On-line Optimization-Based Simulators for Fractured and Non-fractured Reservoirs
DE-FC26-01BC15313
Final Report

September 2, 2001 to August 31, 2005

Milind D. Deo
Department of Chemical and Fuels Engineering
University of Utah
Salt Lake City, Utah 84112

Disclaimer: This report was prepared an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Contributors

Professor Craig Forster
Professor Mikhael Skliar
Dr. Yi-kun Yang
Yao Fu
Ganesh Balasubramaniam

Abstract

Oil field development is a multi-million dollar business. Reservoir simulation is often used to guide the field management and development process. Reservoir characterization and geologic modeling tools have become increasingly sophisticated. As a result the geologic models produced are complex. Most reservoirs are fractured to a certain extent. The new geologic characterization methods are making it possible to map features such as faults and fractures, field-wide. Significant progress has been made in being able to predict properties of the faults and of the fractured zones.

Traditionally, finite difference methods have been employed in discretizing the domains created by geologic means. For complex geometries, finite-element methods of discretization may be more suitable. Since reservoir simulation is a mature science, some of the advances in numerical methods (linear, nonlinear solvers and parallel computing) have not been fully realized in the implementation of most of the simulators.

The purpose of this project was to address some of these issues.

- One of the goals of this project was to develop a series of finite-element simulators to handle problems of complex geometry, including systems containing faults and fractures.

- The idea was to incorporate the most modern computing tools;use of modular object-oriented computer languages,the most sophisticated linear and nonlinear solvers, parallel computing methods and good visualization tools.

- One of the tasks of the project was also to demonstrate the construction of fractures and faults in a reservoir using the available data and to assign properties to these features.

- Once the reservoir model is in place, it is desirable to find the operating conditions, which would provide the best reservoir performance. This can be accomplished by utilization optimization tools and coupling them with reservoir simulation. Optimization-based reservoir simulation was one of the project goals.

- Providing remote access to the simulators developed was also one of the project objectives.

The basic methods development is presented in Chapters 1-3. Development of a flux continuous finite element algorithm is presented with example calculations in Chapter 1. This is followed by discussion of three-dimensional, three-phase development in Chapter 2. A different numerical method, the mixed finite element method is presented in Chapter 3. Verification of the methods developed is described in Chapter 4. Introduction to fractured reservoir simulation is provided in Chapter 5 with an example of a fractured reservoir simulation study of a faulted reservoir in North Sea. Chapter six contains several examples of two dimensional simulations, while chapter 7 contains examples of three-dimensional simulation. In Chapter 8 optimization techniques are discussed. Chapter 9 contains a roadmap to use the remote programming interface for the fractured reservoir simulator.

Contents

1 Formulation of the Control Volume Method **1**
1.1 Introduction 1
1.2 Governing equations 3
1.3 Upstream weighting in the finite difference formulation 4
1.4 Preliminaries 4
1.5 FEM and CVFE 5
1.6 The Control Volume Method 6
1.6.1 Control volume formulation 6
1.6.2 Control volume discretization 6
1.6.3 Formulation of partial residual functions 7
1.6.4 Upstream weighting 10
1.7 Flux continuity 11
1.7.1 CVM 12
1.7.2 FEM and CVFE 12
1.8 Mass conservation 14
1.9 Numerical experiments 15
1.9.1 Single triangle example 15
1.9.2 Five-spot injection problem 17
1.10 Conclusions 19

2 The Control Volume Finite Element Method **22**
2.1 Synopsis 22
2.2 Area Coordinate System and Interpolation Functions 22
2.3 Three-Dimensional, Two-Phase Control Volume Formulation 28
2.4 Three-Dimensional, Three-Phase Simulations 34

3 The Mixed Finite Element Method **38**
3.1 Introduction 38
3.1.1 The lowest order Raviart–Thomas space 38
3.2 Application of the MFEM in reservoir simulation 40
3.3 Multiphase MFE formulation 40

3.3.1 Multiphase flow equations 40
3.3.2 Discretization 41
3.3.3 Discretized multiphase flow equations 43
3.3.4 Flux continuity 44

4 Verification **45**
4.1 One-layer case study 45
4.2 Two-layer case study 45
4.3 Summary 46

5 Fractured Reservoir Simulation **61**
5.1 Reservoir Rock Fractures 61
5.2 Fracture Information from the Field 62
5.3 Single-Porosity Model 64
5.4 Discrete-Fracture Model 64
5.5 Case Studies 65

6 Examples **79**
6.1 Example 1 79
6.2 Example 2 79
6.3 Example 3 81

7 Example of Three-Dimensional Simulation **86**

8 OPTIMIZATION TECHNIQUE AND APPLICATIONS **93**
8.1 Description of the Optimization Problem 93
8.2 Overview of Nonlinear Optimization Techniques and Applications 94
8.2.1 Optimal Control Theory 94
8.2.2 Dynamic Programming and its Variations 95
8.2.3 Gradient Based Methods 96
8.2.4 Newton's Method and its Improvements 97
8.3 Optimization Methodology 99
8.3.1 The quasi-Newton method 99
8.3.2 Toolkit for Advanced Optimization 100
8.3.3 Formulation of the Cost Function 102
8.3.4 Gradient Computation 102
8.3.5 Optimization Algorithm 103
8.4 Case Studies 104
8.4.1 Case I 104
8.4.2 Case II 108
8.4.3 Case III 110

