grid blocks

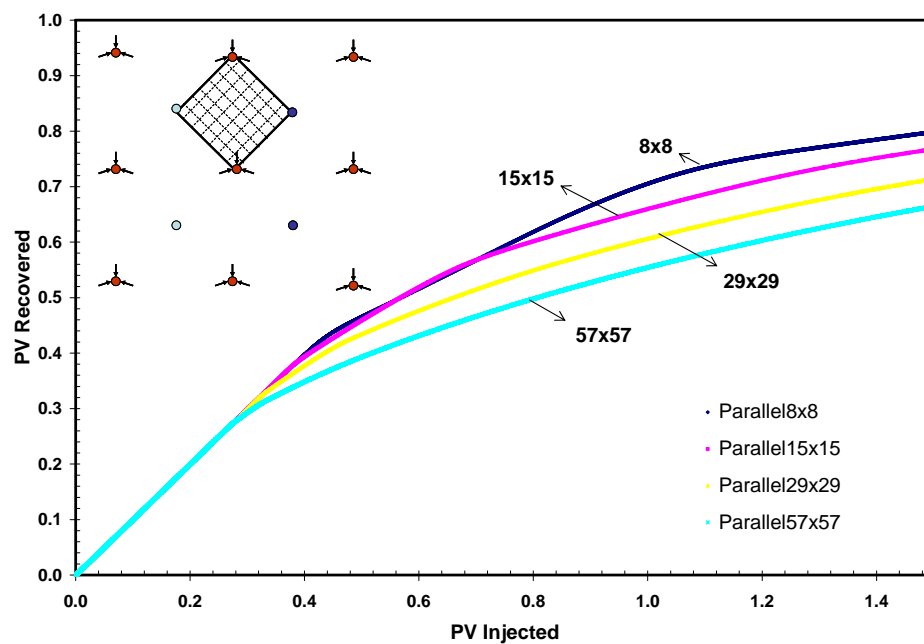


Fig. 4.13 – Predicted performance at $M=10.0$ for different number of parallel grid blocks

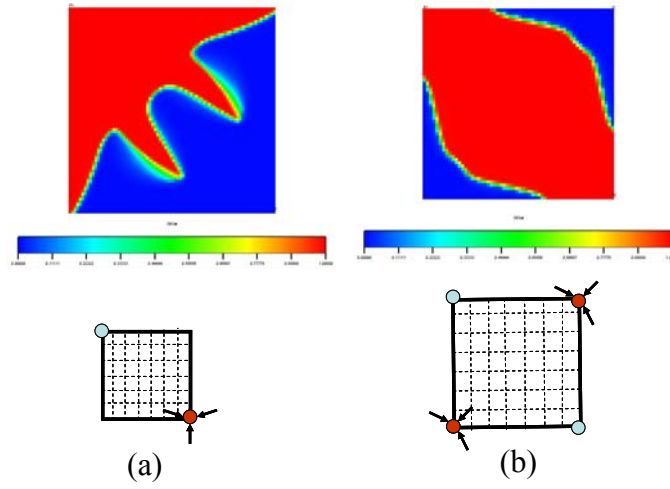


Fig. 4.14 – Saturation distribution map for (a) diagonal model, and (b) parallel model at $PV_{inj} = 1.0$ for $M = 10.0$

CHAPTER V

PROGRAM VALIDATION AND PERFORMANCE OF HGB MODEL

This chapter provides an example problem to validate the Sim2D simulator, illustration of the grid orientation effects in conventional Cartesian grid, as well as the application of HGB model. Whenever possible, the Sim2D solution is compared with a commercially available black oil simulator on the same problem, namely GeoQuest's (2003A) and Eclipse™ 100 (ECL™ 100). Single-point upstream weighting of mobility and IMPES solution mode were used in all runs.

To test the viability of the HGB grid, a two-dimensional IMPES simulator was developed and HGB grid is incorporated. Since the HGB grid cannot be validated “directly” with any commercial simulators to the best of the author's knowledge, the Cartesian grid model in Sim2D is validated with rectangular Cartesian grid models in ECL™ 100 as shown in earlier section. Once the algorithm is validated, it is then applied to the HGB grid.

5.1 Program Validation

This example case is based on a 2-D reservoir model grid of 5x5. The two-phase model contains one producer and one injector well. Both injector and producer are under a constant rate constraint. The well is rate constrained to a 100 scf/Day oil production. The reservoir is homogeneous and is initially at 5000 psi of undersaturated oil and connate water. The simulation was run until the minimum bottomhole pressure (BHP) of 2000 psi is reached. Other reservoir and simulation data is shown in **Table 5.1** and **Fig. 5.1**. The oil and water rates as well as the water cut performance are shown in **Fig. 5.2** while the Sim2D pressure solutions are presented in **Figs. 5.3-5.8** along with the results of the same problem runs on ECL™ 100.

Table 5.1 – Reservoir data

Area of Reservoir	50000 ft ² or 1.15 acres
Grid Block Dimension, $\Delta x = \Delta y$ (ft)	100
Reservoir Thickness (ft)	10
Permeability (mD)	100
Porosity	0.20
Initial Water Saturation	0.20
Initial Oil Saturation	0.80
Well Radius (ft)	0.33
Initial Pressure (psi)	5000
Minimum Bottomhole Pressure (psi)	2000
Rock Compressibility (1/psi)	3E-06

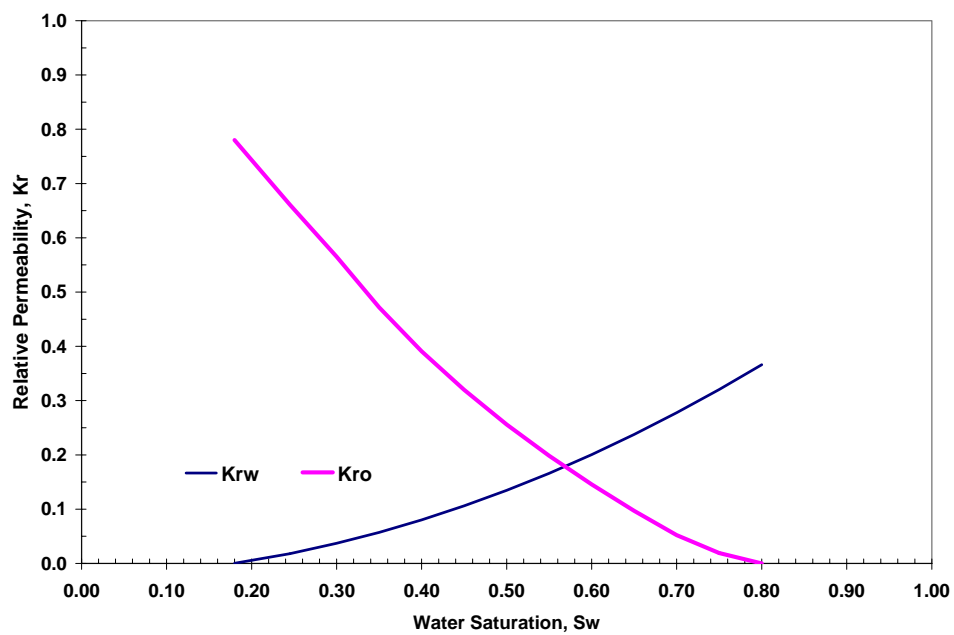


Fig. 5.1 – Relative permeability curve

Table 5.2 – PVT data

Pressure (psi)	Oil FVF (rcf/scf)	Oil Viscosity (cp)	Oil Compressibility (1/psi)	Water FVF (rcf/scf)	Water Viscosity (cp)	Water Compressibility (1/psi)
6014.7	1.0620	1.0400	2.51E-06	1.0190	0.5060	3.00E-06
5014.7	1.0647	0.8951	2.51E-06	1.0221	0.5060	3.00E-06
4014.7	1.0673	0.7705	2.51E-06	1.0251	0.5060	3.00E-06
3014.7	1.0700	0.6631	2.51E-06	1.0282	0.5060	3.00E-06
2514.7	1.0714	0.6152	2.51E-06	1.0298	0.5060	3.00E-06
2014.7	1.0727	0.5708	2.51E-06	1.0313	0.5060	3.00E-06
1514.7	1.0741	0.5295	2.51E-06	1.0328	0.5060	3.00E-06
1014.7	1.0754	0.4913	2.51E-06	1.0344	0.5060	3.00E-06
514.7	1.0768	0.4558	2.51E-06	1.0360	0.5060	3.00E-06
14.7	1.0781	0.4228	2.51E-06	1.0375	0.5060	3.00E-06

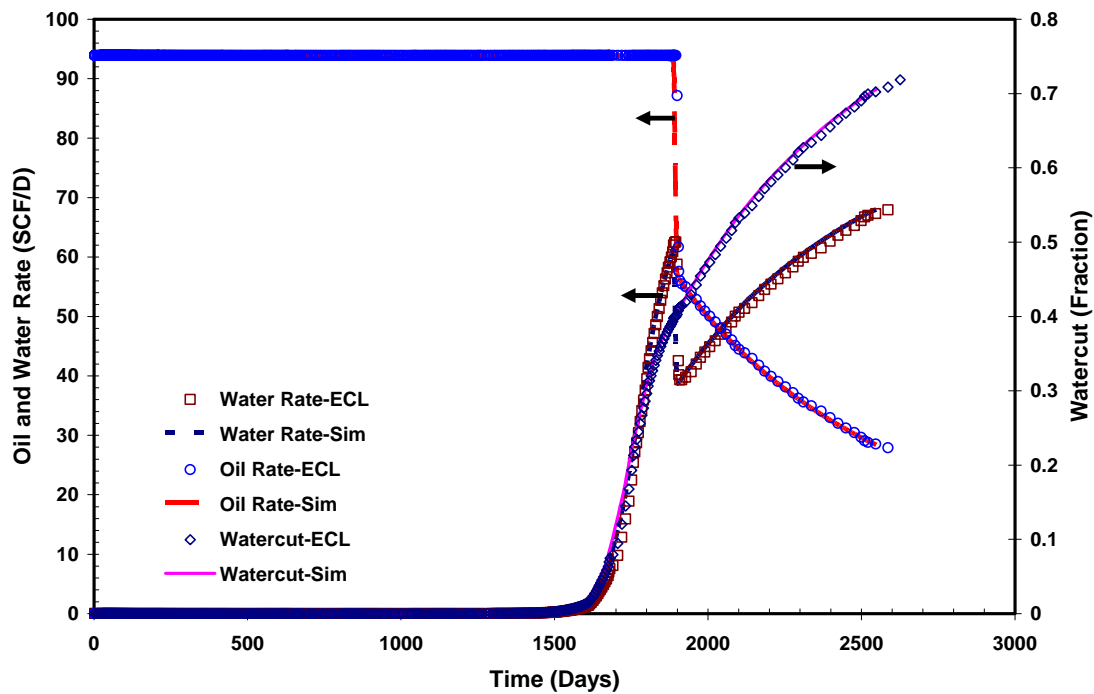


Fig. 5.2 – Comparison of Sim2D oil and water rates and watercut with ECL™ 100 showing good agreement between the two simulators

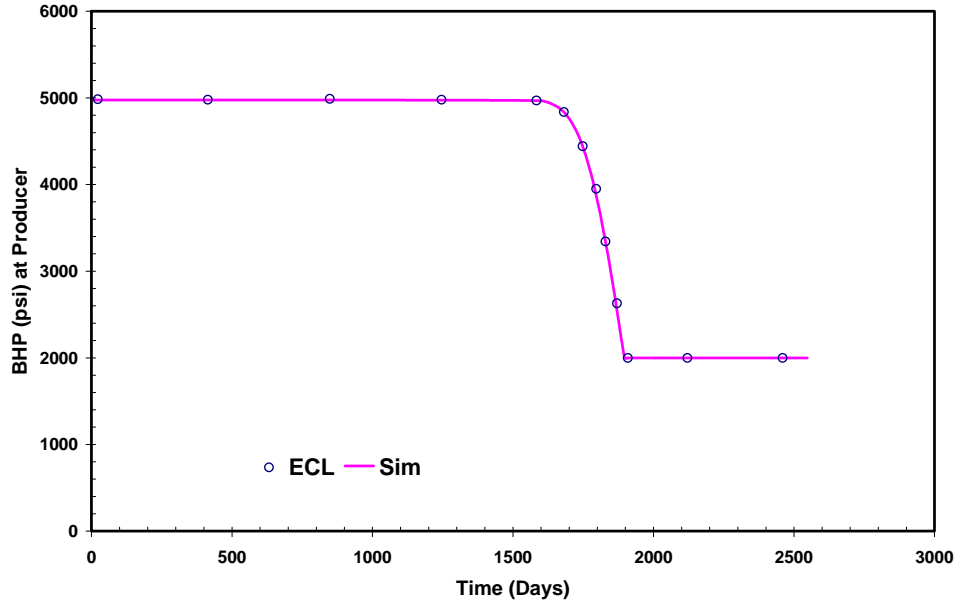


Fig. 5.3 – Comparison of Sim2D well bottomhole pressure at producer with ECL™ 100

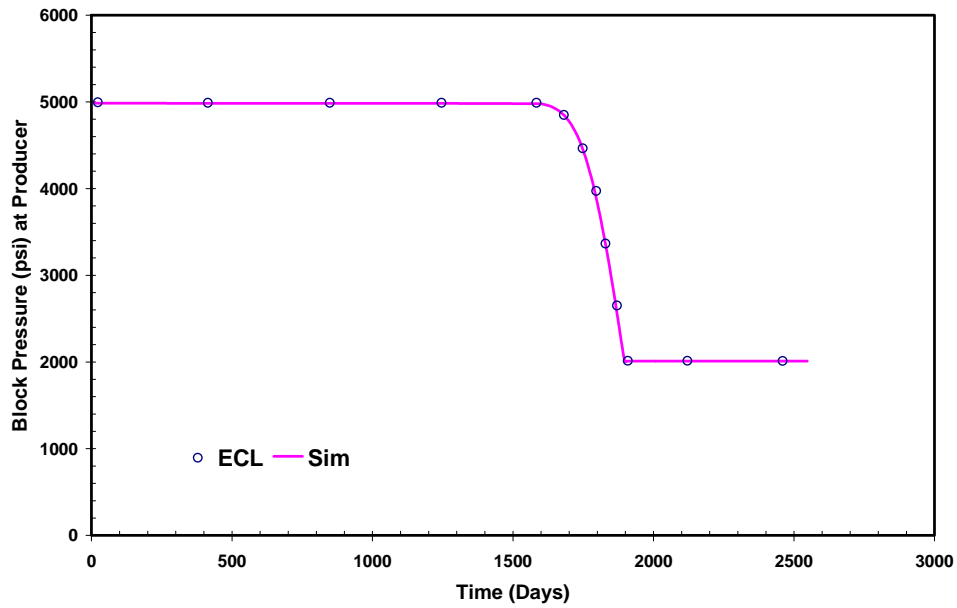


Fig. 5.4 – Comparison of Sim2D well block pressure at producer with ECL™ 100

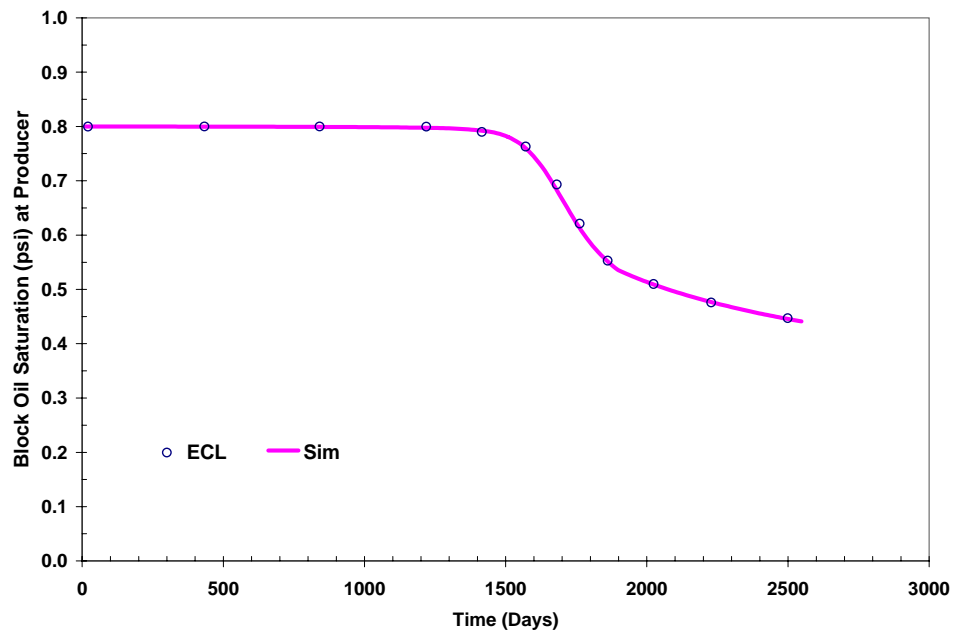


Fig. 5.5 – Comparison of Sim2D well block oil saturation at producer with ECL™ 100

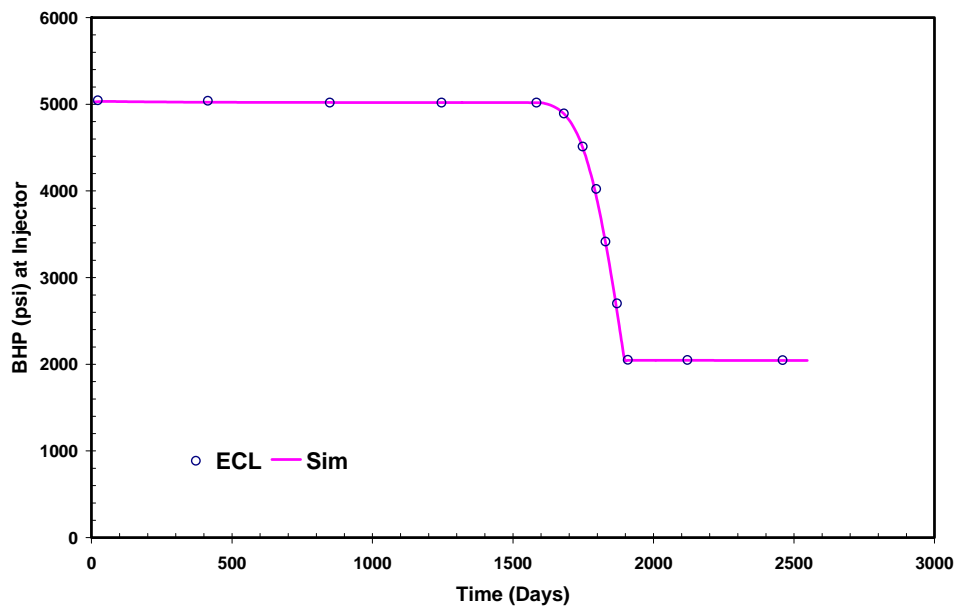


Fig. 5.6 – Comparison of Sim2D well bottomhole pressure at injector with ECL™ 100