9 Parallel Computation **123**

10 The Reservoir Simulator Interface **125**
10.1 Introduction . . . 125
10.1.1 Overview . . . 125
10.1.2 Objects . . . 126
10.1.3 Design . . . 126
10.2 Interface Development . . . 128
10.2.1 Work Flows . . . 128
10.2.2 Features . . . 129
10.3 Installation . . . 129
10.3.1 Server . . . 129
10.3.2 Client . . . 130
10.3.3 Environments Requirement . . . 130
10.4 Client Side Reservoir Domain Data Input . . . 131
10.4.1 Start from a new domain . . . 131
10.4.2 Start from an existing domain . . . 131
10.4.3 Reservoir domain coordinates modifications . . . 132
10.5 XML Input File . . . 135
10.5.1 "*_ori.xml" file (rough domain file) . . . 136
10.5.2 CVFE Simulation XML input file ("*_fin.xml" file) . . . 140
10.6 Operations . . . 141
10.6.1 Write property informations into the XML file . . . 141
10.6.2 Mesh the rough domain with reservoir features . . . 145
10.6.3 Simulate the fine domain . . . 145
10.7 Ancillary Programs . . . 145
10.7.1 Triangle Mesh Viewer . . . 146
10.7.2 Results Images Viewer . . . 146
10.7.3 XML Source File Viewer . . . 146
10.7.4 Interface Input/Output Console . . . 146
10.7.5 Some Defintions . . . 151

List of Figures

1.1 An example control volume mesh. The triangulation $\mathcal{T}$ is in solid lines and the control volumes $\mathcal{B}$ are in dashed lines. 5
1.2 A control volume with its boundaries across several triangular elements. 7
1.3 Decomposition of a control volume into several subvolumes. 7
1.4 Unit outward normals of subvolume $b_{i,m}$ in triangle t_m. 8
1.5 A schematic representation of the applicable upstream nodes and flux directions. 10
1.6 An illustration of the concept of flux continuity. 11
1.7 Acute, right-angled, and obtuse triangles used in the discussion of the consequences of applying either the potential- or flux-based upstream condition. 16
1.8 Five-spot injection prodcution pattern used as an example calculation problem. 17
1.9 Legend for water contour plots. 20
1.10 Comparison of the water saturation contours of the diagonal grid problem solved by the CVM (left) and the CVFE (right). From top to bottom are the contours at 0.1, 0.2 and 0.4 domain pore volume of water injected. 20
1.11 Comparison of the water saturation contours of the parallel grid problem solved by the CVM (left) and the CVFE (right). From top to bottom are the contours at 0.1, 0.2 and 0.4 domain pore volume of water injected. 21
1.12 Comparison of the oil production rates for the CVM and the CVFE on the diagonal and the parallel grids. 21

2.1 Definition of the natural coordinate of a tetrahedral element . . . 24
2.2 A tetrahedron element with associated control volumes 28

3.1 Unit outward normals on triangle's three edges. 39

4.1 The domain measures 1000 ft in the x-direction, 500 ft in the y-direction and 50 ft in the z-direction. The horizontal production well is represented by the the orange line and the horizontal injection well is represented by the blue line. Wells are placed at the center of the domain along the x-direction, and each well measures 800 ft in length. 47
4.2 The mesh used for the one-layer case study for the UFES. 47
4.3 Cumulative oil production for the one-layer case study. 48
4.4 A more detailed cumulative oil production plot for the one-layer case study (between zero and 1000 days). 48
4.5 Oil production rate for the one-layer case study. 49
4.6 A more detailed oil production plot for the one-layer case study (between 91 and 200 days). Notice that the y-axis has a higher resolution. 49
4.7 Water cut for the one-layer case study. 50
4.8 Water cut data between 91 and 200 days for the one-layer case study. Notice that the y-axis has a higher resolution 50
4.9 Gas oil ratio for the one-layer case study. 51
4.10 Gas oil ratio between 91 and 130 days for the one-layer case study. 51
4.11 The mesh used for the two-layer case study for the UFES. 52
4.12 Cumulative oil production for the two-layer case study. 53
4.13 Cumulative oil production curve between 0 and 1000 days for the two-layer case study. 53
4.14 Oil production rate for the two-layer case study. 54
4.15 Oil production rate between 91 and 200 days for the two-layer case study. 54
4.16 Water cut for the two-layer case study. 55
4.17 Water cut between 91 and 200 days for the two-layer case study. . 55
4.18 Gas oil ratio for the two-layer case study. 56
4.19 Gas oil ratio between 91 and 130 days for the two-layer case study. 56
4.20 Cumulative oil production for the two-layer case study. 57
4.21 Cumulative oil production curve between 0 and 1000 days for the two-layer case study. 57
4.22 Oil production rate for the two-layer case study. 58
4.23 Oil production rate between 91 and 200 days for the two-layer case study. 58
4.24 Water cut for the two-layer case study. 59
4.25 Water cut between 91 and 200 days for the two-layer case study. . 59
4.26 Gas oil ratio for the two-layer case study. 60
4.27 Gas oil ratio between 91 and 130 days for the two-layer case study. 60