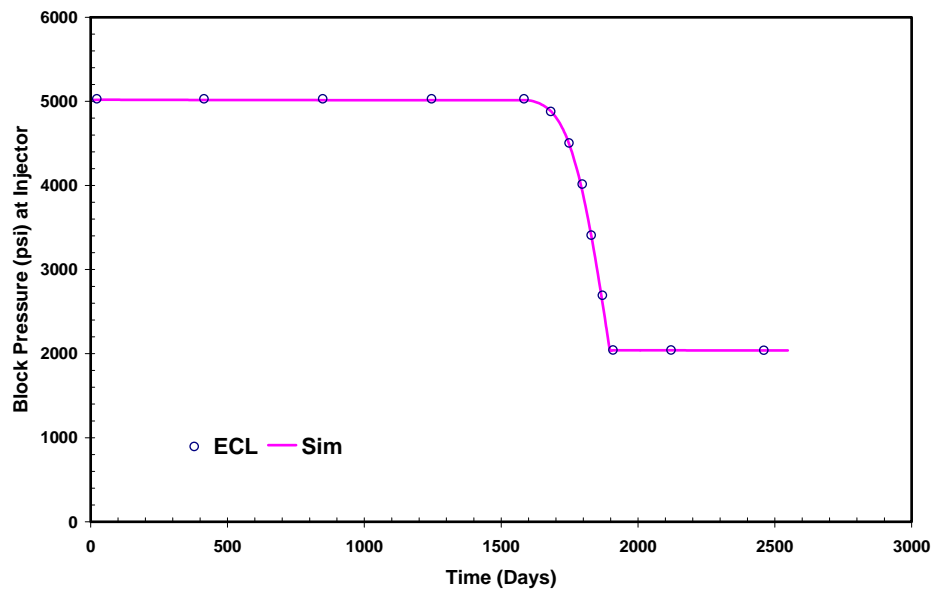


Fig. 5.7 – Comparison of Sim2D well block pressure at injector with ECL™ 100

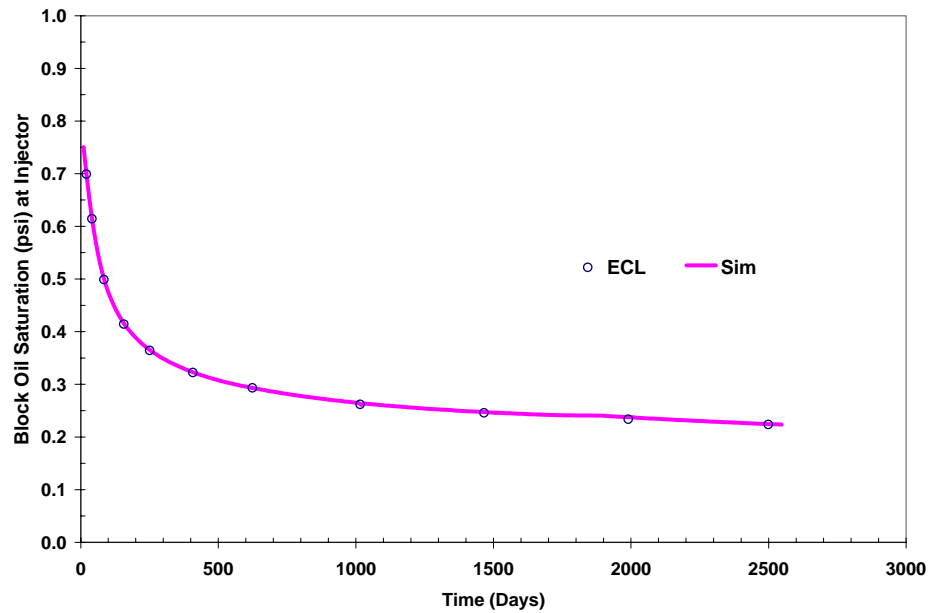


Fig. 5.8 – Comparison of Sim2D well block oil saturation at injector with ECL™ 100

5.2 Use of HGB Grid to Reduce the Grid Orientation Effect

Using the HGB scheme, one half of a five-spot model was simulated with a four-well, diagonal HGB grid (145 grid blocks), and one quarter five-spot pattern model were chosen to run a two-well, parallel HGB grid (85 grid blocks) as defined and shown in **Fig. 5.9**. The reservoir is assumed to be homogeneous and isotropic. Additional rock and fluid properties as well as simulation data that are relevant are given in **Table 5.3**. Other reservoir and simulation data is shown in **Fig. 5.1 and Table 5.4**. Different mobility ratios of 0.5, 1.0, 10.0 and 50.0 were used in these cases respectively. Porosity and permeability modifications were also performed on the boundary blocks so that the five-spot pattern can be simulated. The locations of the injectors and producers coincide with the locations of the block centers. The well model is also modified to reflect this. This is shown in **Fig. 5.10**.

Table 5.3- Data used for HGB pattern simulations

Rock Permeability, k	= 100 mD
Porosity, ϕ	= 0.20
Net Pay Thickness	= 10 ft
Producer-Injector Distance	= 825 ft
Production Rate, q_o	= 18 STB/D
Injection Rate, q_w	= 18 STB/D
Initial Water Saturation	= 0.20
Initial Oil Saturation	= 0.80
Initial Pressure	= 5000 psi
Area of Reservoir (Parallel)	= 15 acres
Area of Reservoir (Diagonal)	= 7.5 acres

Table 5.4 – 2-Phase PVT data (for M=0.5)

Pressure (psi)	Oil FVF (rcf/scf)	Oil Viscosity (cp)	Oil Compressibility (1/psi)	Water FVF (rcf/scf)	Water Viscosity (cp)	Water Compressibility (1/psi)
6014.7	1.0620	1.9300	2.51E-06	1.02E+00	1.0000	3.00E-06
5014.7	1.0647	1.6611	2.51E-06	1.02E+00	1.0000	3.00E-06
4014.7	1.0673	1.4297	2.51E-06	1.02E+00	1.0000	3.00E-06
3014.7	1.0700	1.2306	2.51E-06	1.03E+00	1.0000	3.00E-06
2514.7	1.0714	1.1417	2.51E-06	1.03E+00	1.0000	3.00E-06
2014.7	1.0727	1.0592	2.51E-06	1.03E+00	1.0000	3.00E-06
1514.7	1.0741	0.9827	2.51E-06	1.03E+00	1.0000	3.00E-06
1014.7	1.0754	0.9116	2.51E-06	1.03E+00	1.0000	3.00E-06
514.7	1.0768	0.8458	2.51E-06	1.03E+00	1.0000	3.00E-06
14.7	1.0781	0.7847	2.51E-06	1.03E+00	1.0000	3.00E-06

From **Fig. 5.11**, we can see that the parallel and diagonal HGB grid model give very similar results for both favorable and unfavorable mobility ratios cases that were run. This is because flow can progress in several different directions in the octagonal grid blocks. The result is that the differences between the parallel and diagonal orientation are greatly reduced. The parallel HGB grid always predicted a higher areal sweep efficiency than the diagonal HGB grid. At lower mobility ratios, the pore volume recovered is higher as the sweep mimics a piston-like displacement. Even so, the discrepancies between these two grids in HGB have a maximum relative difference of approximately 6% (**Table 5.5**) and it is believed to be caused by the presence of the square grid blocks.

Figs. 5.12-5.13 show the saturation distribution map for the parallel and diagonal HGB grid, respectively. The movement of the saturation front is faster when the mobility ratio increases. This is due to the fact that the displacing fluid is moving at a much higher velocity than oil, the displaced fluid. Fingering of the displacing fluid also results in faster breakthrough times.

This result shows that HGB can reduce significantly the grid orientation effect by reducing the rotational variance in the model and hence the differences in results between the parallel and diagonal HGB grids.

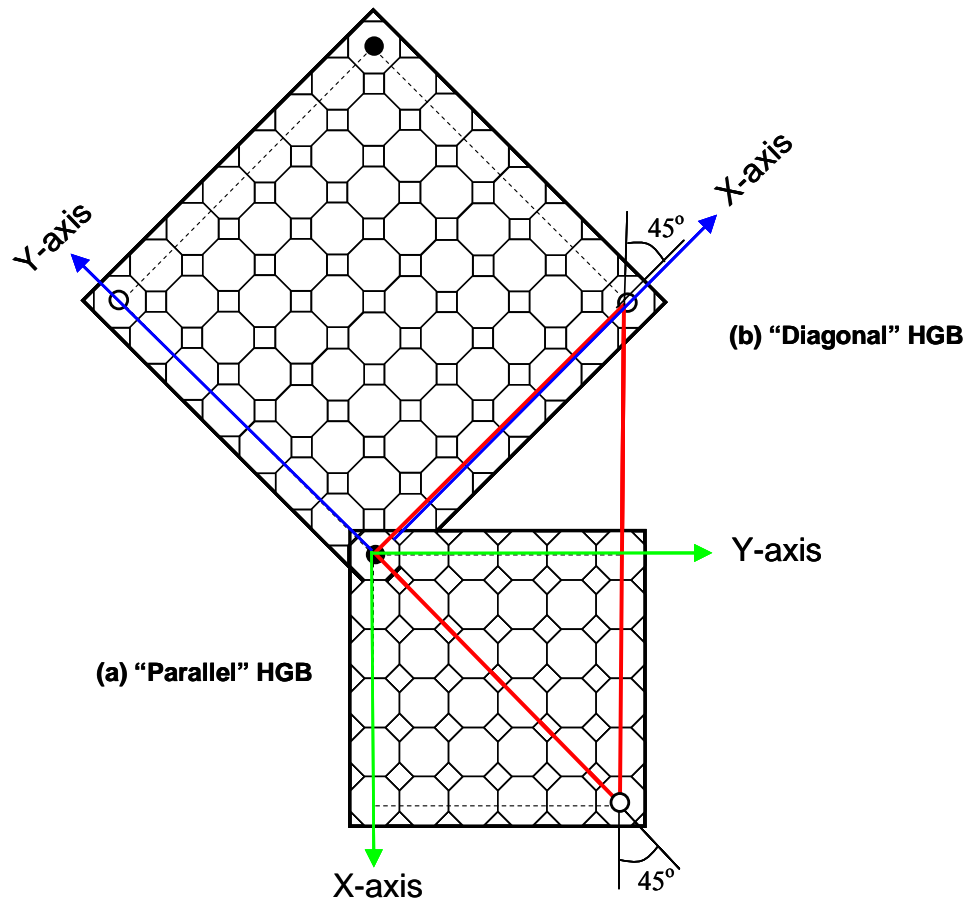


Fig. 5.9 – (a) Parallel and (b) diagonal grid orientation in HGB grid

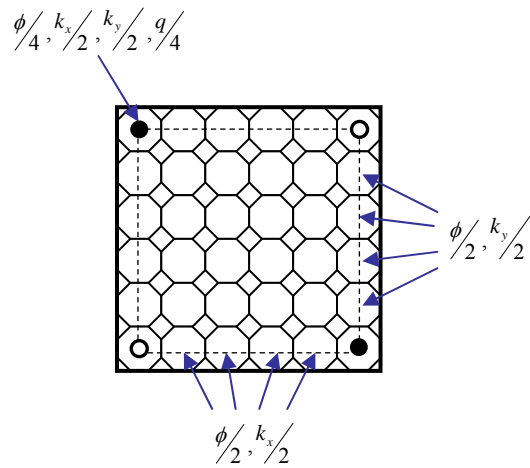


Fig. 5.10 – Porosity, permeability and well model modifications

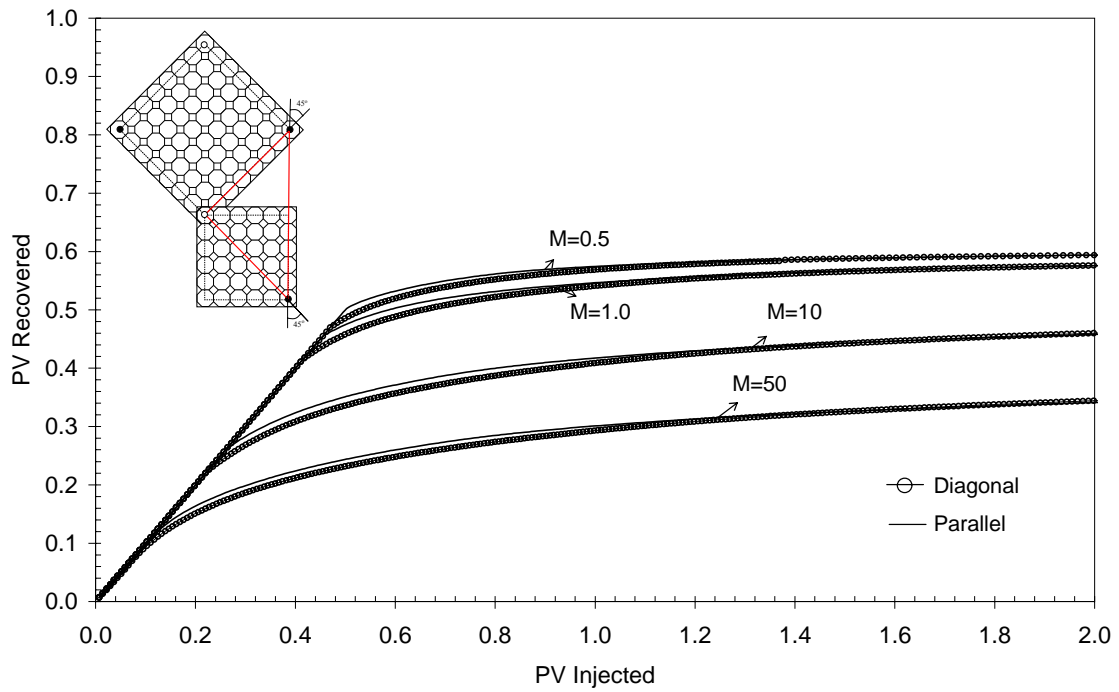


Fig. 5.11 – Influence of mobility ratios on the predicted performance of HGB grid

Table 5.5 – Relative difference between parallel and diagonal models of various mobility ratios for HGB grid models in Fig. 5.10

M	Relative Difference (%)
50.0	5.92
10.0	4.59
1.0	2.24
0.5	1.11

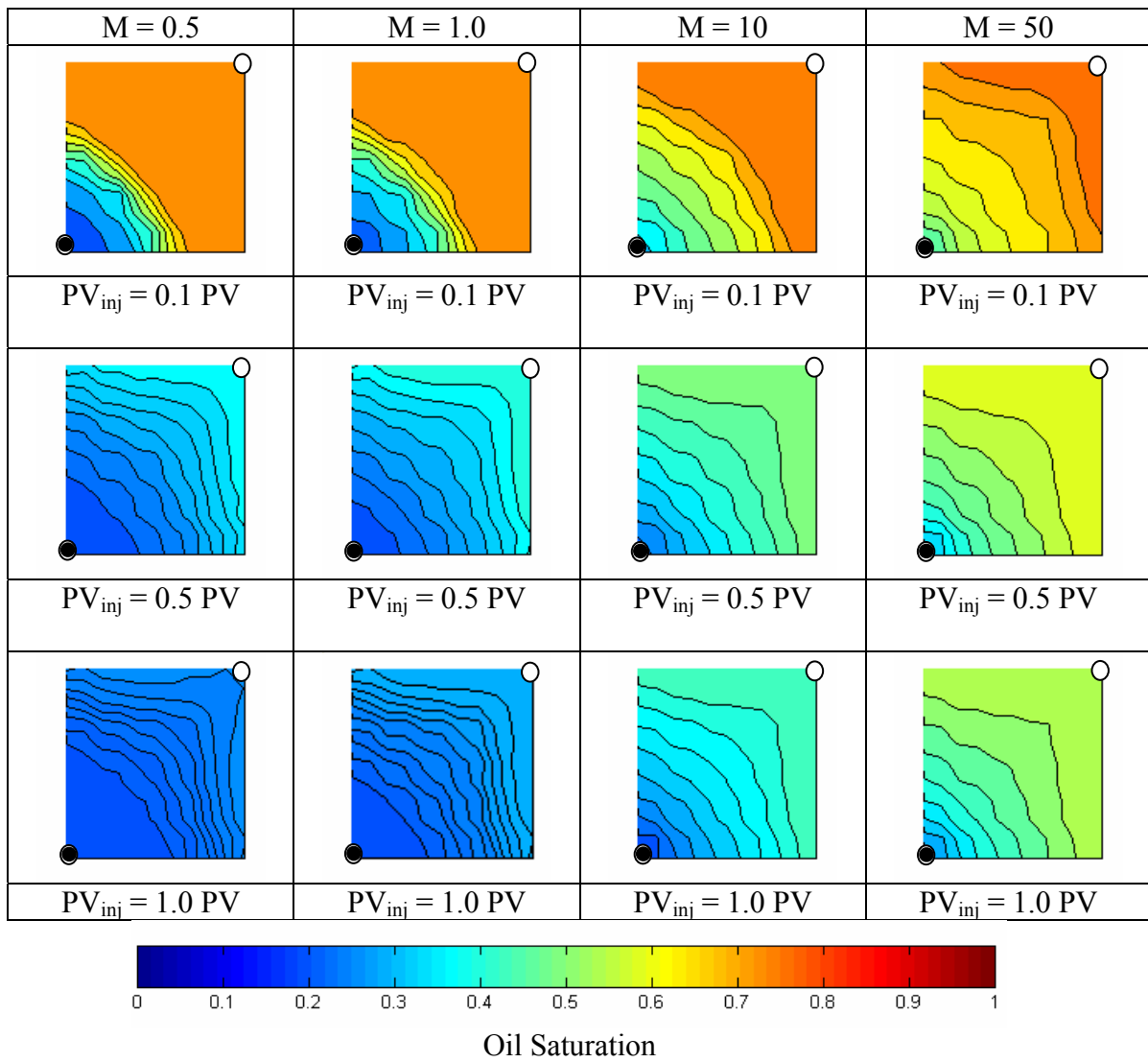


Fig. 5.12 – Saturation distribution map for parallel HGB grid as shown in Fig. 5.9 (a) at various mobility ratios