5.1 Cross-section view of a fault. 62
5.2 Areal view of some joints. 63

5.3 Cross-section view of the formation of a joint caused by overburden pressure. 63
5.4 The formation of a pair of conjugated fractures due to tectonic force. 63
5.5 A fractured domain Ω. 66
5.6 The original fractured domain. 66
5.7 The fractured domain after the fracture is replaced by a rectangle; the width of the fracture has been enlarged for visibility. 67
5.8 Triangular mesh of the domain using the single-porosity model; there are 415 nodes and 780 triangles in this particular mesh. . . . 67
5.9 Triangular mesh of the domain using the discrete-fracture model; there are 97 nodes and 160 triangles in this particular mesh. . . . 68
5.10 The fractured domain. Different matrix and fracture properties are shown in the legend. 68
5.11 The incorporation of the outer domain to simulate the three flowing boundaies. 69
5.12 The placement of injection and production wells for Case I; the blue and red lines are the horizontal injection and production wells, respectively. 69
5.13 The placement of injection and production wells for Case II; the blue and red lines are the horizontal injection and production wells, respectively. 70
5.14 The placement of injection and production wells for Case III; the blue and red lines are the horizontal injection and production wells, respectively. 71
5.15 The placement of injection and production wells for Case IV; the blue and red lines are the horizontal injection and production wells, respectively. 71
5.16 Water saturation distribution of Case I at 1000 days. 72
5.17 Water saturation distribution of Case II at 1000 days. 72
5.18 Water saturation distribution of Case III at 1000 days. 73
5.19 Water saturation distribution of Case IV at 1000 days. 73
5.20 Water saturation distribution of Case I at 5000 days. 74
5.21 Water saturation distribution of Case II at 5000 days. 74
5.22 Water saturation distribution of Case III at 5000 days. 75
5.23 Water saturation distribution of Case IV at 5000 days. 75
5.24 Water saturation distribution of Case I at 9000 days. 76
5.25 Water saturation distribution of Case II at 9000 days. 76
5.26 Water saturation distribution of Case III at 9000 days. 77
5.27 Water saturation distribution of Case IV at 9000 days. 77
5.28 Cumulative oil production versus time for the four case studies. . 78

6.1 Plan view of a domain with intersecting fractures is shown. Water injectors are in blue and producers are shown in red. 80

6.2 Water saturations as a result of a waterflood in a system with negative rock matrix capillary pressure and zero fracture capillary pressure 80
6.3 Water saturations as a result of a waterflood in a system with positive rock matrix capillary pressure and zero fracture capillary pressure 81
6.4 Complex domain with two sets of intersecting fractures (white lines), deviating horizontal injectors (blue lines) and horizontal producers (red lines) is shown. 82
6.5 Waterflood at an early time in the domain shown in Figure 6.4 is presented. 82
6.6 Waterflood at a late time in the domain shown in Figure 6.4 is presented. 83
6.7 A common waterflood unit in the Greater Monument Butte field. The well spacing is 40 acres. All the wells are hydraulically fractured. Slightly different fracture orientations and fracture half lengths of 200 feet area are assumed 84
6.8 Waterflood at an early time in the domain shown in Figure 6.7 is presented. 84
6.9 Waterflood at a a late time in the domain shown in Figure 6.7 is presented. 85

7.1 The three-dimensional domain showing the fractures and the tetrahedral mesh 87
7.2 Another view of the three-dimensional domain with fractures and the mesh created 87
7.3 Water saturations in the three-dimensional domain with fractures after 181 days of injection 88
7.4 Water saturations at 181 days. Y-Z cross-sections at x values of 174, 380, 488 and 593 feet are shown. 88
7.5 A plan view (X-Y section) at z=10 feet. Water saturations are after 181 days of injection are shown. 89
7.6 A plan view (X-Y section) at z=40 feet. Water saturations after 181 days of injection are shown. 89
7.7 Water saturations at 1003 days. Y-Z cross-sections at x values of 174, 380, 488 and 593 feet are shown. 90
7.8 A plan view (X-Y section) at z=25 feet. Water saturations are after 1003 days of injection are shown. 90
7.9 Water saturations in the three-dimensional domain with fractures after 181 days of injection; capillary pressures in the rock matrix are positive 91
7.10 Water saturations for positive capillary pressure in the rock matrix at 181 days. Y-Z cross-sections at x values of 174, 380, 488 and 593 feet are shown. 92

7.11 A plan view (X-Y section) at z=25 feet. Water saturations are after 1003 days of injection and the rock matrix capillary pressure is positive 92

8.1 Finite-element mesh of the domain used in Case I. 105
8.2 *Study Ib:* Cost function for a two-stage optimization problem. . . 106
8.3 Finite-element mesh of the fractured domain used in Case II. . . . 109
8.4 Relative Permeabilities of the fluids for domain in Case II. 110
8.5 Capillary Pressures in the matrix and fractures for domain in Case II. 111
8.6 Diagrammatic description of the optimization method developed by Yeten *et al.* 112
8.7 Finite-element mesh of the domain used in Case III. 113
8.8 *Study Ia:* Cumulative oil production comparison. 115
8.9 *Study Ia:* Water cut comparison. 116
8.10 *Study Ib:* Cumulative oil production comparison for the two-stage problem. 116
8.11 *Study Ib:* Water cut comparison for the two-stage problem. 117
8.12 *Study Ib:* Cumulative oil production comparison for the five-stage problem. 117
8.13 *Study Ib:* Water cut comparison for the five-stage problem. 118
8.14 *Study Ic:* Cumulative oil production comparison. 118
8.15 *Study Ic:* Water cut comparison. 119
8.16 *Study Id:* Cumulative oil production comparison. 119
8.17 *Study Id:* Water cut comparison. 120
8.18 *Study IIa:* Cumulative oil production comparison. 120
8.19 *Study IIa:* Water cut comparison. 121
8.20 *Study IIb:* Cumulative oil production for 5000 days. 121
8.21 *Study IIb:* Water cut for 5000 days. 122