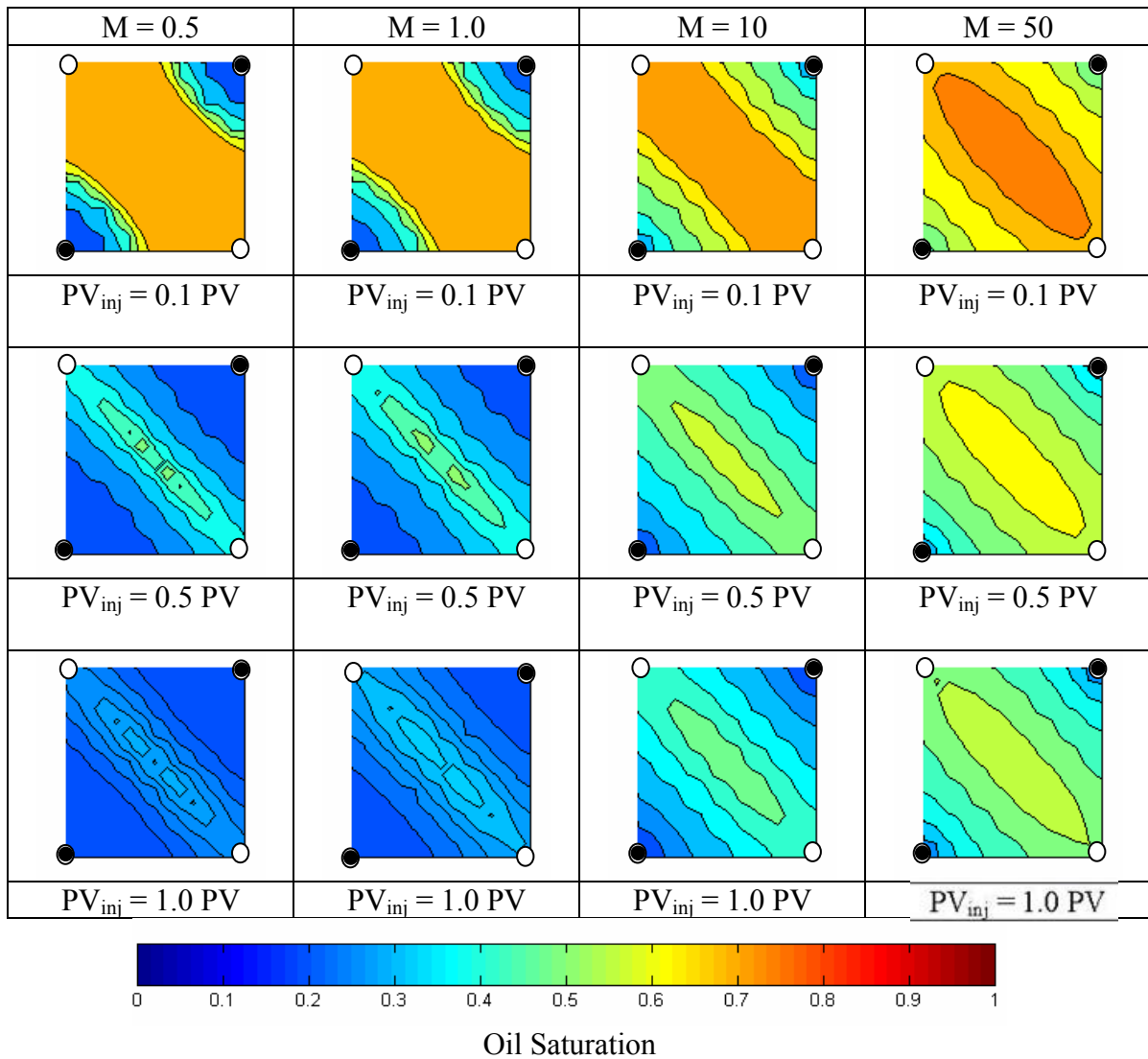


Fig. 5.13 – Saturation distribution map for diagonal HGB grid as shown in Fig. 5.9 (b) at various mobility ratios

5.2.1 HGB Sensitivity

In the previous example, we have shown that the HGB grid is less sensitive to grid orientation than conventional Cartesian grid, especially at unfavorable mobility ratios. The next point for investigation is the sensitivity of the HGB grid to the grid dimension, or in other words, its sensitivity to the number of grid subdivisions in the model. We have seen that in Cartesian grids, the parallel grid is more sensitive to the effect of grid size than the diagonal grid. We will also see how the performance of the HGB grid compares to the Cartesian grid at a similar number of grid blocks.

Four different sets of grid numbers were run in HGB (diagonal HGB and parallel HGB) and the results are compared to similar models run in Cartesian grid (diagonal Cartesian and parallel Cartesian). Since the size of each grid block in the HGB model is dependent upon its shape, i.e. whether it is octagonal, or rectangular, an average area per grid block is calculated for each set of runs. The closest possible average area per grid block and the number of grid blocks are then applied to the corresponding Cartesian grid. The grid numbers and grid block dimensions are shown in **Tables 5.6-5.7** for the HGB and Cartesian grid respectively. For a given set of grid dimension, HGB and its Cartesian grid counterpart have the same pore volume and well locations.

Furthermore, since we have shown that the grid orientation effect can be reduced by refining the grid at a lower mobility ratio (as shown in Chapter IV), all these cases are run at $M = 0.5$. The reservoir and rock properties are similar to those shown in Fig. 1 and **Tables 5.3-5.4**.

Figs. 5.14 through **5.17** show the calculated performance for the HGB grids and their corresponding Cartesian grids. Results using the HGB grids are always in between those calculated on Cartesian grids.

However, the differences between the diagonal HGB and parallel HGB are reduced as the number of grid blocks in the model was increased. In fact, the performances of the parallel and diagonal HGB grid models tend to converge as the grid spacing is refined.

Table 5.6 – Averaged area per grid block for the HGB grid

Number of HGB Grid Blocks	Averaged area per grid block			
	Parallel		Diagonal	
	ft ²	ac	ft ²	ac
50	6806.25	0.156	13612.5	0.313
98	3472.58	0.080	6945.15	0.159
200	1701.56	0.039	3403.13	0.078
392	868.14	0.020	1736.29	0.040
800	425.39	0.010	850.78	0.020

Table 5.7 – Averaged area per grid block for the Cartesian grid

Number of Cartesian Grid Blocks	Parallel		Diagonal	
	Averaged Area (ac)	$\Delta x = \Delta y$ (ft)	Averaged Area (ac)	$\Delta x = \Delta y$ (ft)
49	0.319	117.8571	0.159	83.3376
100	0.156	82.5000	0.078	58.3363
196	0.080	58.9286	0.040	41.6688
400	0.039	41.2500	0.020	29.1682
784	0.020	29.4643	0.010	20.8344

Likewise, the differences between diagonal Cartesian and parallel Cartesian are reduced as a smaller grid dimension is used. As we have shown in Chapter IV, at low mobility ratios, i.e., $M = 0.5$, the diagonal Cartesian grid is insensitive to the number of grid blocks in the model. Contrarily, when more refined grid blocks are used in the parallel Cartesian grid, the oil recovery would increase and the results would converge to a single recovery curve after an increase in a certain number of grid blocks.

The results between the HGB and Cartesian grid models are summarized in **Table 5.8**. For reasons mentioned earlier, these two models are compared at a similar averaged area per grid block. As the number of grid blocks are increased and the size of grid blocks are reduced, we can see that both models give a smaller relative difference in pore volume recovered between the parallel and diagonal grid than when coarser grids are used. More importantly, the HGB grid performs better by consistently giving a smaller relative difference in pore volume recovered compared to the Cartesian grid at similar averaged area per grid block for all the cases studied. This indicates that the

HGB is more effective in reducing the grid orientation error than the conventional Cartesian grid.

Results of simulation runs at $M=0.5$ are summarized in **Figs. 5.18-5.19** for the parallel and diagonal HGB grid, respectively. Performance of the parallel HGB grid is not sensitive to the number of grid blocks in the model, as all the four models give similar results. On the other hand, oil recovery predicted by the diagonal HGB model increased as the number of grid blocks increased, and the results converged when the number of grid blocks is at 200 and higher.

Table 5.8 – Relative difference between parallel and diagonal grid for both HGB and Cartesian grids at $M = 0.5$

	Averaged area per grid block (ac)		Relative Difference in Pore Volume Recovered (%)
	Diagonal	Parallel	
HGB Grid	0.159 (98)*	0.156 (50)*	6.0
	0.078 (200)	0.080 (98)	4.5
	0.040 (392)	0.039 (200)	3.3
	0.020 (800)	0.020 (392)	2.2
Cartesian Grid	0.156 (100)	0.159 (49)	17.0
	0.080 (196)	0.078 (100)	13.0
	0.039 (400)	0.040 (196)	9.3
	0.020 (784)	0.020 (400)	7.9

* number in brackets indicates the number of grid blocks used

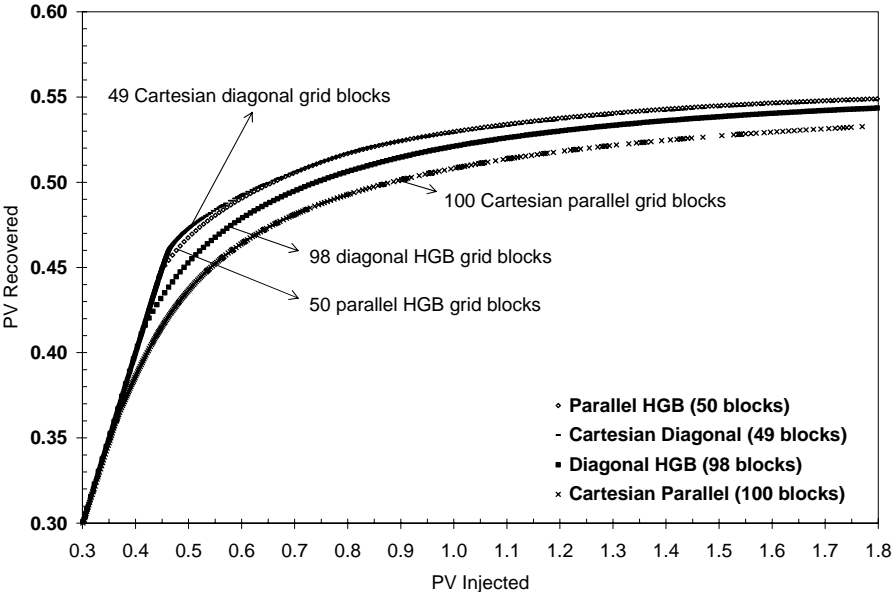


Fig. 5.14 – Comparison between HGB grid (50 and 98 grid blocks) and Cartesian grid (49 and 100 grid blocks) at $M = 0.5$

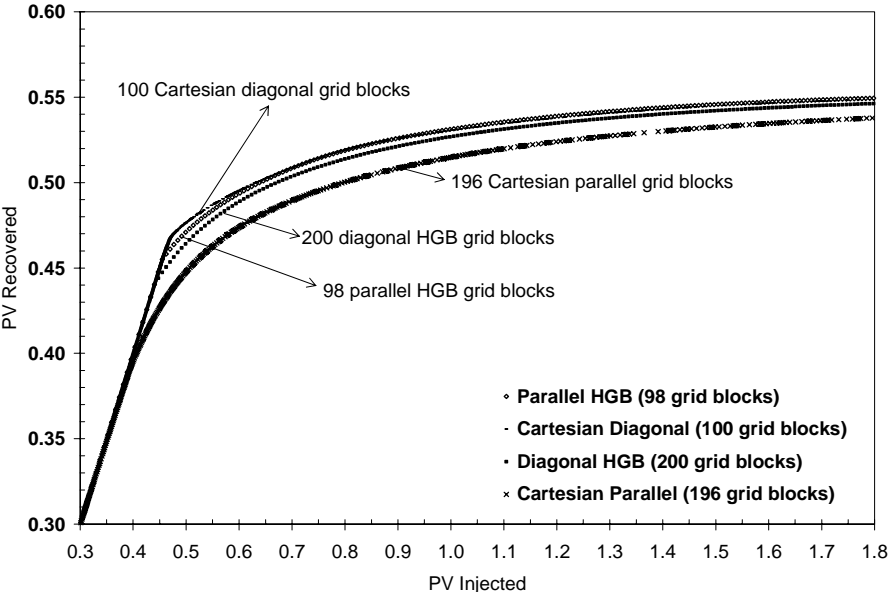


Fig. 5.15 – Comparison between HGB grid (98 and 200 grid blocks) and Cartesian grid (100 and 196 grid blocks) at $M = 0.5$

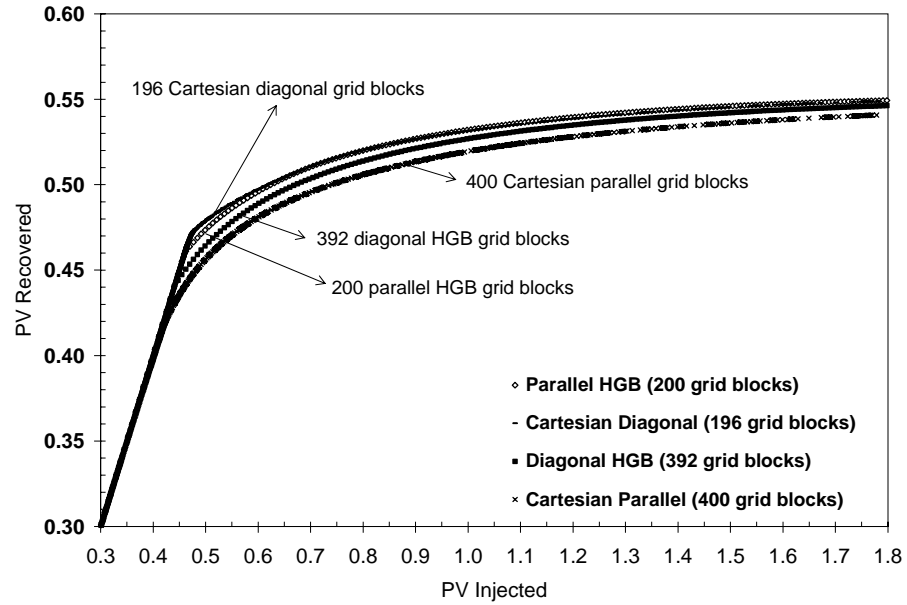


Fig. 5.16 – Comparison between HGB grid (200 and 392 grid blocks) and Cartesian grid (196 and 400 grid blocks) at $M = 0.5$

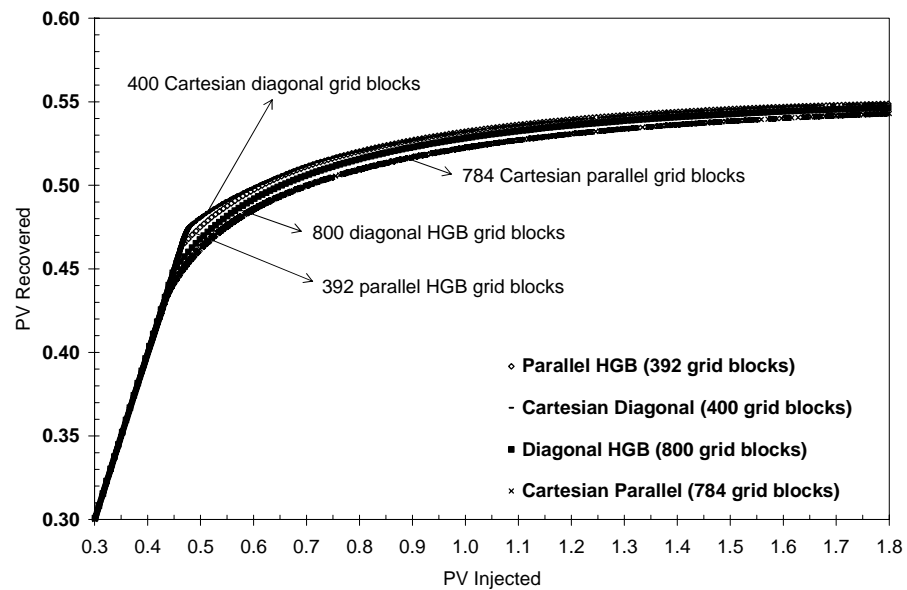


Fig. 5.17 – Comparison between HGB grid (392 and 800 grid blocks) and Cartesian grid (400 and 784 grid blocks) at $M = 0.5$

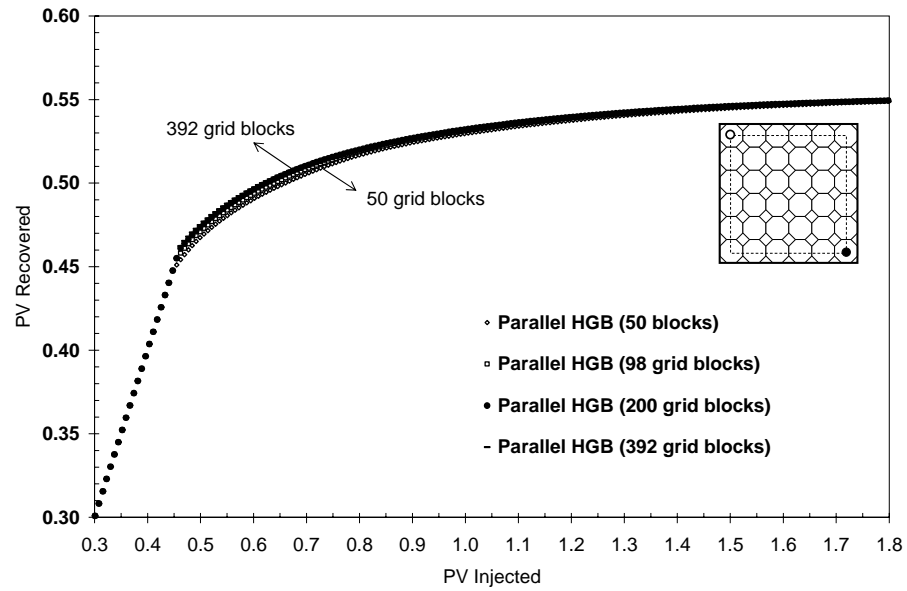


Fig. 5.18 – Effect of grid spacing on parallel HGB grid for $M=0.5$

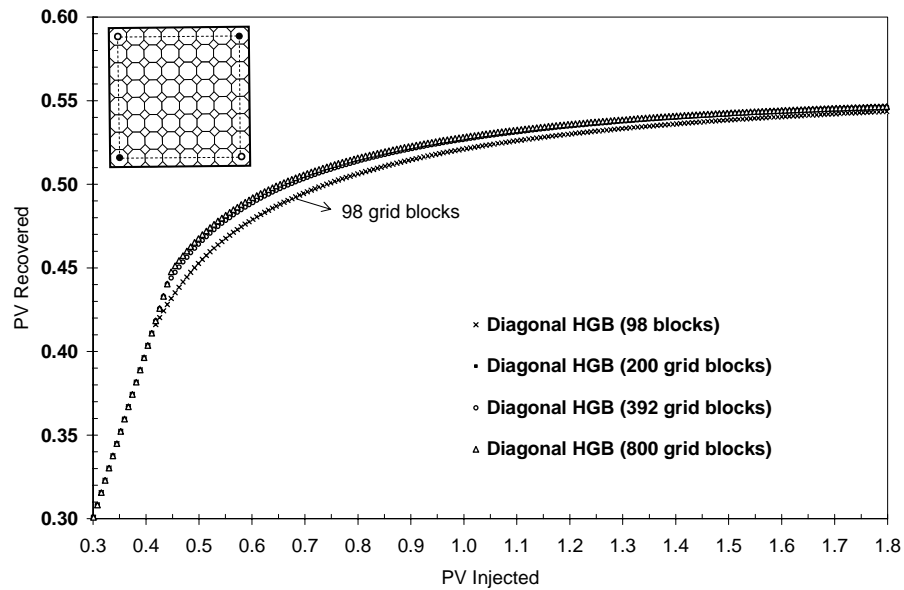


Fig. 5.19 – Effect of grid spacing on diagonal HGB grid for $M=0.5$

CHAPTER VI

CONCLUSIONS

We have shown the effect of grid orientation in conventional Cartesian parallel and diagonal grid. Also, we have successfully developed the HGB grid system and we have shown that HGB is more effective in reducing the grid orientation effect than Cartesian grid. This is attributed to the increase of flow connections in the octagonal grid blocks in HGB grid. On the other hand, the increase in flow connections also means that there would be more terms to solve and hence this would increase the computational time compared to Cartesian grid. Moreover, the construction and numbering or indexing of the HGB grid is not as intuitive as the Cartesian grid. Extension of the HGB grid to a 3-D model would also be a challenging task. However, we would recommend the use of HGB grid for simulations of displacement problems especially at unfavorable mobility ratios.

The following conclusions can be derived from this study:

1. Grid orientation effect was observed in rectangular Cartesian grid models even at isotropic and homogeneous reservoir of $M = 1.0$.
2. Based on this study, grid refinement can help to reduce the grid orientation effect in rectangular Cartesian grid models when there are favorable mobility ratios, i.e. $M=1.0$ or less. However, at an unfavorable mobility ratio of $M=10.0$, it is found that neither parallel nor diagonal orientation can be used reliably. This is because as the number of grid blocks is increased, the performance of diagonal and parallel models actually diverges for the grid spacings investigated in this study.

3. With the increased number of connections in the octagonal grid blocks in HGB grid compared to Cartesian grid, HGB is able to reduce the grid orientation effect even for unfavorable mobility ratio displacement problems ($M = 10.0$), with maximum relative difference of 6% in pore volume recovered between parallel and diagonal HGB grid models for all the cases run. However, the grid orientation effect in HGB model is believed to be caused by the presence of the square grid blocks.
4. Contrary to the Cartesian parallel grid, HGB parallel grid is less sensitive to the number of grid blocks in the model compared to the HGB diagonal grid for $M = 0.5$. Also, at a favorable mobility ratio of $M = 0.5$, the performance of the parallel and diagonal HGB grid models converged as the number of grid blocks is increased.
5. The HGB grid performs better by consistently giving a smaller relative difference between HGB parallel grid and HGB diagonal grid in pore volume recovered (6.0, 4.5, 3.3, and 2.2%) compared to the relative difference between Cartesian parallel grid and Cartesian diagonal grid in pore volume recovered (17.0, 13.0, 9.3, 7.9%) at similar averaged area per grid block for all the four comparison cases studied. This indicates that the HGB is more effective in reducing the grid orientation error than the conventional Cartesian grid.

Recommendations for future work:

1. The numerical solutions obtained from developed simulator and commercial simulators should be compared with analytical solutions. The analytical solution should be used as a reference to investigate the accuracy of the numerical result.
2. In Sim2D, we only have one set of grid block configuration using the combination of octagonal and rectangular grid blocks. In addition, the octagonal

and rectangular grid blocks are regular polygons. It would be interesting to vary these into irregular polygons.

3. The treatment of anisotropic reservoirs has yet to be addressed. A full permeability tensor can be modeled, though there are several challenging issues related to their implementation in a simulator such as the treatment of wells and averaging of permeability at each connection.
4. The range of the test problems need to be extended to multiphase and heterogeneous reservoirs.

REFERENCES

1. Pedrosa, O.A. and Aziz, K.: "Use of Hybrid Grid in Reservoir Simulation", paper SPE 13507, presented at the 1985 SPE Symposium on Reservoir Simulation, Dallas, Texas, 10-13 February.
2. Nacul, E.C.: *Use of Domain Decomposition and Local Grid Refinement in Reservoir Simulation*, Ph.D. Dissertation, Stanford University, Palo Alto, California (March 1991).
3. Ding, Y. and Lemonnier, P.: "Use of Corner Point Geometry in Reservoir Simulation", paper SPE 29933 presented at the 1995 International Meeting on Petroleum Engineering held in Beijing, PR China, 14 -17 Nov..
4. Peaceman, D.W.: "Calculation of Transmissibilities of Grid blocks Defined by Arbitrary Corner Point Geometry", paper SPE 37306 presented at the 1996 SPE Reservoir Simulation Symposium, New Orleans, Louisiana, 31 Jan - 3 Feb.
5. Heinemann, Z.E. and Brand, C.W.: "Modeling Reservoir Geometry with Irregular Grids", paper 18412 presented at the 1989 SPE Symposium on Reservoir Simulation, Houston, Texas, 6-8 February.
6. Palagi, C.L. and Aziz, K.: "Use of Voronoi Grid in Reservoir Simulation", paper SPE 22889 presented at the 1991 SPE Reservoir Simulation Symposium held in New Orleans, Louisiana, 31 Jan - 3 Feb.
7. Gunasekera, D., Cox, J. and Lindsey, P.: "The Generation and Application of K-Orthogonal Grid Systems", paper SPE 37998 presented at the 1997 SPE Symposium on Reservoir Simulation, Dallas, Texas, 8-11 June.
8. Kocberber, S.: "An Automatic, Unstructured Control Volume Generation System for Geologically Complex Reservoirs", paper SPE 38001 presented at the 1997 SPE Symposium on Reservoir Simulation, Dallas, Texas, 8-11 June.

9. Aavatsmark, I., Barkve, T., Boe, O., and Mannseth, T.: “Discretization on Nonorthogonal, Quadrilateral Grids for Inhomogeneous, Anisotropic Media”, *J. Computational Physics* (1996) **127**, 2-14.
10. Verma, S. and Aziz, K.: “A Control Volume Scheme for Flexible Grids in Reservoir Simulation”, paper SPE 37999 presented at the 1997 SPE Symposium on Reservoir Simulation, Dallas, Texas, 8-11 June.
11. Todd, M.R., O’Dell, P.M. and Hirasaki, G.J.: “Methods for Increased Accuracy in Numerical Simulation”, paper SPE 3516 presented at the 1971 PE Annual Fall Meeting, New Orleans, Louisiana, 3-6 October.
12. Holloway, C.C., Thomas, L.K., and Pierson, R.G.: “Reduction of Grid Orientation Effects in Reservoir Simulation”, paper SPE 5522 presented at the 1975 Annual Fall Meeting of the Society of Petroleum Engineers of AIME, Dallas, Texas, 28 September – 1 October.
13. Frauenthal, J.C., Rolaud, B. and Towler, B.F.: “Reduction of Grid-Orientation Effects in Reservoir Simulation with Generalized Upstream Weighting”, paper SPE 11593 presented at the 1983 SPE Symposium on Reservoir Simulation, San Francisco, California, 15-18 November.
14. Vinsome, P.K. and Au A.K.: “One Approach to the Grid Orientation Problem in Reservoir Simulation,” paper SPE 8247 presented at the 1979 Annual Fall Technical Conference and Exhibition of the Society of Petroleum Engineers of AIME, Los Angeles, California, 23-26 September.
15. Yanosik, J.L. and McCracken, T.A.: “A Nine-Point, Finite-Difference Simulator for Realistic Prediction of Adverse Mobility Ratio Displacements”, paper SPE 5734 presented at the 1976 SPE-AIME Symposium of Numerical Simulation of Reservoir Performance, Los Angeles, California, 19-20 February.
16. Ko, S.C.M. and Au, A.D.K.: “A Weighted Nine-Point Finite-Difference Scheme for Eliminating the Grid Orientation Effect in Numerical Reservoir Simulation”, paper SPE 8248 presented at the 1979 SPE Annual Fall Meeting, Las Vegas, Nevada, 23-26 September.

17. Coats, K.H. and Modine, A.D.: "A Consistent Method for Calculating Transmissibilities in Nine-Point Difference Equations", paper SPE 12248 presented at the 1983 Reservoir Simulation Symposium, San Francisco, California, 15-18 November.
18. Shah, P.C.: "A Nine-Point Finite Difference Operator for Reduction of the Grid Orientation Effect," paper SPE 12251 presented at the 1983 Reservoir Simulation Symposium, San Francisco, California, 15-18 November.
19. Ostebo, B. and Kazemi, H: "Mixed Five-Point/Nine-Point Finite-Difference Formulation of Multiphase Flow in Petroleum Reservoirs", paper SPE 21227 presented at the 1991 Reservoir Simulation Symposium, Anaheim, California, 17-20 February.
20. Shiralkar, G.S. and Stephenson, R.E.: "A General Formulation for Simulating Physical Dispersion and a New Nine-Point Scheme", paper SPE 16975 presented at the 1987 Annual Technical Conference and Exhibition, Dallas, Texas, 27-30 September.
21. Vinsome, P.K. and Au A.K.: "One Approach to the Grid Orientation Problem in Reservoir Simulation," paper SPE 8247 presented at the 1979 Annual Fall Technical Conference and Exhibition of the Society of Petroleum Engineers of AIME, Los Angeles, California, 23-26 September.
22. Brand, C.W., Heinemann, J.E., and Aziz, K.: "The Grid Orientation Effect in Reservoir Simulation", paper SPE 21228 presented at the 1991 SPE Symposium on Reservoir Simulation, Anaheim, California, 17-20 February.
23. Chen, W.H. and Durlafsky, L.J.: "Minimization of Grid Orientation Effects Through Use of Higher-Order Finite Difference Methods", paper SPE 22887 presented at the 1991 Annual Technical Conference and Exhibition, Dallas, Texas, 6-9 October.
24. Pinto A.C.C. and Correa, A.C.F.: "High-Resolution Schemes for Conservation Laws: Applications to Reservoir Engineering," paper SPE 24262 presented at the

- 1992 European Petroleum Computer Conference, Stavanger, Norway, 25-27 May.
25. Wolcott, D.S., Kazemi, H., and Dean, R.H.: "A Practical Method for Minimizing the Grid Orientation Effect in Reservoir Simulation", paper SPE 36723 presented at the 1996 Annual Technical Conference and Exhibition, Denver, Colorado, 6-9 October.
 26. Shin, D. and Merchant, A.R.: "Higher-Order Flux Update Function Method for Reduction of Numerical Dispersion and Grid Orientation Effect in Reservoir Simulation", paper SPE 25600 presented at the 1993 Middle East Oil Technical Conference and Exhibition, Bahrain, 3-6 April.
 27. Pruess, K. and Bodvarsson, G.S.: "A Seven-Point Finite Difference Method for Improved Grid Orientation Performance in Pattern Steamfloods," paper SPE 12252 presented at the 1983 Reservoir Simulation Symposium, San Francisco, California, 15-18 November.
 28. Heinemann, Z.E. and Brand, C.W.: "Modeling Reservoir Geometry with Irregular Grids", paper 18412 presented at the 1989 SPE Symposium on Reservoir Simulation, Houston, Texas, 6-8 February.
 29. Mattax, C.C. and Dalton R.L.: *Reservoir Simulation*, Monograph #13, SPE, Richardson, Texas (1990).
 30. Crichlow, H.B.: *Modern Reservoir Engineering – A Simulation Approach*, Prentice Hall, Englewood Cliffs, New Jersey (1977).
 31. Peaceman, D.W.: *Fundamentals of Numerical Reservoir Simulation*, Elsevier, New York (1977).
 32. Aziz K. and Settari A.: *Petroleum Reservoir Simulation*, Elsevier, New York (1979).
 33. Fanchi, J.R.: *Principles of Applied Reservoir Simulation*, Gulf Publishing Company, Houston, Texas (1997).
 34. Wattenbarger, R.A.: *PETE 603 Class Notes*, Department of Petroleum Engineering, Texas A&M University, College Station, Texas (2003).

35. Peaceman, D.W.: “Interpretation of Well-Block Pressures in Numerical Reservoir Simulation”, paper SPE 6893 presented at the 1977 SPE-AIME Annual Fall Technical Conference and Exhibition, Denver, Colorado, 9-12 Oct.
36. Peaceman, D.W.: “Interpretation of Well-Block Pressures in Numerical Reservoir Simulation with Nonsquare Grid Blocks and Anisotropic Permeability”, paper SPE 10528 presented at the 1982 SPE Reservoir Simulation Symposium New Orleans, Louisiana, 31 Jan - 3 Feb.
37. Palagi, C.L.: *Generation and Application of Voronoi Grid to Model Flow in Heterogeneous Reservoirs*, Ph.D. Dissertation, Stanford University, Palo Alto, California, (May 1992).

APPENDIX A

Sim2D VB PROGRAM APPLICATION


The main window consists of 5 tab strips for data entries. The main menus are shown in **Figs. A.1-A.5**, namely Grid Builder, Reservoir Description, Initial Condition, Well Data and Numerical Method. Data file of the PVT tables can be uploaded in .txt form, while the results are output to a MS Excel spreadsheet, although real-time plots are generated while the program is running, as shown in **Figs. A.6-A.16**. Also, the “time step skip” for the output is entered by the user.

Form1

Grid Builder | Reservoir Description | Initial Condition | Well Data | Numerical Method

Grid Model Selection Hybrid Grid-Block

Hybrid Grid-Block (HGB) Model Only



Length of Octagon Side (ft)

Number of Octagons:

X-direction, NX

Y-direction, NY

Constraints NY <=NX; NX, NY minimum = 3

Total Number of Nodes

Hexagonal Grid-Block Model Only
(Future Development)

Length of Hexagon Side (ft)


Number of Hexagons:

X-direction, NX

Y-direction, NY

Total Number of Nodes

Cartesian Grid Model Only



Number of Grid Blocks:

X-direction, NX DX

Y-direction, NY DY

Block Widths:

Enter dx/dy separated by spaces (5 5 10 5 ...)

Submit Update Close Clear

Fig. A.1 – Grid Builder window

Form1

Grid Builder | **Reservoir Description** | **Initial Condition** | **Well Data** | **Numerical Method**

Grid Block Properties - Entire Reservoir:

Porosity (fraction)

Permeability (mD)

Compressibility(1/psi)

Height of Reservoir (ft)

Sim2D Assumptions:

- Constant fluid compressibility and viscosity
- Homogeneous reservoir
- Isotropic reservoir
- No gravity effects
- No capillary effects

Fig. A.2 – Reservoir Description window

Form1

Grid Builder | Reservoir Description | Initial Condition | Well Data | Numerical Method

PVT Properties

Fluid Model

PVT Data File Upload (Text)

Initial Condition

Initial Pressure (psi)

Connate Water Saturation, Swc

Initial Oil Saturation, Soi

Fig. A.3 – Initial Condition window

Form1

Grid Builder | Reservoir Description | Initial Condition | **Well Data** | Numerical Method

Well Data

Number of Wells, nwell:

Constant Rate Only

	Well Location	Well Type	Qo (scf/D)	Qw (scf/D)	Qg (scf/D)
1	43	Constant Rate Producer	100		
2	125	Constant Rate Injector		-100	
3					
4					
5					
6					

Constant Pressure Only

	Pwf (psi)	rw (ft)	skin	Min BHP (psi)	Producer/Injector
1		.33	0	2000	Prod
2		.33	0	2000	Inj
3					
4					
5					
6					

Submit Close Clear

Fig. A.4 – Well Data window

Form1

Grid Builder | Reservoir Description | Initial Condition | Well Data | Numerical Method

Numerical Method Control

Time Step Size (days)

End of Simulation Run (days)

Matrix Solver Control

Maximum Error (psi)

Maximum Iteration

Maximum Sw Change Per Time Step (fraction)

Maximum Time Step Cut (day)

Output Control

Print Step Skip #

Output File (MS Excel)

Fig. A.5 – Numerical Method window

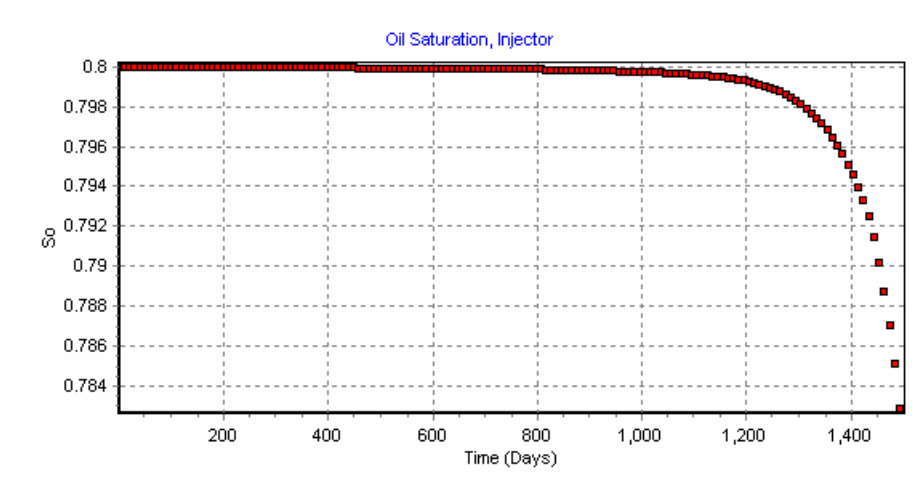


Fig. A.6 – Example showing oil saturation plot of grid block containing injector

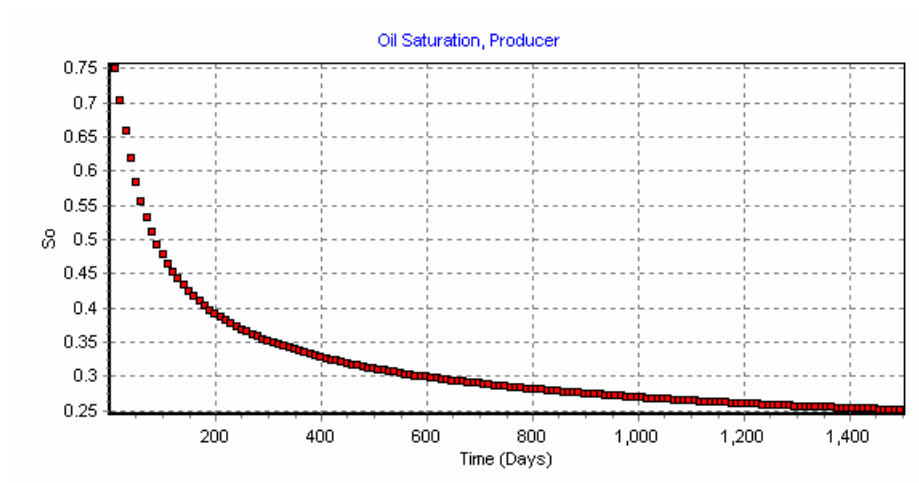


Fig. A.7 – Example showing oil saturation plot of grid block containing producer

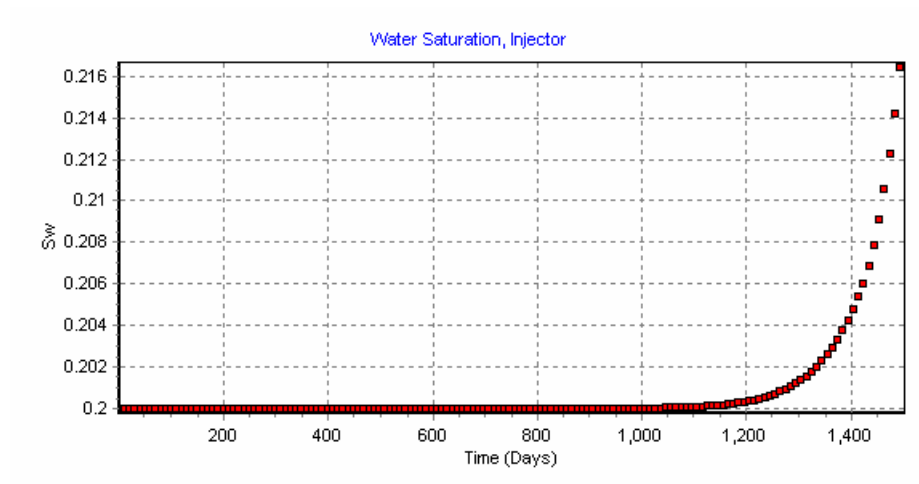


Fig. A.8 – Example showing water saturation plot of grid block containing injector