9.1 Scaleup performance on a parallel linux cluster. A two-dimensional, 250,000 node problem was tested. 124

10.1 Open the Client Side Interface. 147
10.2 Draw a simple domain first. 148
10.3 Draw fractures, injection wells and production wells. 149
10.4 Draw additional fractures, injection wells and production wells. . . 150
10.5 Save the reservoir domain information. 151
10.6 Open a save the reservoir domain. 152
10.7 The opened reservoir domain. 153
10.8 The modified reservoir domain. 154
10.9 The *_ori.xml file. 155
10.10 The modified *_ori.xml file. 156
10.11 The snapshot of “simpleExample1_ori.xml”. 157

10.12 The snapshot of meshed “simpleExample1_ori.xml” 158
10.13 The snapshot of “simpleExample1_ori.xml” 159
10.14 The snapshot of “simpleExample1_ori.xml” with its editor. 160
10.15 The snapshot of meshed “simpleExample1_ori.xml” with its editor. 161
10.16 The middle snapshot of meshed “simpleExample1_ori.xml” simulation (110days). 162
10.17 The final snapshot of meshed “simpleExample1_ori.xml” simulation (580days). 163
10.18 Triangle Mesh Shower V1.0. 164
10.19 Result Images Viewer V1.0. 165
10.20 XML Source File Viewer V1.0. 166
10.21 Interface Input/Output Console V1.0. 167

Chapter 1

Formulation of the Control Volume Method

A flux continuous, locally mass conservative control volume method for the solution of multiphase flow problems in porous media is proposed. This method which involves adaptation of a specific upstream weighting technique is an improvement over previous control volume finite element methods, which were not flux continuous. The previous implementation of upstream weighting in the control volume finite element method required a positive transmissibility condition. Current implementation does not require this condition. Mass conservation at the local level (control volume) has always been an issue in solutions of multiphase flow problems using finite element family of methods. It is shown that when a proper set of control volumes is considered, these numerical methods yields locally mass conservative solutions. Numerical examples that demonstrate the significance of flux continuity are presented.

1.1 Introduction

The governing equations for multiphase fluid flow in porous media are mass conservation of phases [1, 2]. Consider a two-phase, immiscible flow problem,

$$-\nabla \cdot \rho_l \underline{v}_l = \frac{\partial \left(\phi \rho_l S_l\right)}{\partial t} + \tilde{q}_l, \tag{1.1}$$

where $l = \{n, w\}$ represents non-wetting and wetting phases, respectively, ρ_l is the phase density, ϕ is the porosity, S_l is the phase saturation, and $\tilde{q}_l$ is the source term. Darcy's law is used for the flux calculation,

$$\underline{v}_l = -\underline{\underline{k}} \lambda_l \nabla \varphi_l, \tag{1.2}$$

where $\underline{\underline{k}}$ is the rock permeability tensor. Here the phase mobility, $\lambda_l = k_{rl}(S_L)/\mu_l$, is the ratio of phase relative permeability to the phase viscosity, and φ_l is the phase potential.

After (1.2) is substituted into (1.1) and discretized by the numerical method of choice, averaged $\underline{\underline{k}}$, ρ_l, μ_l, and k_{rl} between discretized members need to be determined. To ensure flux continuity between these members the harmonic average of $\underline{\underline{k}}$ is used [1]. In general, ρ_l and μ_l are weak functions of phase pressure; therefore, the arithmetic average is used. The choice of k_{rl} is very important in ensuring that the numerical solution converges to the real solution. It has been argued that use of any type of average for k_{rl} is inappropriate and that upstream implementation is essential [3]. Use of asymmetric weighting functions for k_{rl} are possible [4], but that does not guarantee physically meaningful phase saturation values [5].

Typically, finite difference is the method of choice for discretization in reservoir simulation applications. However, the finite element family of methods are more suitable for problems with complex geometrical domains. The finite element method (FEM) gained popularity in the field of oil reservoir simulation in the late 1960's [6, 7]. Langsrud [8] and Dalen [9] developed the most essential features required for the success of this family of methods in multiphase flow applications—mass-lumping and upstream weighting of k_{rl}. These features are still part of current practice [10, 11, 12, 13]. The mass-lumping scheme is necessary to prevent oscillating phase saturation values. The upstream weighted k_{rl} is determined by a *potential-based* upstream weighting method. In this method, the flux portion of (1.1) is first rearranged to resemble the finite difference formulation, and then by comparing pairs of potential values, the upstream k_{rl} is determined.

The control volume finite element method (CVFE) for oil reservoir simulation was introduced by Forsyth[5]. The upstream implementation strategy for k_{rl} in the CVFE is the potential-based method as in FEM. This has been adopted by other researches [14, 15, 16, 17, 18]. In this method, shapes of the triangular elements should be such that positive transmissibilities are guaranteed.

To distinguish from the CVFE, the method proposed here is called the control volume method (CVM). The only difference between the CVM and the CVFE is the upstream weighting scheme. Unlike the CVFE, the upstream direction in the CVM is determined by the flux direction across each control volume boundary in each triangle. This type of upstream weighting implementation is considered *flux-based*. This idea was first developed by Prakash[19] in single-phase solute transport applications. Forsyth[5] recognized this concept for multiphase flow but applied the potential-based method. A box method using a similar flux-based upstream weighting was proposed by Huber and Helmig[20, equation (18)]. In the box method, the flux direction is determined by the summation of all the fluxes exchanged between two control volumes across multiple triangular or quadrilateral elements.

One of the main objectives of this paper is to highlight the differences between flux- and potential-based upstream weighting methods. It is clearly shown that the flux-based implementation leads to flux continuous solutions, and eliminates limitations imposed on the shapes of the elements.

One of the main reasons for the lack of widespread use of the finite element

family of methods in reservoir simulation is due to the belief that they do not yield locally conservative solutions. In this paper, proofs are provided regarding the local mass conservative aspects of the CVFE, the CVM, and the FEM methods.

In §1.2 the governing equations of multiphase oil reservoir simulation are further discussed. In §1.3, the upstream weighting of properties is explained in the context of the finite difference method. The discretized finite element and control volume finite element formulations are discussed in §1.5. The proposed control volume method is derived in §1.6. Flux continuity and mass conservation properties of the above three methods are discussed in §1.7 and §1.8, respectively. In §1.9, numerical experiments are conducted using the CVFE and CVM to show the significance of flux continuity.

1.2 Governing equations

We consider a bounded polygonal domain Ω in $\mathbb{R}^2$ with boundary Γ. The governing equations are obtained by substituting (1.2) into (1.1).

$$0 = -\nabla \cdot \underline{\underline{k}}\rho_l \lambda_l \nabla \varphi_l + \frac{\partial(\phi \rho_l S_l)}{\partial t} + \tilde{q}_l. \tag{1.3}$$

The permeability tensor $\underline{\underline{k}}$ is symmetric positive definite [2]. The initial conditions are prescribed phase potential and saturation distributions in the domain Ω

$$\varphi_l(\Omega, t = 0) = \varphi_{l0}(\Omega), \quad S_l(\Omega, t = 0) = S_{l0}(\Omega). \tag{1.4}$$

The boundary condition is

$$\underline{v}_l \cdot \hat{\underline{n}}\Big|_\Gamma = 0, \tag{1.5}$$

where $\hat{\underline{n}}$ is the unit outward normal on Γ. Phase potential φ_l is defined as

$$\varphi_l = P_l + \rho_l \frac{g}{g_c} z.$$

where P_l is the phase pressure, g is the gravitational acceleration constant, g_c is a conversion constant and z is the elevation. To emphasize the basic ideas, we will consider only cases where $z = 0$, that is $\varphi_l = P_l$. Phase pressures are related by capillary pressure

$$P_c(S_w) = P_n - P_w. \tag{1.6}$$

and phase saturations are coupled by

$$S_n + S_w = 1. \tag{1.7}$$

Equations (1.3) to (1.7) form the complete two-phase flow problem.

1.3 Upstream weighting in the finite difference formulation

The basic idea of upstream weighting is to choose the property corresponding to the upstream direction of the flux [1]. Using the finite difference formulation as an example, the x-direction flux between any pair of adjacent blocks, i and j, can be written as

$$f_{ij} = -Lk_{ij}\rho_{ij}\lambda_{ij}\frac{\partial\varphi}{\partial x}, \tag{1.8}$$

where L is the length of the common boudary between the pair of blocks and the derivative of phase potential is approximated by $(\varphi_j - \varphi_i)/\Delta x$. The subscripts "$ij$" on the right hand side of (1.8) denote that the type of averaging methods used as discussed in §1.1 to calculate the values of k, ρ, and λ. Rewriting (1.8) with the proper averaging method indicated by subscrips, we have

$$f_{ij} = -Lk_{\text{har}}\rho_{\text{ari}}\frac{k_{r,\text{up}}}{\mu_{\text{ari}}}\frac{\varphi_j - \varphi_i}{\Delta x}. \tag{1.9}$$

where "har" stands for harmonic average, "ari" stands for arithmetic average, and "up" for upstream weighting. Notice that some authors apply upstream weighting to the whole mobility λ, and some others apply upstream weighting to ρ also. In this paper we apply upstream weighting to both ρ and λ; therefore, (1.9) becomes

$$f_{ij} = -Lk_{\text{har}}\rho_{\text{up}}\lambda_{\text{up}}\frac{\varphi_j - \varphi_i}{\Delta x}. \tag{1.10}$$

The upstream properties are then determined as follows. Consider the common boundary between blocks i and j

1. when $\varphi_i > \varphi_j$, the flux is from block i to block j; therefore, up $= i$;
2. when $\varphi_i < \varphi_j$, the flux is from block j to block i; therefore, up $= j$.

For the finite difference method, the transmissibility, $L/\Delta x$, is always positive. As a result, the potential- and flux-based upstream weighting schemes coincide.

1.4 Preliminaries

Referring to Figure 1.1, let $\mathcal{T} = \{t\}$ denote a regular partition of the domain Ω into triangular elements. The dual mesh (control volumes) $\mathcal{B} = \{b\}$ of $\mathcal{T}$ is constructed by connecting the barycenter and the midpoint of sides of every triangle $t \in \mathcal{T}$ with straight lines.