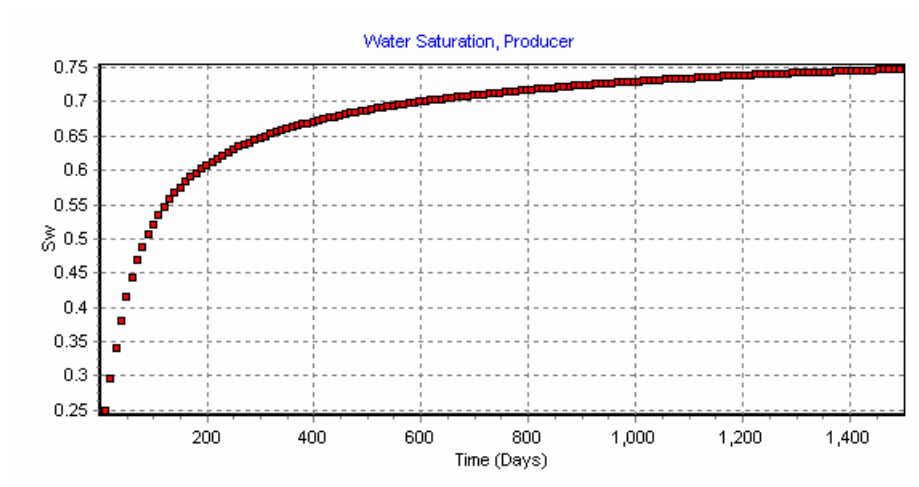


Fig. A.9 – Example showing water saturation of grid block containing producer

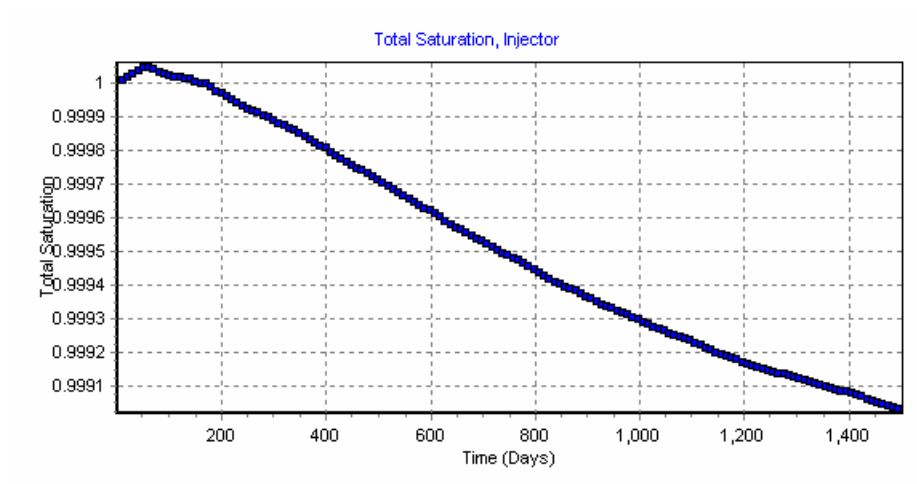


Fig. A.10 – Example showing total saturation plot of grid block containing injector

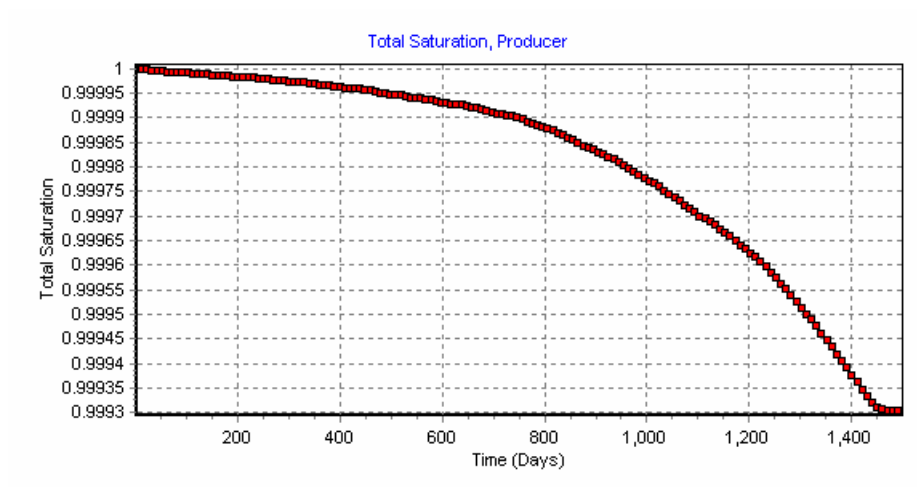


Fig. A.11 – Example showing total saturation plot of grid block containing producer

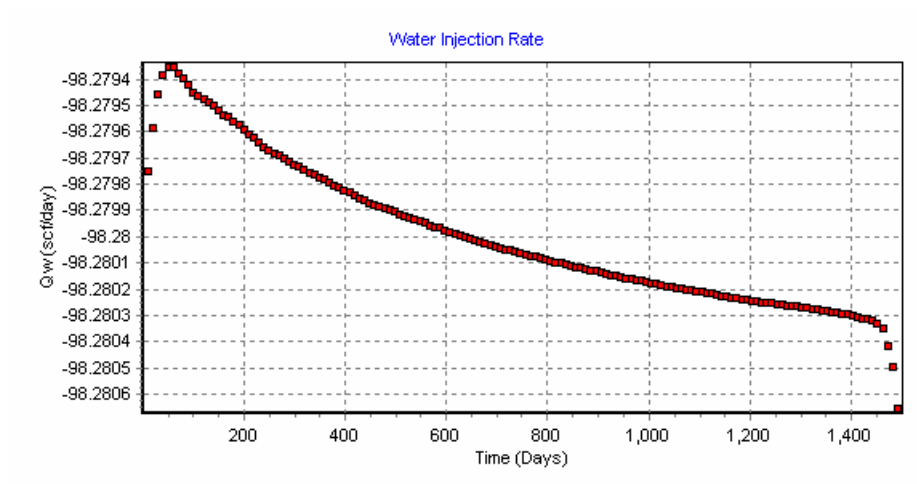


Fig. A.12 – Example showing water injection rate plot

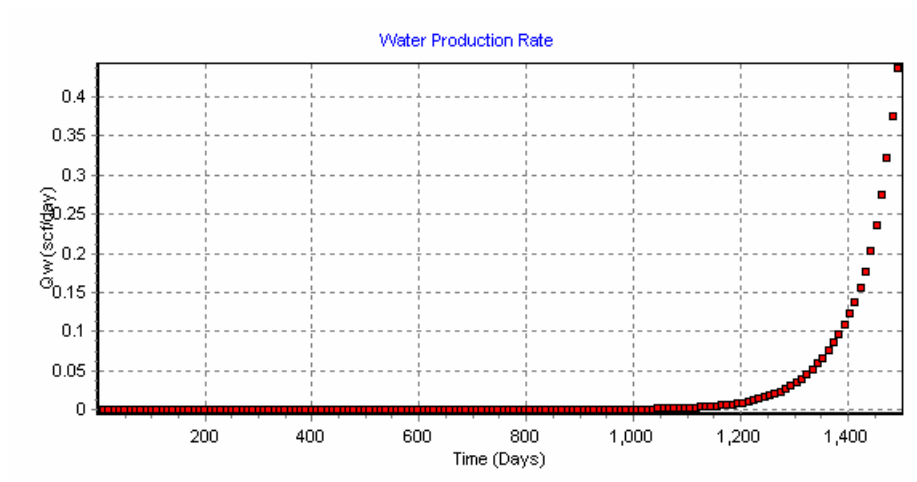


Fig. A.13 – Example showing water production rate plot

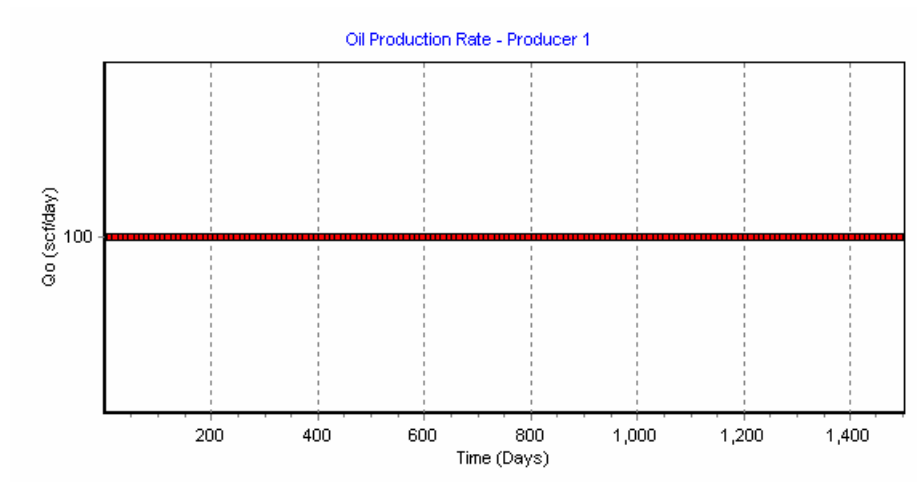


Fig. A.14 – Example showing oil production rate plot in a constant production rate case

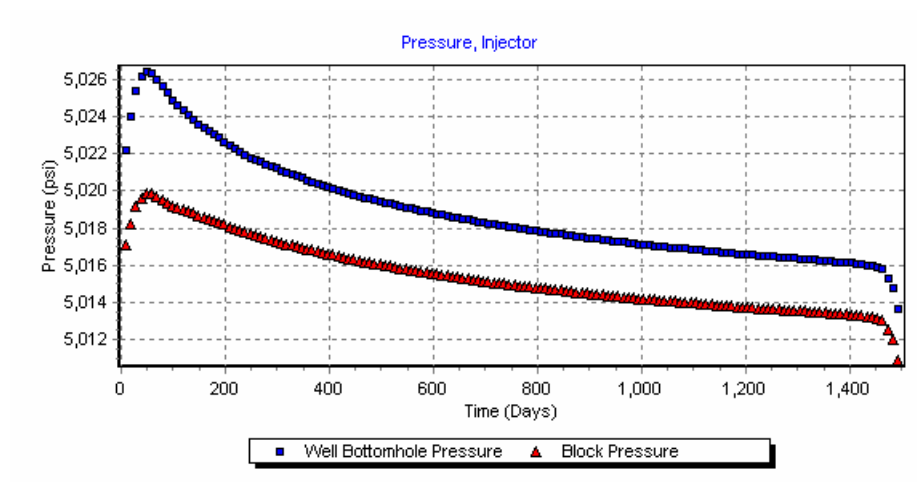


Fig. A.15 – Example showing pressure profiles plot of injector

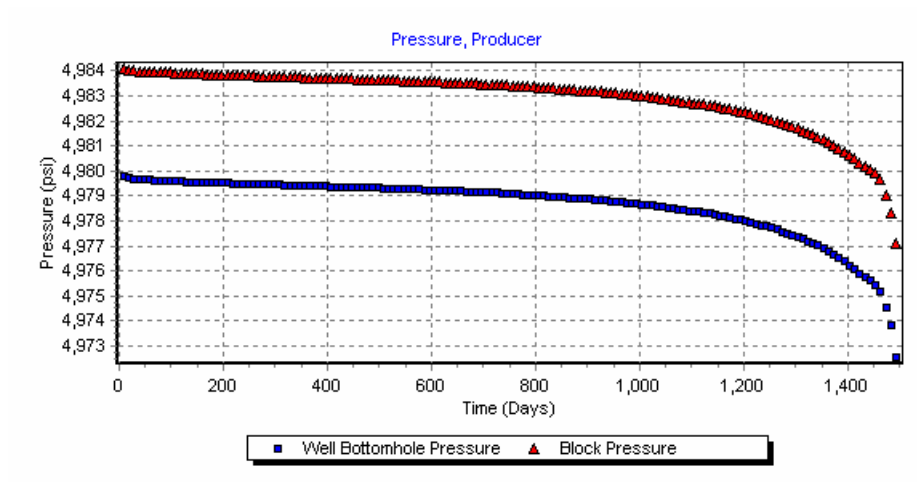


Fig. A.16 – Example showing pressure profiles plot of producer

APPENDIX B

Sim2D VB SOURCE CODE

'NFR Group - Texas A&M University - Fall 2003

'Assumptions:-

'1) Homogeneous reservoir

'2) No gravity effects

'3) No capillary pressure

'4) Isotropic reservoir

'The equations assumed FIELD units and they are shown as follows:-

'Oil and Water Formation Volume Factor : rcf/scf

'Oil and Water Rate : scf/D

'Permeability : mD

'Pressures : psi

'Grid dimensions : ft

Option Explicit

Option Base 1

'Defining all variables

'Private constants and variables apply to all procedures in module

Public FindIndexRow As Boolean

Public Type notzeroIndex

n As Integer

indexRow() As Integer

End Type

Public rowFill() As notzeroIndex

Private Prod_Inj() As String

Private back_to_origin As Boolean

Private Cut_Sat As Boolean

Private Pn() As Double
 Private Pold() As Double
 Private oct_oct As Double
 Private oct_rect_tri As Double
 Private Amat() As Double
 Private Bmat() As Double
 Private Itmax As Long
 Private ErrorEst As Double
 Private NodeType() As Byte

'Time Variables

Private t_step As Integer
 Private t_stepmax As Double
 Private delt As Double
 Private tmax As Double
 Private step As Double
 Private SatCut As Integer
 Private MaxCut As Integer

'Spatial Variables

Private NX As Integer
 Private NY As Integer
 Private xD() As Double, yD() As Double
 Private delx() As Double, dely() As Double
 Private dx() As Double, dy() As Double
 Private gridType As String
 Private u As Integer
 Private v As Integer
 Private NN As Integer
 Private m As Double
 Private p As Double, q As Double, r As Double, S As Double, W As Double, no As Double, ht As Double
 Private areaOctagon As Double, areaSquare As Double, areaCornerTriangle As Double, areaWallTriangle As Double
 Private TL As Integer, BL As Integer, TR As Integer, BR As Integer
 Private wecount As Double, sncount As Double, owecount As Double, osncount As Double
 Private nodeArea() As Byte

'West side wall, East side wall, South side wall, North side wall

Private WS As Integer, SS As Integer
 Private WSW() As Double, ESW() As Double, SSW() As Double, NSW() As Double
 Private DN() As Integer

Private OSW As Double, ONE As Double, ONW As Double, OSE As Double, OWS As Double

Private OSS As Double, OSSW() As Double, ONSW() As Double, OWSW() As Double

Private OESW() As Double, Oct() As Double, Oct_XY() As Double, OXY() As Double

Private Start() As Double, Oend() As Double, CenterO() As Double, Squares() As Double

Private startcount As Integer, count As Integer

Private M_P() As Double

'Fluid Property Variables

Private Comp As Double, Ct() As Double

Private MaxSat As Double

Private Soi() As Double, Swi() As Double, SLi() As Double

Private Stotal() As Double

Private Swc As Double, So As Double

Private LambdaO() As Double, LambdaW() As Double, LambdaT() As Double

Private Transm() As Double

Private por() As Double, Por_new() As Double

Private Perm1 As Double

Private Perm2 As Double

Private Porc As Double

'Initial Condition Variables

Private Pinit As Double

Private Sor As Double

'Average rock and fluid properties

Private kavg As Double

Private kavgw As Double, miuoavgW As Double, BavgW As Double, delxW As Double

Private kavge As Double, miuoavgE As Double, BavgE As Double, delxE As Double

Private Perm() As Double

Private KROUPS As Double

Private KRWUPS As Double

'Coefficients of Matrix A and B

Private beta() As Double

Private AT() As Double

Private ACT() As Double, MB() As Double

Private AO() As Double, AW() As Double

Private ATtemp As Double

Private AoSat() As Double, AwSat() As Double

Private SumAO As Double, SumAw As Double

'Fluid Property Variables from Input Table

Private miuo() As Double, Bo() As Double, Co() As Double

Private miuw() As Double, Bw() As Double, Cw() As Double

Private Rso() As Double

Private p_pvt() As Double

Private Bo_pvt() As Double, miuo_pvt() As Double, Co_pvt() As Double, Rso_pvt() As Double

Private Bw_pvt() As Double, miuw_pvt() As Double, Cw_pvt() As Double

Private Sw_Tab() As Double, Krw_tab() As Double, Kro_tab() As Double, Krg_tab() As Double

Private Sw_int() As Double, Krwi() As Double, Kro_i() As Double, Krg_i() As Double

Private Rso_i() As Double

Private Son() As Double, Swn() As Double, Sg_n() As Double

Private Npvt As Integer, nRelperm As Integer, PVT() As Double

Private SL_tab() As Double, SL_int() As Double, nSL As Integer, SLN() As Double

Private miuo_avg() As Double, Bo_avg() As Double

Private miuw_avg() As Double, Bw_avg() As Double

'Well terms/properties

Private delp() As Double, dp As Double

Private Nwell As Single

Private iloc() As Single

Private WellType() As String

Private bhp As Double

Private Pwf() As Double, Pwfn() As Double

Private Qo() As Double, Qw() As Double, Qt() As Double

Private Qon() As Double, Qwn() As Double, Qtn() As Double

Private WConst() As Single

Private rw() As Double, skin() As Double, Jmodel() As Double, ro() As Double

Private IOIP As Double, Np As Double, Ni As Double, Nt As Double, Mbe As Double

'production parameters

'Chord Slope Terms

Private Vpn() As Double, Vp() As Double

Private Bon() As Double

Private Bwn() As Double

'MBE

Private OOIP() As Double, WWIP() As Double

Private OOIPn() As Double, WWIPn() As Double

Private CumOil As Double, CumWater As Double

Private TotalOIP As Double, TotalWIP As Double

Private MatBal() As Double

Private MatBalE() As Double

Sub Main()

```
'Time Step Skip
FindIndexRow = False
Dim prevStep As Double, nextStep As Double
Dim position As Integer, skipStep As Integer, countStep As Integer
position = 1
countStep = 1
```

```
Dim i As Integer, t As Double
SatCut = 0
Call Clear_Report
Call ReadData_Input
```

```
'Read PVT and RelPerm Data from Worksheet
Call Read_PVT
```

```
If (gridType = "HGB") Then
    Call Dimension
End If
```

```
Call Memory_Allocation
```

```
If (gridType = "HGB") Then
    Call GridHGB
ElseIf (gridType = "SQ") Then
    Call GridSQ
End If
```

```
Call Initial
Call New_Trans
```

```
If Nwell <> 0 Then
    Call Well
End If
```

```
t_stepmax = Int(tmax / delt)
t_step = 1
```

```
Do While t < tmax
```

prev:

DoEvents

t = t + delt

Cut_Sat = False

Call Interpol

Call MatrixB2

Call MatrixA

Call bicgstab(Pn(), Bmat(), Amat(), ErrorEst, Itmax, Pn())