Let $\mathcal{P}^1(\mathcal{T})$ denote the space of continuous piecewise linear polynomials associated with $\mathcal{T}$. The usual nodal basis for $\mathcal{P}^1(\mathcal{T})$ is denoted by $\{L_i\}$, which satisfies

$$L_i(v_j) = \delta_{ij}$$

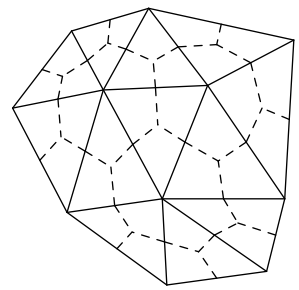

Figure 1.1: An example control volume mesh. The triangulation $\mathcal{T}$ is in solid lines and the control volumes $\mathcal{B}$ are in dashed lines.

where v_j is a vertex in the triangulation. Let $\mathcal{P}^0(\mathcal{B})$ denote the space of discontinuous piecewise constants with respect to $\mathcal{B}$. Define

$$\mathcal{P}^0_+(\mathcal{T}) = \{c \in \mathcal{P}^0(\mathcal{T}) : \ c > 0\}$$

to be the space of discontinuous piecewise positive constants with respect to $\mathcal{T}$.

1.5 FEM and CVFE

The similarity of linear finite elements and the box method for Laplacian problems has been discussed [21]. It has been shown that the formulation derived from linear finite elements and the CVFE are the same for incompressible single phase flow problems [18].

The multiphase finite element formulation used in the field of oil reservoir simulation finds the approximations of phase potential and phase saturation in $\mathcal{P}^1(\mathcal{T})$. The mass-lumping scheme is applied to stabilize the phase saturation values. As a result of mass-lumping, we can consider that the phase saturation solution is sought in $\mathcal{P}^0(\mathcal{B})$. Therefore, a set of control volumes is implied in the FEM. The CVFE method also finds its discretized phase potential in $\mathcal{P}^1(\mathcal{T})$ and phase saturation in $\mathcal{P}^0(\mathcal{B})$. If the potential-based upstream weighting scheme is applied to the FEM and the CVFE, and both ρ and λ are taken to be the upstream weighted values then the formulations derived from both methods are exactly the same. As a result, the control volumes implied in the FEM are the same as the control volumes $\mathcal{B}$ defined in the CVFE.

For both methods, we consider both $\underline{k}$ and ϕ are in $\mathcal{P}^0_+(\mathcal{T})$. The three residual functions for any triangle $t \in \mathcal{T}$ derived by both methods are

$$\begin{aligned} F_i = &- A\Big[\big(\rho\lambda\big)_{\mathrm{up}(ij)} T_{ij}(\varphi_j - \varphi_i) + \big(\rho\lambda\big)_{\mathrm{up}(ik)} T_{ik}(\varphi_k - \varphi_i)\Big] \\ &+ \frac{A}{3}\frac{\partial(\phi\rho_i S_i)}{\partial t} \qquad (i,j,k = 1,2,3 \quad \text{and} \quad i \neq j \neq k), \end{aligned} \tag{1.11}$$

where A is the area of the triangle. Note that the subscript l and the source term

in (1.3) are omitted for simplicity. The transmissibility is defined as

$$T_{ij} = -\Big(k_{xx}\frac{\partial L_i}{\partial x}\frac{\partial L_j}{\partial x} + k_{xy}\frac{\partial L_i}{\partial x}\frac{\partial L_j}{\partial y} + k_{yx}\frac{\partial L_i}{\partial y}\frac{\partial L_j}{\partial x} + k_{yy}\frac{\partial L_i}{\partial y}\frac{\partial L_j}{\partial y}\Big). \tag{1.12}$$

The potential-based upstream operator is defined by

$$\mathrm{up}(ij) = \begin{cases} i & \text{if } \varphi_i > \varphi_j, \\ j & \text{if } \varphi_i < \varphi_j. \end{cases} \tag{1.13}$$

Notice that when the two flux terms of (1.11) are considered separately there is no significant physical meaning associated with either one of them. It is arranged so only to resemble (1.10).

A *positive transmissibility condition* is necessary to guarantee $T_{ij} > 0$ [16]. Negative transmissibilities are physically unrealistic and also produce unacceptable saturation values.

1.6 The Control Volume Method

In this section, we derive the CVM from a finite element point of view with a focus on the explict expression for local fluid flux.

The basic concept of the CVM is to use the fluid potential values on $\mathcal{T}$ for flux calculation; the flux so obtained is then used for mass balance on $\mathcal{B}$. Take any triangle $t \in \mathcal{T}$ as an example; after establishing the flux direction in the triangle, the fluid exchanged between the three control volumes in the triangle can be calculated.

1.6.1 Control volume formulation

The residual function for a control volume $b_i \in \mathcal{B}$ with boundary Γ^i is obtained by taking the integrated form of (1.1):

$$\begin{aligned} F^i = 0 &= \int_{b_i} \nabla \cdot \rho \underline{v} + \frac{\partial(\phi\rho S)}{\partial t}\, d\underline{x} \\ &= \int_{\Gamma^i} \rho \underline{v} \cdot \underline{\hat{n}}\, ds + \int_{b_i} \frac{\partial(\phi\rho S)}{\partial t}\, d\underline{x} \end{aligned} \tag{1.14}$$

Here $\underline{\hat{n}}$ is the unit outward normal on Γ^i. Note that the subscript l and the source term in (1.3) are omitted for simplicity.

1.6.2 Control volume discretization

During the process of computation, it is difficult to evaluate (1.14) because a control volume is usually distributed accross several triangular elements as shown

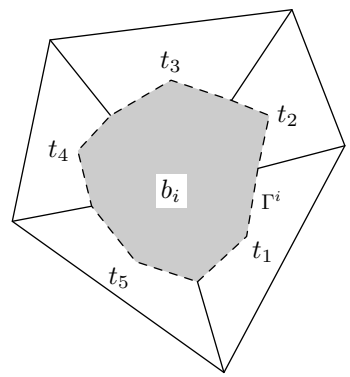

Figure 1.2: A control volume with its boundaries across several triangular elements.