For i = 1 To NN

 If Pn(i) < 0 Then Stop

Next i

Call Interpol

Call UpdateSat

If Cut_Sat = True And SatCut <= MaxCut Then

 If SatCut = MaxCut Then

 SatCut = 0

 GoTo MaxIter

 End If

 Cut_Sat = False

 t = t - delt

 delt = delt / 2

 Back_To_Previous

 SatCut = SatCut + 1

 GoTo prev

End If

MaxIter:

 Call Material_Balance

 skipStep = 1000

 If (skipStep = countStep) Then

 Call Print_Result(position, t)

 Call Print_Result_WSat(position, t)

 Call Print_Result_OSat(position, t)

 Call Print_Result_TSat(position, t)

 Call Print_Well(position, t)

 Call Print_Pold(position, t)

 Call Print_ORate(position, t)

 Call Print_WRate(position, t)

 Call Print_TRate(position, t)

```

    Call Print_MBE(position, t)
    Call Print_Result_MatBal(position, t)

    prevStep = t
    position = position + 1
    countStep = 1
End If

countStep = countStep + 1
t_step = t_step + delt

'Update Properties
Call UpdateProperties

If Nwell <> 0 Then
    Call Well_New2
End If

Loop

End Sub

*****

Sub Print_Result(ByVal step As Integer, ByVal t As Double)
Dim i As Integer
With Sheets("Pn")
    .Cells(i + 1, step + 1) = t
    For i = 1 To NN
        .Cells(i + 1, step + 1) = Pn(i)
    Next
End With

End Sub

*****

Sub Print_Pold(ByVal step As Integer, ByVal t As Double)
Dim i As Integer
With Sheets("Pold")
    .Cells(i + 1, step + 1) = t
    For i = 1 To NN
        .Cells(i + 1, step + 1) = Pold(i)
    Next

```

End With

End Sub

Sub Back_To_Previous()

Dim i As Integer

For i = 1 To NN

 Pn(i) = Pold(i)

 Son(i) = Soi(i)

 Swn(i) = Swi(i)

 Qon(i) = Qo(i)

 Qwn(i) = Qw(i)

 Qtn(i) = Qt(i)

 Pwfn(i) = Pwf(i)

 Vpn(i) = Vp(i)

Next i

End Sub

Sub Print_Result_WSat(ByVal step As Integer, ByVal t As Double)

Dim i As Integer

With Sheets("WaterSat")

 .Cells(i + 1, step + 1) = t

For i = 1 To NN

 .Cells(i + 1, step + 1) = Swn(i)

Next

End With

End Sub

Sub Print_Result_OSat(ByVal step As Integer, ByVal t As Double)

Dim i As Integer

With Sheets("OilSat")

 .Cells(i + 1, step + 1) = t

For i = 1 To NN

 .Cells(i + 1, step + 1) = Son(i)

Next

End With

End Sub

Sub Print_Result_TSat(ByVal step As Integer, ByVal t As Double)

Dim i As Integer

With Sheets("TotalSat")

.Cells(i + 1, step + 1) = t

For i = 1 To NN

.Cells(i + 1, step + 1) = Stotal(i)

Next

End With

End Sub

Sub ReadData_Input()

Dim i As Integer, k As Integer, j As Integer

'Read input data from data sheet of workbook

With Worksheets("Data")

gridType = .Cells(6, 10).Value

If (gridType = "HGB") Then

u = .Cells(4, 2).Value

v = .Cells(5, 2).Value

m = .Cells(4, 5).Value

ElseIf (gridType = "SQ") Then

NX = .Cells(2, 13).Value

NY = .Cells(2, 16).Value

NN = NX * NY

ReDim dx(NN)

ReDim dy(NN)

k = 1

For j = 1 To NY

For i = 1 To NX

dx(k) = .Cells(2 + i, 13).Value

k = k + 1

Next


```

Next
k = 1
For j = 1 To NX
  For i = 1 To NY
    dy(k) = .Cells(2 + i, 16).Value
    k = k + 1
  Next
Next
End If

Pinit = .Cells(10, 2).Value
Comp = .Cells(7, 5).Value
ht = .Cells(8, 5).Value
delt = .Cells(13, 5).Value
tmax = .Cells(14, 5).Value
Swc = .Cells(11, 2).Value
Sor = .Cells(12, 2).Value
Porc = .Cells(5, 5).Value
bhp = .Cells(22, 9).Value
Perm1 = .Cells(6, 5).Value
Perm2 = Perm1
Nwell = .Cells(20, 3).Value
Itmax = Range("maxIT")
ErrorEst = Range("errorEST")
MaxSat = Range("maxsat")
MaxCut = Range("maxcut")

End With
End Sub

*****

Sub ReadData_Tables()
'Read input data from PVT Tables

Dim txttmp As String
Dim i As Byte
Dim AddR As String
AddR = ActiveWorkbook.Path
Open AddR & "\2D2Pv5.TXT" For Input As 1
Line Input #1, txttmp
Input #1, Npvt

```

```

ReDim p_pvt(1 To Npvt)
ReDim Bo_pvt(1 To Npvt)
ReDim Co_pvt(1 To Npvt)
ReDim Rso_pvt(1 To Npvt)
ReDim miuo_pvt(1 To Npvt)
ReDim Bw_pvt(1 To Npvt)
ReDim miuw_pvt(1 To Npvt)
ReDim Cw_pvt(1 To Npvt)
ReDim Kro_tab(1 To Npvt)
ReDim Krw_tab(1 To Npvt)
ReDim Sw_Tab(1 To Npvt)

```

```

Line Input #1, txttmp
For i = 1 To Npvt
  Input #1, p_pvt(i), Bo_pvt(i), Rso_pvt(i), miuo_pvt(i), Co_pvt(i), Bw_pvt(i),
  miuw_pvt(i), _
  Cw_pvt(i)
Next i

```

```

Line Input #1, txttmp
Line Input #1, txttmp
Input #1, nRelperm

```

```

ReDim Sw_Tab(1 To nRelperm)
ReDim Krw_tab(1 To nRelperm)
ReDim Kro_tab(1 To nRelperm)

```

```

Line Input #1, txttmp
For i = 1 To nRelperm
  Input #1, Sw_Tab(i), Krw_tab(i), Kro_tab(i)
  'Debug.Print Sw_Tab(i), Krw_tab(i), Kro_tab(i)
Next

```

```

Close #1

```

```

End Sub

```

```

*****

```

```

Sub Read_PVT()
Dim i As Integer, j As Integer

```

```

'Read input data from PVT sheet of workbook
With Worksheets("PVT")

```

```

    Npvt = Range("nPVT")
    nRelperm = Range("nRelPerm")
End With

ReDim p_pvt(1 To Npvt): ReDim Bo_pvt(1 To Npvt): ReDim Co_pvt(1 To Npvt):
ReDim Rso_pvt(1 To Npvt)
ReDim miuo_pvt(1 To Npvt)
ReDim Bw_pvt(1 To Npvt): ReDim miuw_pvt(1 To Npvt): ReDim Cw_pvt(1 To Npvt)
ReDim Sw_Tab(1 To nRelperm): ReDim Kro_tab(1 To nRelperm): ReDim Krw_tab(1
To nRelperm)

With Worksheets("PVT")
    For i = 1 To Npvt
        p_pvt(i) = .Cells(3 + i, 1).Value
        Bo_pvt(i) = .Cells(3 + i, 2).Value
        Rso_pvt(i) = .Cells(3 + i, 3).Value
        miuo_pvt(i) = .Cells(3 + i, 4).Value
        Co_pvt(i) = .Cells(3 + i, 5).Value
        Bw_pvt(i) = .Cells(3 + i, 6).Value
        miuw_pvt(i) = .Cells(3 + i, 7).Value
        Cw_pvt(i) = .Cells(3 + i, 8).Value
    Next i

    For j = 1 To nRelperm
        Sw_Tab(j) = .Cells(67 + j, 1).Value
        Krw_tab(j) = .Cells(67 + j, 2).Value
        Kro_tab(j) = .Cells(67 + j, 3).Value
    Next j
End With

End Sub

*****

Sub Dimension()

'Total Number of Blocks

$$NN = (v * u) + 2 * (v - 1) + 2 * (u - 1) + 4 + (v - 1) * (u - 1)$$


'Dimensions of polygons
p = m
q = p
r = q

```

```

S = 0.5 * m
W = S
no = (0.5 + 1 / Sqr(2)) * m

```

```

oct_oct = 2 * no
oct_rect_tri = no + S

```

```

'Areas of Polygons

```

```

areaSquare = m * m

```

```

areaOctagon = 2 * (m / Sqr(2)) ^ 2 + 4 * m * (m / Sqr(2)) + m ^ 2

```

```

areaCornerTriangle = 0.25 * m * m

```

```

areaWallTriangle = 0.5 * m * m

```

```

End Sub

```

```

*****

```

```

Sub GridSQ()

```

```

Dim i As Integer, j As Integer, k As Integer

```

```

Dim E As Integer, W As Integer, n As Integer, S As Integer

```

```

ReDim DN(NN, 8)

```

```

E = 8

```

```

n = 5

```

```

W = 6

```

```

S = 7

```

```

k = 1

```

```

For j = 1 To NY

```

```

    For i = 1 To NX

```

```

        If i <> 1 Then

```

```

            DN(k, W) = k - 1

```

```

        End If

```

```

        If i <> NX Then

```

```

            DN(k, E) = k + 1

```

```

        End If

```

```

        If j <> 1 Then

```

```

            DN(k, n) = k - NX

```

```

        End If

```

```

    If j <> NY Then
      DN(k, S) = k + NX
    End If

    k = k + 1
  Next
Next

For i = 1 To NN
  For j = 1 To 8
    If DN(i, j) <> 0 Then Perm(i, DN(i, j)) = Perm1
  Next
Next

End Sub

*****

Sub GridHGB()
Dim i As Integer, j As Integer, k As Integer, kk As Integer, A As Integer, temp As Integer
Dim ii As Integer, jj As Integer
ReDim DN(NN, 8)

' Corners
BL = 1
TL = 2

For i = 1 To v - 1
  TL = TL + (2 * i + 1)
Next i

BR = NN - (TL - BL)
TR = NN

' West Side Wall
wecount = 0
ReDim WSW(v)
WS = 2

For i = 1 To v - 1
  WSW(i) = WS
  wecount = wecount + 1
  WS = WS + ((2 * i) + 1)

```

```

Next i

'East Side Wall
ReDim ESW(v)

For i = 1 To wecount
    ESW(i) = NN - (WSW(i) - BL)
Next i

'South Side Wall
sncount = 0
SS = 4
ReDim SSW(u)
For i = 1 To u - 1
    k = i
    If k > v - 1 Then
        k = v - 1
    End If
    SSW(i) = SS
    sncount = sncount + 1
    SS = SS + ((2 * (k + 1)) + 1)
Next i

'North Side Wall
ReDim NSW(u)
For i = 1 To sncount
    NSW(i) = NN - (SSW(i) - BL)
Next i

'Octagons Corners
OSW = 3
ONE = NN - 2
ONW = 3

For i = 1 To v - 1
    ONW = ONW + (2 * i + 1)
Next i

OSE = ONE - (ONW - OSW)

'Octagon West Side Walls
OWS = 3
owecount = 0
ReDim OWSW(v)

```

```

For i = 1 To v - 2
    OWS = OWS + (2 * i + 1)
    OWSW(i) = OWS
    owecount = owecount + 1
Next i

'Octagon East Side Walls
ReDim OESW(v)

For i = 1 To owecount
    OESW(i) = ONE - (OWSW(i) - OSW)
Next i

'Octagon South Side Walls
OSS = 3
osncount = 0
ReDim OSSW(u)
i = 2

For j = 1 To u - 2
    k = i
    If k > v Then
        k = v
    End If
    OSS = OSS + (2 * k + 1)
    OSSW(i - 1) = OSS
    osncount = osncount + 1
    i = i + 1
Next j

'Octagon North Side Walls
ReDim ONSW(u)

For i = 1 To osncount
    ONSW(i) = ONE - (OSSW(i) - OSW)
Next i

'Non-corner Octagon
ReDim Start(owecount + 1 + osncount)
ReDim Oend(owecount + 1 + osncount)
ReDim CenterO(2 * owecount * osncount)
ReDim Squares((owecount + 2) * (osncount + 2))
startcount = 0

```

```

For i = 1 To owecount
  Start(i) = OWSW(i)
  Oend(i) = OESW(i)
  startcount = startcount + 1
Next i

```

```

Start(owecount + 1) = ONW
Oend(owecount + 1) = OSE

```

```

For i = osncount + owecount + 1 To owecount + 2 Step -1
  Start(i) = ONSW(i - owecount - 1)
  Oend(i) = OSSW(i - owecount - 1)
  startcount = startcount + 1
Next i

```

```

Call Sorter(Start(), owecount + 1# + osncount)
Call Sorter(Oend(), owecount + 1# + osncount)

```

'Map octagons & squares

```

k = 1
kk = 1
For i = 1 To startcount + 1
  For j = Start(i) + 1 To Oend(i) - 1 Step 2
    Squares(kk) = j
    kk = kk + 1
  Next j

  For j = Start(i) + 2 To Oend(i) - 1 Step 2
    CenterO(k) = j
    k = k + 1
  Next j
Next i

```

'Map octagons to x & y coordinates

```

ReDim Oct(u * v)
j = 1

```

```

For i = 1 To owecount
  Oct(i) = OWSW(j)
  j = j + 1
Next i
k = i

```



```

j = 1
For i = k To (owecount + k)
    Oct(i) = OESW(j)
    j = j + 1
Next i
k = i - 1

```

```

j = 1
For i = k To (osncount + k)
    Oct(i) = OSSW(j)
    j = j + 1
Next i
k = i - 1

```

```

j = 1
For i = k To (osncount + k)
    Oct(i) = ONSW(j)
    j = j + 1
Next i
k = i - 1

```

```

j = 1
For i = k To ((owecount * osncount) + k - 1)
    Oct(i) = CenterO(j)
    j = j + 1
Next i

```

```

k = i

```

```

Oct(k) = OSW
Oct(k + 1) = ONW
Oct(k + 2) = OSE
Oct(k + 3) = ONE

```

```

Call Sorter(Oct(), (u * v))
ReDim Oct_XY(u * 2, v * 2)
i = 1
j = 1
count = 1
Oct_XY(1, 1) = Oct(1)

```

```

For A = 2 To (u * v)
    If Oct(A) = Oct(A - 1) + 3 Then
        If (j + count) > v Then

```

```

        temp = count - (v - j)
        count = count - temp
    End If
    j = j + count
    i = i - (count - 1)

    count = 1
    ElseIf Oct(A) = Oct(A - 1) + 2 Then
        j = j - 1
        i = i + 1
        count = count + 1
    End If

Oct_XY(i, j) = Oct(A)

Next A

'Remap Octagons to normal XY Grid
ii = 1
ReDim OXY(u * 2, v * 2)

For i = 1 To u
    jj = 1
    For j = 1 To v
        OXY(ii, jj) = Oct_XY(i, j)
        jj = jj + 2
    Next j
    ii = ii + 2
Next i

'Map Squares to x & y coordinates
For i = 2 To (u - 1) * 2 Step 2
    For j = 2 To (v - 1) * 2 Step 2
        OXY(i, j) = OXY(i - 1, j + 1) + 1
    Next j
Next i

'2D Array for Directions - DN
'1D Array for Node Area - nodeArea

'Corners
DN(1, 1) = 3
DN(TL, 4) = ONW
DN(TR, 3) = ONE

```

DN(BR, 2) = OSE

nodeArea(1) = 3
 nodeArea(TL) = 3
 nodeArea(TR) = 3
 nodeArea(BR) = 3

'West Side Walls

DN(WSW(1), 4) = OSW
 DN(WSW(1), 1) = OWSW(1)
 DN(WSW(wecount), 4) = OWSW(owecount)
 DN(WSW(wecount), 1) = ONW

nodeArea(WSW(1)) = 2
 nodeArea(WSW(wecount)) = 2

j = 1
 For i = 2 To wecount - 1
 DN(WSW(i), 4) = OWSW(j)
 DN(WSW(i), 1) = OWSW(j + 1)
 nodeArea(WSW(i)) = 2
 j = j + 1
 Next i

'East Side Walls

Call Sorter(ESW(), wecount)
 Call Sorter(OESW(), owecount)

DN(ESW(1), 2) = OESW(1)
 DN(ESW(1), 3) = OSE
 DN(ESW(wecount), 2) = ONE
 DN(ESW(wecount), 3) = OESW(owecount)

nodeArea(ESW(1)) = 2
 nodeArea(ESW(wecount)) = 2

j = 1
 For i = 2 To wecount - 1
 DN(ESW(i), 3) = OESW(j)
 DN(ESW(i), 2) = OESW(j + 1)
 nodeArea(ESW(i)) = 2
 j = j + 1
 Next i

'South Side Walls

DN(SSW(1), 1) = OSSW(1)

DN(SSW(1), 2) = OSW

DN(SSW(sncount), 1) = OSE

DN(SSW(sncount), 2) = OSSW(osncount)

nodeArea(SSW(1)) = 2

nodeArea(SSW(sncount)) = 2

j = 1

For i = 2 To sncount - 1

 DN(SSW(i), 2) = OSSW(j)

 DN(SSW(i), 1) = OSSW(j + 1)

 nodeArea(SSW(i)) = 2

j = j + 1

Next i

'North Side Walls

Call Sorter(NSW(), sncount)

Call Sorter(ONSW(), osncount)

DN(NSW(1), 3) = ONW

DN(NSW(1), 4) = ONSW(1)

DN(NSW(sncount), 3) = ONSW(osncount)

DN(NSW(sncount), 4) = ONE

nodeArea(NSW(1)) = 2

nodeArea(NSW(sncount)) = 2

j = 1

For i = 2 To sncount - 1

 DN(NSW(i), 3) = ONSW(j)

 DN(NSW(i), 4) = ONSW(j + 1)

 nodeArea(NSW(i)) = 2

j = j + 1

Next i

'Octagons Corners

DN(OSW, 1) = OXY(2, 2)

DN(OSW, 2) = WSW(1)

DN(OSW, 3) = BL

DN(OSW, 4) = SSW(1)

DN(OSW, 5) = OWSW(1)

DN(OSW, 8) = OSSW(1)

DN(ONW, 1) = NSW(1)
 DN(ONW, 2) = TL
 DN(ONW, 3) = WSW(wecount)
 DN(ONW, 4) = OXY(2, 2 * v - 2)
 DN(ONW, 7) = OWSW(owecount)
 DN(ONW, 8) = ONSW(1)

DN(ONE, 1) = TR
 DN(ONE, 2) = NSW(sncount)
 DN(ONE, 3) = OXY(u * 2 - 2, 2 * v - 2)
 DN(ONE, 4) = ESW(wecount)
 DN(ONE, 6) = ONSW(osncount)
 DN(ONE, 7) = OESW(owecount)

DN(OSE, 1) = ESW(1)
 DN(OSE, 2) = OXY(u * 2 - 2, 2)
 DN(OSE, 3) = SSW(sncount)
 DN(OSE, 4) = BR
 DN(OSE, 5) = OESW(1)
 DN(OSE, 6) = OSSW(osncount)

nodeArea(OSW) = 0
 nodeArea(ONW) = 0
 nodeArea(ONE) = 0
 nodeArea(OSE) = 0

'Octagon Walls

'Octagon West Side Walls

i = 1

count = 2

For j = 3 To v + v - 3 Step 2

DN(OXY(i, j), 1) = OXY(i + 1, j + 1)

DN(OXY(i, j), 2) = WSW(count)

DN(OXY(i, j), 3) = WSW(count - 1)

DN(OXY(i, j), 4) = OXY(i + 1, j - 1)

DN(OXY(i, j), 5) = OXY(i, j + 2)

DN(OXY(i, j), 7) = OXY(i, j - 2)

DN(OXY(i, j), 8) = OXY(i + 2, j)

nodeArea(OXY(i, j)) = 0

count = count + 1

Next j

'Octagon East Side Walls

i = u + (u - 1)

```

count = 2
For j = 3 To v + v - 3 Step 2
  DN(OXY(i, j), 1) = ESW(count)
  DN(OXY(i, j), 2) = OXY(i - 1, j + 1)
  DN(OXY(i, j), 3) = OXY(i - 1, j - 1)
  DN(OXY(i, j), 4) = ESW(count - 1)
  DN(OXY(i, j), 5) = OXY(i, j + 2)
  DN(OXY(i, j), 6) = OXY(i - 2, j)
  DN(OXY(i, j), 7) = OXY(i, j - 2)
  nodeArea(OXY(i, j)) = 0
  count = count + 1
Next j

```

'Octagon South Side Walls

```

j = 1
count = 2
For i = 3 To u + u - 3 Step 2
  DN(OXY(i, j), 1) = OXY(i + 1, j + 1)
  DN(OXY(i, j), 2) = OXY(i - 1, j + 1)
  DN(OXY(i, j), 3) = SSW(count - 1)
  DN(OXY(i, j), 4) = SSW(count)
  DN(OXY(i, j), 5) = OXY(i, j + 2)
  DN(OXY(i, j), 6) = OXY(i - 2, j)
  DN(OXY(i, j), 8) = OXY(i + 2, j)
  nodeArea(OXY(i, j)) = 0
  count = count + 1
Next i

```

'Octagon North Side Walls

```

j = 2 * v - 1
count = 2
For i = 3 To u + u - 3 Step 2
  DN(OXY(i, j), 1) = NSW(count)
  DN(OXY(i, j), 2) = NSW(count - 1)
  DN(OXY(i, j), 3) = OXY(i - 1, j - 1)
  DN(OXY(i, j), 4) = OXY(i + 1, j - 1)
  DN(OXY(i, j), 6) = OXY(i - 2, j)
  DN(OXY(i, j), 7) = OXY(i, j - 2)
  DN(OXY(i, j), 8) = OXY(i + 2, j)
  nodeArea(OXY(i, j)) = 0
  count = count + 1
Next i

```

'Center Octagons

```

For i = 3 To 2 * u - 3 Step 2
  For j = 3 To 2 * v - 3 Step 2
    DN(OXY(i, j), 1) = OXY(i + 1, j + 1)
    DN(OXY(i, j), 2) = OXY(i - 1, j + 1)
    DN(OXY(i, j), 3) = OXY(i - 1, j - 1)
    DN(OXY(i, j), 4) = OXY(i + 1, j - 1)
    DN(OXY(i, j), 5) = OXY(i, j + 2)
    DN(OXY(i, j), 6) = OXY(i - 2, j)
    DN(OXY(i, j), 7) = OXY(i, j - 2)
    DN(OXY(i, j), 8) = OXY(i + 2, j)
    nodeArea(OXY(i, j)) = 0
  Next j
Next i

'Squares
For i = 2 To 2 * u - 2 Step 2
  For j = 2 To 2 * v - 2 Step 2
    DN(OXY(i, j), 1) = OXY(i + 1, j + 1)
    DN(OXY(i, j), 2) = OXY(i - 1, j + 1)
    DN(OXY(i, j), 3) = OXY(i - 1, j - 1)
    DN(OXY(i, j), 4) = OXY(i + 1, j - 1)
    nodeArea(OXY(i, j)) = 1
  Next j
Next i

For i = 1 To NN
  For j = 1 To 8
    If DN(i, j) <> 0 Then Perm(i, DN(i, j)) = Perm1
  Next
Next

End Sub

*****

Sub Memory_Allocation()
ReDim Pold(NN): ReDim Pn(NN): ReDim Pwf_f(NN)
ReDim Soi(NN): ReDim Swi(NN): ReDim SLi(NN)
ReDim Son(NN): ReDim Swn(NN): ReDim Stotal(NN)
ReDim Vp(NN)
ReDim nodeArea(NN)
ReDim por(NN)
ReDim Perm(NN, NN)
ReDim Ct(NN)

```

```

ReDim rowFill(NN)
ReDim miuo(NN): ReDim Bo(NN): ReDim Co(NN)
ReDim miuw(NN): ReDim Bw(NN): ReDim Cw(NN)
ReDim Kroi(NN): ReDim Krwi(NN)
ReDim LambdaO(NN): ReDim LambdaW(NN): ReDim LambdaT(NN)
ReDim beta(NN): ReDim MB(NN): ReDim Transm(NN, 8)
ReDim miuo_avg(NN, 8): ReDim miuw_avg(NN, 8)
ReDim Bo_avg(NN, 8): ReDim Bw_avg(NN, 8)
ReDim AO(NN, 8): ReDim AW(NN, 8): ReDim AT(NN, 8)
ReDim ACT(NN)
ReDim Qo(NN): ReDim Qw(NN): ReDim Qt(NN)
ReDim Qon(NN): ReDim Qwn(NN): ReDim Qtn(NN)

```

```

If Nwell <> 0 Then
    ReDim iloc(Nwell): ReDim WConst(Nwell): ReDim WellType(Nwell)
    ReDim rw(Nwell): ReDim ro(Nwell): ReDim skin(Nwell)
End If

```

```

ReDim Jmodel(NN)
ReDim Pwf(NN): ReDim Pwfn(NN)
ReDim LambdaO(NN), LambdaW(NN), LambdaG(NN), LambdaT(NN)
ReDim Vpn(NN)
ReDim Bon(NN): ReDim Bwn(NN)
ReDim OOIP(NN): ReDim WWIP(NN)
ReDim OOIPn(NN): ReDim WWIPn(NN)
ReDim MatBal(NN): ReDim MatBalE(NN)

```

```
End Sub
```

```
*****
```

```
Sub Sorter(Int_no() As Double, Int_ArraySize As Double)
```

```
Dim temp As Double
```

```
Dim i As Integer, j As Integer
```

```
For i = Int_ArraySize To 1 Step -1
```

```
    For j = 2 To i
```

```
        If Int_no(j - 1) > Int_no(j) Then
```

```
            temp = Int_no(j - 1)
```

```
            Int_no(j - 1) = Int_no(j)
```

```
            Int_no(j) = temp
```

```
        End If
```

```
    Next j
```

```
Next i
```


End Sub

Sub Initial()

Dim i As Integer, j As Integer, k As Integer

Dim SumC1 As Byte, SumC2 As Byte

'Initializes time and arrays for pressure, and coefficients.

Nt = 0

Np = 0

Ni = 0

For i = 1 To NN

 Pold(i) = Pinit

 Pn(i) = Pold(i)

 Pwfn(i) = Pwf(i)

 Soi(i) = Sor

 Swi(i) = Swc

 Son(i) = Soi(i)

 Swn(i) = Swi(i)

 por(i) = Porc

Next i

If (gridType = "HGB") Then

 For i = 1 To NN

 SumC1 = 0

 For j = 1 To 8

 If DN(i, j) <> 0 Then SumC1 = SumC1 + 1

 Next j

 If SumC1 >= 5 Then

 Vp(i) = por(i) * areaOctagon * ht

 ElseIf SumC1 = 4 Then

 Vp(i) = por(i) * areaSquare * ht

 ElseIf SumC1 = 2 Then

 Vp(i) = por(i) * areaWallTriangle * ht

 ElseIf SumC1 = 1 Then

 Vp(i) = por(i) * areaCornerTriangle * ht

 End If

 Next i

```

ElseIf (gridType = "SQ") Then
  For i = 1 To NN
    Vp(i) = por(i) * dx(i) * dy(i) * ht
  Next i
End If

```

```

Call Call_PVT(Pold())
Call Mobilities

```

```

TotalOIP = 0#
TotalWIP = 0#

```

```

For i = 1 To NN
  Bon(i) = Bo(i)
  Bwn(i) = Bw(i)
  Vpn(i) = Vp(i)
  OOIP(i) = Vp(i) * Soi(i) 'rcf
  WWIP(i) = Vp(i) * Swi(i) 'rcf
  TotalOIP = TotalOIP + OOIP(i) 'rcf
  TotalWIP = TotalWIP + WWIP(i) 'rcf
Next i

```

```

CumOil = 0#: CumWater = 0#

```

```

End Sub

```

```

*****

```

```

Sub Interpol()
  Call Call_PVT(Pn())
  Call Avg_PVT
  Call Mobilities
End Sub

```

```

*****

```

```

Sub MatrixB()
  Dim i As Integer, j As Integer
  Dim SumC1 As Byte
  ReDim Bmat(NN)

```

```

For i = 1 To NN
  Ct(i) = (Son(i) * Co(i)) + (Swn(i) * Cw(i)) + Comp
Next i

```

```

For i = 1 To NN
  beta(i) = Vpn(i) * Ct(i) / delt
  MB(i) = -beta(i) * Pold(i) 'rcf/Day
  Bmat(i) = MB(i)
Next i

For i = 1 To Nwell
  j = iloc(i)

  Select Case WellType(i)
    Case "ORate", "WRate"
      MB(j) = (-beta(j) * Pold(j)) + Qo(j) + Qw(j) 'rcf/Day
      Bmat(j) = MB(j)

    Case "Pres"
      MB(j) = (-beta(j) * Pold(j)) - (Jmodel(i) * LambdaT(j) * Pwf(j)) 'rcf/Day
      'MB(j) = (-beta(j) * Pn(j)) + (Jmodel(i) * LambdaO(j) * (Pn(j) - Pwf(j))) +
      (Jmodel(i) * LambdaW(j) * (Pn(j) - Pwf(j))) 'rcf/Day
      'MB(j) = -beta(j) * Pn(j) + Qo(j) + Qw(j) 'rcf/Day
      Bmat(j) = MB(j)
  End Select
Next i

End Sub

```

```

'*****

```

```

Sub Call_PVT(Px() As Double)
Dim i As Integer

For i = 1 To NN
  miuo(i) = Interpolate(Px(i), p_pvt(), miuo_pvt())
  Bo(i) = Interpolate(Px(i), p_pvt(), Bo_pvt())
  Co(i) = Interpolate(Px(i), p_pvt(), Co_pvt())
  miuw(i) = Interpolate(Px(i), p_pvt(), miuw_pvt())
  Bw(i) = Interpolate(Px(i), p_pvt(), Bw_pvt())
  Cw(i) = Interpolate(Px(i), p_pvt(), Cw_pvt())
  Kro(i) = Interpolate(Swi(i), Sw_Tab(), Kro_tab())
  Krwi(i) = Interpolate(Swi(i), Sw_Tab(), Krw_tab())
Next i

End Sub

```

Function Interpolate(y As Double, mm() As Double, Nm() As Double) As Double

Dim i As Double

Dim A1 As Double, A2 As Double, B1 As Double, B2 As Double

If mm(LBound(mm)) > mm(UBound(mm)) Then

For i = LBound(mm) To UBound(mm) - 1

If y <= mm(i) And y > mm(i + 1) Then

A1 = mm(i)

A2 = mm(i + 1)

B1 = Nm(i)

B2 = Nm(i + 1)

End If

If y > mm(LBound(mm)) Then

A1 = mm(LBound(mm))

A2 = mm(LBound(mm) + 1)

B1 = Nm(LBound(mm))

B2 = Nm(LBound(mm) + 1)

End If

If y < mm(UBound(mm)) Then

A1 = mm(UBound(mm))

A2 = mm(UBound(mm) - 1)

B1 = Nm(UBound(mm))

B2 = Nm(UBound(mm) - 1)

End If

Next i

Else

For i = 1 To UBound(mm) - 1

If y >= mm(i) And y <= mm(i + 1) Then

A1 = mm(i)

A2 = mm(i + 1)

B1 = Nm(i)

B2 = Nm(i + 1)

End If

If y < mm(LBound(mm)) Then

A1 = mm(LBound(mm))

A2 = mm(LBound(mm) + 1)

B1 = Nm(LBound(mm))

B2 = Nm(LBound(mm))

End If

If y > mm(UBound(mm)) Then

A1 = mm(UBound(mm))

A2 = mm(UBound(mm) - 1)

```

        B1 = Nm(UBound(mm))
        B2 = Nm(UBound(mm))
    End If

    Next i
End If
Interpolate = B1 + (B2 - B1) / (A2 - A1) * (y - A1)

End Function

*****

Sub Avg_PVT()

Dim i As Integer, j As Integer
For i = 1 To NN
    For j = 1 To 8
        If (DN(i, j) <> 0) Then
            miuo_avg(i, j) = ArithAvg(miuo(i), miuo(DN(i, j)))
            miuw_avg(i, j) = ArithAvg(miuw(i), miuw(DN(i, j)))
            Bo_avg(i, j) = ArithAvg(Bo(i), Bo(DN(i, j)))
            Bw_avg(i, j) = ArithAvg(Bw(i), Bw(DN(i, j)))
        End If
    Next j
Next i

End Sub

*****

Sub Trans()
Dim i As Integer, j As Integer, k As Byte
Dim E As Integer, W As Integer, n As Integer, S As Integer
Dim SumC1 As Byte
Dim SumC2 As Byte

E = 8
n = 5
W = 6
S = 7

If (gridType = "HGB") Then
    For i = 1 To NN
        SumC1 = 0

```

```

For k = 1 To 8
  If DN(i, k) <> 0 Then SumC1 = SumC1 + 1
Next

For j = 1 To 8
  If (DN(i, j) <> 0) Then

    kavg = HaAvg(Perm(i, DN(i, j)), Perm(DN(i, j), i))
    SumC2 = 0
    For k = 1 To 8
      If DN(DN(i, j), k) <> 0 Then SumC2 = SumC2 + 1
    Next

    If (SumC1 >= 5 And SumC2 >= 5) Then
      Transm(i, j) = 0.00633 * kavg * m * ht / oct_oct
    ElseIf SumC1 <> SumC2 And (SumC1 <= 4 Or SumC2 <= 4) Then
      Transm(i, j) = 0.00633 * kavg * m * ht / oct_rect_tri
    End If

  End If
Next j

Next i

ElseIf (gridType = "SQ") Then

  For i = 1 To NN
    For j = 1 To 8
      If (DN(i, j) <> 0) Then
        kavg = HaAvg(Perm(i, DN(i, j)), Perm(DN(i, j), i))

        If (j = W) Then
          Transm(i, j) = 0.00633 * kavg * dy(i) * ht / dx(i)
        ElseIf (j = S) Then
          Transm(i, j) = 0.00633 * kavg * dx(i) * ht / dy(i)
        ElseIf (j = E) Then
          Transm(i, j) = 0.00633 * kavg * dy(i) * ht / dx(i)
        ElseIf (j = n) Then
          Transm(i, j) = 0.00633 * kavg * dx(i) * ht / dy(i)
        End If

      End If
    Next j

  End If
Next i

```

```

    Next i
End If

```

```

End Sub

```

```

*****

```

```

Function HaAvg(A As Double, B As Double) As Double
HaAvg = 2 * A * B / (A + B)
End Function

```

```

*****

```

```

Function ArithAvg(A As Double, B As Double) As Double
ArithAvg = (A + B) / 2
End Function

```

```

*****

```

```

Sub MatrixA()
Dim inc As Integer
Dim i As Integer, j As Integer, k As Integer
Dim NAT(), sumXt As Double
ReDim AoSat(NN): ReDim AwSat(NN)
ReDim NAT(NN)
ReDim Amat(NN, NN)

```

```

'ADD UP TRANSM FROM ALL DIRECTIONS

```

```

For i = 1 To NN
  For j = 1 To 8
    If (DN(i, j) <> 0) Then
      Call Kr_upstream(i, DN(i, j))
      Debug.Print KROUPS, KRWUPS
      AO(i, j) = Transm(i, j) * KROUPS / (miuo_avg(i, j) * Bo_avg(i, j)) 'scf/psi-Day
      AW(i, j) = Transm(i, j) * KRWUPS / (miuw_avg(i, j) * Bw_avg(i, j))
    End If
  Next j
Next i

```

```

For i = 1 To NN
  sumXt = 0
  For j = 1 To 8
    If (DN(i, j) <> 0) Then

```

```

        AT(i, j) = (Bon(i) * AO(i, j)) + (Bwn(i) * AW(i, j)) 'rcf/psi-Day
        Amat(i, DN(i, j)) = AT(i, j)
        sumXt = sumXt + AT(i, j)
    End If
Next j

    NAt(i) = sumXt
    ACT(i) = -NAt(i) - beta(i)    'rcf/psi-Day
    Amat(i, i) = ACT(i)
Next i

For i = 1 To Nwell
    j = iloc(i)

    Select Case WellType(i)
    Case "Pres"
        ACT(j) = -NAt(j) - beta(j) - (Jmodel(i) * LambdaT(j))    'rcf/psi-Day
        Amat(j, j) = ACT(j)
    End Select
Next i

If FindIndexRow = False Then
For i = 1 To NN
    inc = 0
    For j = 1 To NN
        If Amat(i, j) <> 0 Then
            rowFill(i).n = inc + 1
            inc = inc + 1
            ReDim Preserve rowFill(i).indexRow(inc)
            rowFill(i).indexRow(inc) = j
        End If
    Next
Next
FindIndexRow = True
End If

End Sub

*****

Sub Kr_upstream(A As Integer, B As Integer)
'Single-point mobility-weighting

If Pn(A) >= Pn(B) Then

```



```

    KROUPS = Kro(A)
    KRWUPS = Krwi(A)
Else
    KROUPS = Kro(B)
    KRWUPS = Krwi(B)
End If

End Sub

Sub Kr_upstream2(A As Integer, B As Integer)
'Two-point mobility-weighting

If Pn(A) >= Pn(B) Then
    If A = 1 Or A = NN Then
        KROUPS = Kro(A)
        KRWUPS = Krwi(A)
    Else
        'For equal gridblock lengths
        KROUPS = 1.5 * Kro(A) - 0.5 * Kro(B)
        KRWUPS = 1.5 * Krwi(A) - 0.5 * Krwi(B)
    End If
Else
    If A = 1 Or A = NN Then
        KROUPS = Kro(B)
        KRWUPS = Krwi(B)
    Else
        KROUPS = 1.5 * Kro(B) - 0.5 * Kro(A)
        KRWUPS = 1.5 * Krwi(B) - 0.5 * Krwi(A)
    End If
End If

If (KRWUPS < 0) Then
    KRWUPS = 0
End If

If (Kro(A) > Kro(B)) Then
    If (KROUPS > Kro(A)) Then
        KROUPS = Kro(A)
    End If
Else
    If (KROUPS > Kro(B)) Then
        KROUPS = Kro(B)
    End If