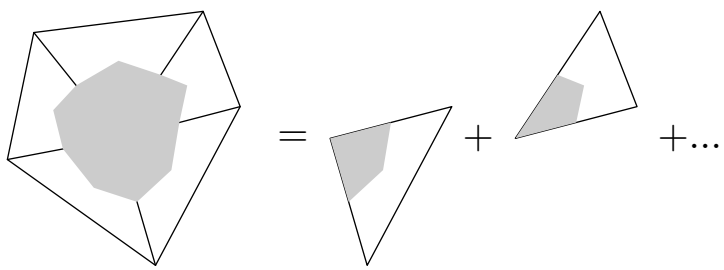

Figure 1.3: Decomposition of a control volume into several subvolumes.

in Figure 1.2. A more convenient approach is then to use an element-by-element method to add up the contributions from subvolumes $b_{i,m} = b_i \cap t_m$. Figure 1.3 shows this concept. The residual function for the control volume b_i in Figure 1.2 can then be obtained by

$$F^i = \sum_{m=1}^{5} F^i_m. \tag{1.15}$$

The partial residual function F^i_m represents the part of F^i which is contributed by $b_{i,m}$ and is defined as

$$F^i_m = \int_{\Gamma^i_m} \rho \underline{v} \cdot \underline{\hat{n}} \, ds + \int_{b_{i,m}} \frac{\partial(\phi \rho S)}{\partial t} \, d\underline{x}, \tag{1.16}$$

where $\Gamma^i_m = \Gamma^i \cap t_m$.

1.6.3 Formulation of partial residual functions

As shown in Figure 1.4, the partial residual function F^i_m of $b_{i,m}$ is derived in this section. The same procedure can be applied to obtain F^j_m and F^k_m.

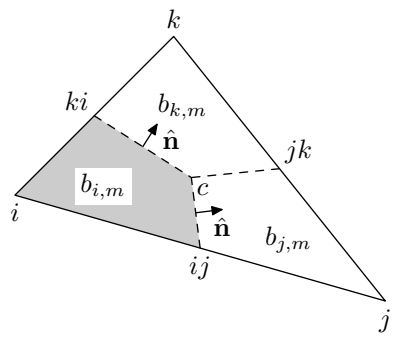

Figure 1.4: Unit outward normals of subvolume $b_{i,m}$ in triangle t_m.

Equation (1.16) is rewritten for $b_{i,m}$ as

$$\begin{aligned} F_m^i &= \int_{\overline{cij}+\overline{cki}} \rho \underline{v} \cdot \underline{\hat{n}}\, ds + \int_{b_{i,m}} \frac{\partial(\phi\rho S)}{\partial t}\, d\underline{x} \\ &= \int_{\overline{cij}} \rho \underline{v} \cdot \underline{\hat{n}}\, ds + \int_{\overline{cki}} \rho \underline{v} \cdot \underline{\hat{n}}\, ds + \int_{b_{i,m}} \frac{\partial(\phi\rho S)}{\partial t}\, d\underline{x} \end{aligned} \tag{1.17}$$

where $\underline{\hat{n}}$ is the unit outward normal of the corresponding boundary as shown in Figure 1.4. Define the fluxes flowing out of $b_{i,m}$ through $\overline{cij}$ and $\overline{cki}$ as

$$f_{i,\overline{cij}} = \int_{\overline{cij}} \rho \underline{v} \cdot \underline{\hat{n}}\, ds \quad \text{and} \quad f_{i,\overline{cki}} = \int_{\overline{cki}} \rho \underline{v} \cdot \underline{\hat{n}}\, ds,$$

respectively. Also, define the total flux flowing out of $b_{i,m}$ as

$$f_i = f_{i,\overline{cij}} + f_{i,\overline{cki}},$$

then equation (1.17) becomes

$$F_m^i = f_i + \int_{b_{i,m}} \frac{\partial(\phi\rho S)}{\partial t}\, d\underline{x}. \tag{1.18}$$

Notice that f_i in equation (1.18) represents the flux portion of the partial residual function.

The flux across $\overline{cij}$ is evaluated as

$$\begin{aligned} f_{i,\overline{cij}} &= \int_{\overline{cij}} \rho \underline{v} \cdot \underline{\hat{n}}\, ds = \int_{\overline{cij}} \rho\big(v_x\, \underline{\hat{i}} + v_y\, \underline{\hat{j}}\big) \cdot \big(n_x\, \underline{\hat{i}} + n_y\, \underline{\hat{j}}\big)\, ds \\ &= \int_{\overline{cij}} \rho(v_x n_x + v_y n_y)\, ds, \end{aligned} \tag{1.19}$$

where

$$v_x = -\lambda_{\text{up}}\Big(k_{xx}\frac{\partial\varphi}{\partial x} + k_{xy}\frac{\partial\varphi}{\partial y}\Big), \qquad v_y = -\lambda_{\text{up}}\Big(k_{yx}\frac{\partial\varphi}{\partial x} + k_{yy}\frac{\partial\varphi}{\partial y}\Big).$$