```

End If

If (KRWUPS > 1#) Then

 KRWUPS = 1

End If

If (KROUPS < 0#) Then

 KROUPS = 0

End If

End Sub

Sub Well()

Dim i As Integer, j As Integer, k As Integer, ii As Integer

Dim SumC1 As Byte

Dim SumC2 As Byte

Dim pi As Double

ReDim Prod_Inj(Nwell)

pi = 22# / 7#

With Worksheets("Data")

 For i = 1 To Nwell

 iloc(i) = .Cells(21 + i, 1).Value

 WConst(i) = .Cells(21 + i, 6).Value

 WellType(i) = .Cells(21 + i, 2).Text

 rw(i) = .Cells(21 + i, 7).Value

 skin(i) = .Cells(21 + i, 8).Value

 Prod_Inj(i) = .Cells(21 + i, 10).Value

 If (gridType = "HGB") Then

 SumC1 = 0

 For k = 1 To 8

 If DN(iloc(i), k) <> 0 Then SumC1 = SumC1 + 1

 Next k

 If SumC1 = 4 Then

 'Square

 ro(i) = 0.208 * (m / 2)

```

ElseIf SumC1 = 2 Then
  'Wall Triangle
  ro(i) = Exp((2 * (m / oct_rect_tri) * Log(oct_rect_tri) - 3.142857143) / (2 * (m
/ oct_rect_tri)))

ElseIf SumC1 = 1 Then
  'Corner Triangle
  ro(i) = Exp(((m / oct_rect_tri) * Log(oct_rect_tri) - 1.5707963) / (m /
oct_rect_tri))

End If

For j = 1 To 8
  If (DN(iloc(i), j) <> 0) Then
    SumC2 = 0
    For k = 1 To 8
      If DN(DN(iloc(i), j), k) <> 0 Then SumC2 = SumC2 + 1
    Next k

    'Internal Octagons
    If (SumC1 = 8 And SumC2 < 5) Then
      ro(i) = oct_oct * Exp(-6.285714286 / (8 * m / oct_oct))

    ElseIf SumC1 = 8 And SumC2 >= 5 Then
      ro(i) = oct_rect_tri * Exp(-6.285714286 / (8 * m / oct_rect_tri))

    'Octagon Walls
    ElseIf SumC1 = 7 Then
      ro(i) = ((4 * ((m / oct_rect_tri) * Log(oct_rect_tri)) + _
        2 * ((m / oct_oct) * Log(oct_oct))) - 4.712389) / _
        (4 * (m / oct_rect_tri) + 2 * (m / oct_oct))

    ElseIf SumC1 = 6 Then
      ro(i) = ((4 * ((m / oct_rect_tri) * Log(oct_rect_tri)) + _
        3 * ((m / oct_oct) * Log(oct_oct))) - 5.4977871) / _
        (4 * (m / oct_rect_tri) + 3 * (m / oct_oct))

    End If
  End If
Next j

ElseIf (gridType = "SQ") Then
  ro(i) = 0.14 * (((dx(iloc(i)) ^ 2) + (dy(iloc(i)) ^ 2)) ^ 0.5)
End If

```

Next i

For i = 1 To Nwell

 j = iloc(i)

 Select Case WellType(i)

 Case "ORate"

 Qo(j) = (.Cells(21 + i, 3).Value) 'rcf/d

 Qw(j) = Qo(j) * LambdaW(j) / LambdaO(j) 'rcf/d

 Qt(j) = Qo(j) + Qw(j) 'rcf/d

 Pwf(j) = Pn(j) - Qt(j) / (Jmodel(i) * LambdaT(j)) 'psi

 Case "WRate"

 Qw(j) = (.Cells(21 + i, 4).Value) 'rcf/d

 If UCase(Prod_Inj(i)) = "PROD" Then

 Qo(j) = Qw(j) * LambdaO(j) / LambdaW(j) 'rcf/d

 End If

 Qt(j) = Qo(j) + Qw(j) 'rcf/d

 Pwf(j) = Pn(j) - Qt(j) / (Jmodel(i) * LambdaT(j)) 'psi

 Case "Pres"

 Pwf(j) = .Cells(21 + i, 5).Value 'psi

 End Select

Next i

End With

For i = 1 To NN

 Qon(i) = Qo(i)

 Qwn(i) = Qw(i)

 Qtn(i) = Qt(i)

 Pwfn(i) = Pwf(i)

Next i

End Sub

Sub Mobilities()

Dim i As Integer

For i = 1 To NN

 LambdaO(i) = Kroi(i) / miuo(i)

 LambdaW(i) = Krwi(i) / miuw(i)

```

    LambdaT(i) = LambdaO(i) + LambdaW(i)
Next i

```

```
End Sub
```

```
*****
```

```
Sub Chord_slope()
Dim i As Integer

```

```

    For i = 1 To NN
        Vpn(i) = Vp(i) * (1 + Comp * (Pn(i) - Pold(i)))
        Bon(i) = Bo(i) * (1 - Co(i) * (Pn(i) - Pold(i)))
        Bwn(i) = Bw(i) * (1 - Cw(i) * (Pn(i) - Pold(i)))
    Next i

```

```
End Sub
```

```
*****
```

```
Sub UpdateSat()
```

```

Dim i As Integer, j As Integer
Dim p_ref As Double, AoTot_DELP As Double, AwTot_DELP As Double

```

```
Call Chord_slope
```

```

For i = 1 To NN
    AoTot_DELP = 0
    AwTot_DELP = 0
    p_ref = Pn(i)

    For j = 1 To 8
        If DN(i, j) <> 0 Then AoTot_DELP = AoTot_DELP + (AO(i, j) * (Pn(DN(i, j)) -
p_ref)) 'scf/Day
        If DN(i, j) <> 0 Then AwTot_DELP = AwTot_DELP + (AW(i, j) * (Pn(DN(i, j)) -
p_ref))
    Next
    Son(i) = (Bon(i) / Vpn(i)) * (delt * (AoTot_DELP - Qon(i) / Bo(i)) + (Vp(i) * Soi(i) /
Bo(i)))
    Swn(i) = (Bwn(i) / Vpn(i)) * (delt * (AwTot_DELP - Qwn(i) / Bw(i)) + (Vp(i) *
Swi(i) / Bw(i)))

```

```

    If Son(i) < Sw_Tab(LBound(Sw_Tab)) Then

```

```

    Son(i) = Sw_Tab(LBound(Sw_Tab))
End If

If Swn(i) > Sw_Tab(UBound(Sw_Tab)) Then
    Swn(i) = Sw_Tab(UBound(Sw_Tab))
End If

Next i

For i = 1 To NN
    If ((Abs(Swn(i) - Swi(i)) >= MaxSat) Or (Abs(Son(i) - Soi(i)) >= MaxSat)) Then
        Cut_Sat = True
    Exit For
    End If
Next

For i = 1 To NN
    Stotal(i) = Son(i) + Swn(i)
Next i

End Sub

*****

Sub Well_New()
Dim i As Integer, j As Integer, count As Integer
ReDim delp(NN)

Call Call_PVT(Pn())
Call Mobilities

LoopAgain:
For i = 1 To Nwell
    j = iloc(i)
    Select Case WellType(i)
        Case "ORate"
            Qon(j) = Qo(j) 'rcf/d
            Qwn(j) = Qon(j) * LambdaW(j) / LambdaO(j) 'rcf/d
            Qtn(j) = Qon(j) + Qwn(j) 'rcf/d
            Pwfn(j) = Pn(j) - Qtn(j) / (Jmodel(i) * LambdaT(j)) 'psi
            If Qon(j) < 0 Then
                MsgBox "xxxx"
            End If
    End Select
Next i

```

```

    If Pwfn(j) <= bhp Then

        WellType(i) = "Pres"
        GoTo LoopAgain
    End If

Case "WRate"
    Qwn(j) = Qw(j) 'rcf/d
    If UCase(Prod_Inj(i)) = "PROD" Then
        Qon(j) = Qwn(j) * LambdaO(j) / LambdaW(j) 'rcf/d
    Else
        Qon(j) = 0
    End If
    Qtn(j) = Qon(j) + Qwn(j) 'rcf/d
    Pwfn(j) = Pn(j) - Qtn(j) / (Jmodel(i) * LambdaT(j)) 'psi

Case "Pres"
    Pwfn(j) = bhp 'psi
    If UCase(Prod_Inj(i)) = "PROD" And Pn(j) <= Pwfn(j) Then
        Qon(j) = 0: Qwn(j) = 0: Qtn(j) = 0: Jmodel(i) = 0
        Qo(j) = 0: Qw(j) = 0: Qt(j) = 0
        Stop
        GoTo Next_Well
    End If

    If UCase(Prod_Inj(i)) = "PROD" And Pn(j) > Pwfn(j) Then
        Qon(j) = Jmodel(i) * LambdaO(j) * (Pn(j) - Pwfn(j)) 'rcf/d
    End If

    Qwn(j) = Jmodel(i) * LambdaW(j) * (Pn(j) - Pwfn(j)) 'rcf/d
    Qtn(j) = Qon(j) + Qwn(j) 'rcf/d

    If Pwfn(j) < bhp Then Stop
    If Pwfn(j) > Pn(j) Then Stop
End Select
Next_Well:
Next i

End Sub

*****

Sub UpdateProperties()

```

```
Dim i As Integer
```

```
For i = 1 To NN
    Pold(i) = Pn(i)
    Soi(i) = Son(i)
    Swi(i) = Swn(i)
    Qo(i) = Qon(i)
    Qw(i) = Qwn(i)
    Qt(i) = Qtn(i)
    Pwf(i) = Pwfn(i)
    Vp(i) = Vpn(i)
Next i
```

```
End Sub
```

```
*****
```

```
Sub Print_Well(ByVal step As Integer, ByVal t As Double)
```

```
Dim i As Integer
With Sheets("Pwf")
    .Cells(i + 1, step + 1) = t
    For i = 1 To NN
        .Cells(i + 1, step + 1) = Pwfn(i)
    Next i
End With
```

```
End Sub
```

```
*****
```

```
Sub Print_ORate(ByVal step As Integer, ByVal t As Double)
```

```
Dim i As Integer

With Sheets("ORate")
    .Cells(i + 1, step + 1) = t
    For i = 1 To NN
        .Cells(i + 1, step + 1) = Qon(i) / Bon(i)
    Next
End With
```

```
End Sub
```



```
*****
```

```
Sub Print_WRate(ByVal step As Integer, ByVal t As Double)
Dim i As Integer
```

```
With Sheets("WRate")
    .Cells(i + 1, step + 1) = t
    For i = 1 To NN
        .Cells(i + 1, step + 1) = Qwn(i) / Bwn(i)
    Next
End With
```

```
End Sub
```

```
*****
```

```
Sub Print_TRate(ByVal step As Integer, ByVal t As Double)
Dim i As Integer
```

```
With Sheets("TRate")
    .Cells(i + 1, step + 1) = t
    For i = 1 To NN
        .Cells(i + 1, step + 1) = Qtn(i)
    Next
```

```
End With
```

```
End Sub
```

```
*****
```

```
Sub CalcNp()
Dim i As Integer, j As Integer, k As Byte, t As Double
Dim SumC1 As Byte: Dim SumC2 As Byte
Nt = 0
```

```
For i = 1 To NN
    Nt = Nt + Vpn(i) / Bon(i)
    If t_step <> 0 Then
        If Qon(i) > 0 Then Np = Np + Qon(i) / Bon(i) * delt
        If Qwn(i) < 0 Then Ni = Ni - Qwn(i) / Bwn(i) * delt
    End If
Next i
```

```
If t_step = 0 Then IOIP = Nt
  Mbe = Abs((IOIP + Ni - Nt - Np) / (IOIP + Ni) * 100)
```

```
End Sub
```

```
*****
```

```
Sub Print_MBE(ByVal step As Integer, ByVal t As Double)
```

```
With Sheets("FIP")
```

```
  .Cells(1, 2) = "Time,Days"
  .Cells(1, 3) = "OIP,rcf"
  .Cells(1, 4) = "WIP,rcf"
  .Cells(1, 5) = "OOIP,rcf"
  .Cells(1, 6) = "OWIP,rcf"
  .Cells(1, 7) = "Cum. Oil Recovery"
  .Cells(1, 8) = "Cum. Water Injected"
  .Cells(step + 1, 1) = step
  .Cells(step + 1, 2) = t
  .Cells(step + 1, 3) = CumOil
  .Cells(step + 1, 4) = CumWater
  .Cells(step + 1, 5) = TotalOIP
  .Cells(step + 1, 6) = TotalWIP
  .Cells(step + 1, 7) = CumOil / TotalOIP '* 100
  .Cells(step + 1, 8) = CumWater / TotalOIP '* 100
```

```
End With
```

```
End Sub
```

```
*****
```

```
Sub Cum_production()
```

```
Dim i As Integer
```

```
Dim CumO() As Double, CumW() As Double
```

```
'Cumulative Oil/Water Produced/Injected
```

```
For i = 1 To NN
```

```
  CumOil = CumOil + Qon(i) * delt
```

```
  CumWater = CumWater + Qwn(i) * delt
```

```
Next i
```

```
End Sub
```

```
*****
```

```

Sub Material_Balance()
Dim i As Integer

'MBE for every block
Call Cum_production

    For i = 1 To NN
        OOIPn(i) = Vpn(i) * Son(i) 'rcf
        WWIPn(i) = Vpn(i) * Swn(i) 'rcf

        If Qon(i) <> 0 Or Qwn(i) <> 0 Then
            MatBal(i) = (OOIPn(i) + WWIPn(i)) / (OOIP(i) + WWIP(i) - Qon(i) * delt +
Qwn(i) * delt)
            MatBalE(i) = (((OOIPn(i) + WWIPn(i)) / (OOIP(i) + WWIP(i) - Qon(i) * delt +
Qwn(i) * delt)) - 1) * 100
        Else
            MatBal(i) = (OOIPn(i) + WWIPn(i)) / (OOIP(i) + WWIP(i))
            MatBalE(i) = (((OOIPn(i) + WWIPn(i)) / (OOIP(i) + WWIP(i))) - 1) * 100
        End If
    Next i

End Sub

```

```

Sub Print_Result_MatBal(ByVal step As Integer, ByVal t As Double)
Dim i As Integer
With Sheets("MatBal")
    .Cells(i + 1, step + 1) = t

    For i = 1 To NN
        .Cells(i + 1, step + 1) = MatBal(i)
    Next i

End With

End Sub

```

```

Sub Calc_dxdy()
Dim i As Integer, im As Integer, ip As Integer
Dim j As Integer, jm As Integer, jp As Integer
Dim k As Integer

```

```

ReDim dx(NN)
ReDim dy(NN)
ReDim delx(NX, NY)
ReDim dely(NX, NY)

```

```

k = 1
For j = 1 To NY
  For i = 1 To NX

```

```

    im = i - 1: ip = i + 1
    jm = j - 1: jp = j + 1

```

```

    If i = 1 Then im = i
    If i = NX Then ip = NX

```

```

    If j = 1 Then jm = j
    If j = NY Then jp = NY

```

```

    delx(i, j) = (xD(i, j) - xD(im, j)) / 2 + (xD(ip, j) - xD(i, j)) / 2
    dx(k) = delx(i, j)

```

```

    dely(i, j) = (yD(i, j) - yD(i, jm)) / 2 + (yD(i, jp) - yD(i, j)) / 2
    dy(k) = dely(i, j)

```

```

    k = k + 1
  Next i
Next j

```

```

End Sub

```

```

'#####

```

```

Option Base 1

```

```

'*****

```

```

Sub bicgstab(x0() As Double, B() As Double, A() As Double, resErr As Double, ByVal
Itmax As Integer, x() As Double)
Dim n As Integer
Dim S() As Double, p() As Double, t() As Double
Dim ErrTol As Double
Dim r() As Double, hatr0() As Double

```

```

n = UBound(B):

ReDim rho(Itmax + 1)
ReDim r(n): ReDim hatr0(n)
ReDim S(n): ReDim p(n): ReDim t(n)

ErrTol = resErr * Norm(B())
x() = x0()
If Norm(x()) <> 0 Then
  For i = 1 To n
    AX = 0
    For j = 1 To rowFill(i).n

      AX = AX + A(i, rowFill(i).indexRow(j)) * x(rowFill(i).indexRow(j))
    Next
    r(i) = B(i) - AX
  Next
Else
  r() = B()
End If

hatr0() = r()
k = 0: rho(1) = 1: alpha = 1: omega = 1
ReDim v(n): ReDim p(n):

For i = 1 To n
  rho(2) = rho(2) + hatr0(i) * r(i)
Next

zeta = Norm(r()):

Do While ((zeta > ErrTol) And (k < Itmax - 1))
  k = k + 1

  If omega = 0 Then
    GoTo Err
  End If

  beta = (rho(k + 1) / rho(k)) * (alpha / omega)

  For i = 1 To n
    p(i) = r(i) + beta * (p(i) - omega * v(i))
  Next

```

```

tau = 0
For i = 1 To n
  AX = 0
  For j = 1 To rowFill(i).n
    AX = AX + A(i, rowFill(i).indexRow(j)) * p(rowFill(i).indexRow(j))
  Next
  v(i) = AX
  tau = tau + hatr0(i) * v(i)
Next

If tau = 0 Then
  GoTo Err
End If

alpha = rho(k + 1) / tau

For i = 1 To n
  S(i) = r(i) - alpha * v(i)
Next

tau = 0

For i = 1 To n
  AX = 0
  For j = 1 To rowFill(i).n
    AX = AX + A(i, rowFill(i).indexRow(j)) * S(rowFill(i).indexRow(j))
  Next
  t(i) = AX
  tau = tau + t(i) ^ 2
Next

If tau = 0 Then
  GoTo Err
End If

AX = 0
For i = 1 To n
  AX = AX + t(i) * S(i)
Next

omega = AX / tau

AX = 0
For i = 1 To n

```

```

    AX = AX + (hatr0(i) * t(i))
    x(i) = x(i) + alpha * p(i) + omega * S(i)
    r(i) = S(i) - omega * t(i)
Next

```

```

    rho(k + 2) = -omega * AX
    zeta = Norm(r())

```

```

Loop

```

```

Exit Sub
Err:
MsgBox "Error"
End Sub

```

```

*****

```

```

Function Norm(RR1() As Double)
Dim i As Integer, SumX As Double
Norm = 0
For i = 1 To UBound(RR1)
    Norm = Norm + RR1(i) ^ 2
Next
Norm = Norm ^ 0.5
End Function

```

VITA

Name: Emeline E. Chong

Permanent Address: 8B, Jalan Meritam,
96000 Sibul, Sarawak,
Malaysia.
E-mail: ak970097@hotmail.com

Educational Background: B.S., Petroleum Engineering
Malaysia University of Technology (UTM),
Skudai, Johor,
Malaysia.
(May 1997 – April 2001)

M.S., Petroleum Engineering
Texas A&M University,
College Station,
Texas, U.S.A.
(September 2002 - December 2004)