To obtain the unit outward normal $\underline{\hat{n}}$ along a line $\overline{\alpha\beta}$, a transformation matrix is introduced

$$T_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{1.20}$$

By multipling T_θ to any vector $\underline{v}$, a new vector $\underline{v}_\theta$ rotated θ degree with respect to $\underline{v}$ is obtained. Let $\overrightarrow{\alpha\beta} = (x_\beta - x_\alpha, y_\beta - y_\alpha)$ represent the vector from node α to node β; then

$$\underline{\hat{n}}_\theta = T_\theta \, \frac{\overrightarrow{\alpha\beta}}{\|\overrightarrow{\alpha\beta}\|} \tag{1.21}$$

is the unit vector formed by rotating $\overrightarrow{\alpha\beta}$ by θ degrees. To find the unit outward normal $\underline{\hat{n}}$ of the control volume $b_{i,m}$ along $\overline{cij}$, we let $\theta = -\pi/2$ then,

$$\underline{\hat{n}} = T_{\frac{-\pi}{2}} \frac{\overleftarrow{cij}}{\|\overleftarrow{cij}\|} = \frac{(y_c - y_{ij})}{\|\overleftarrow{cij}\|}\,\underline{\hat{i}} + \frac{(x_{ij} - x_c)}{\|\overleftarrow{cij}\|}\,\underline{\hat{j}}$$

where $\overleftarrow{cij} = (x_c - x_{ij}, y_c - y_{ij})$. Therefore,

$$\begin{aligned} f_{i,\overline{cij}} = \int_{\overline{cij}} \rho_{\text{up}}\lambda_{\text{up}} \Bigg[&-\Big(k_{xx}\frac{\partial\varphi}{\partial x} + k_{xy}\frac{\partial\varphi}{\partial y}\Big)\frac{(y_c - y_{ij})}{\|\overleftarrow{cij}\|} \\ &- \Big(k_{yx}\frac{\partial\varphi}{\partial x} + k_{yy}\frac{\partial\varphi}{\partial y}\Big)\frac{(x_{ij} - x_c)}{\|\overleftarrow{cij}\|}\Bigg]\, ds. \end{aligned} \tag{1.22}$$

The phase potential in (1.22) is approximated by $\varphi_h \in \mathcal{P}^1(\mathcal{T})$ and

$$\varphi_h(\underline{x}) = L_i(\underline{x})\varphi_i + L_j(\underline{x})\varphi_j + L_k(\underline{x})\varphi_k$$

where φ_i, φ_j, and φ_k are the phase potential values at triangular vertices. Consequentlly, the derivatives of φ_h are constants in $t \in \mathcal{T}$; therefore, (1.22) can be written as

$$\begin{aligned} f_{i,\overline{cij}} = \rho_{\text{up}}\lambda_{\text{up}} \Big[&-\Big(k_{xx}\frac{\partial\varphi}{\partial x} + k_{xy}\frac{\partial\varphi}{\partial y}\Big)(y_c - y_{ij}) \\ &- \Big(k_{yx}\frac{\partial\varphi}{\partial x} + k_{yy}\frac{\partial\varphi}{\partial y}\Big)(x_{ij} - x_c)\Big]. \end{aligned} \tag{1.23}$$

The flux $f_{i,\overline{cki}}$ can be derived in the same way as $f_{i,\overline{cij}}$ and as a result f_i is fully defined. Let S be approximated by $S_h \in \mathcal{P}^0(\mathcal{B})$ then $S_h(b_i) = S_i$ where S_i is the saturation value of control volume b_i. The partial residual function of $b_{i,m}$ can then be written as

$$F_m^i = f_i + b_{i,m}\frac{\partial(\phi\rho_i S_i)}{\partial t}, \tag{1.24}$$

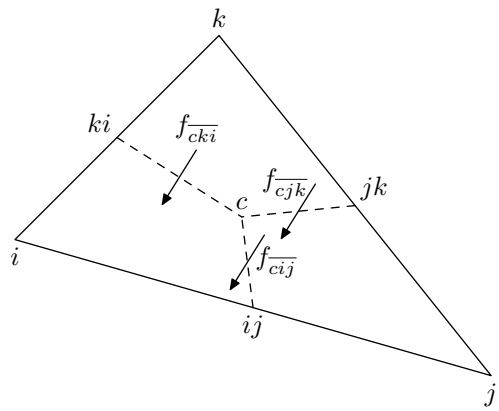

Figure 1.5: A schematic representation of the applicable upstream nodes and flux directions.

where $b_{i,m} = \int_{b_{i,m}} d\underline{x}$. Other fluxes can be calculated in the same way and are summarized here.

$$f_{i,\overline{cij}} = -f_{j,\overline{cij}} = \rho_{\text{up}}\lambda_{\text{up}}\Big[-\Big(k_{xx}\frac{\partial\varphi}{\partial x} + k_{xy}\frac{\partial\varphi}{\partial y}\Big)(y_c - y_{ij}) - \Big(k_{yx}\frac{\partial\varphi}{\partial x} + k_{yy}\frac{\partial\varphi}{\partial y}\Big)(x_{ij} - x_c)\Big], \tag{1.25a}$$

$$f_{j,\overline{cjk}} = -f_{k,\overline{cjk}} = \rho_{\text{up}}\lambda_{\text{up}}\Big[-\Big(k_{xx}\frac{\partial\varphi}{\partial x} + k_{xy}\frac{\partial\varphi}{\partial y}\Big)(y_c - y_{jk}) - \Big(k_{yx}\frac{\partial\varphi}{\partial x} + k_{yy}\frac{\partial\varphi}{\partial y}\Big)(x_{jk} - x_c)\Big], \tag{1.25b}$$

$$f_{k,\overline{cki}} = -f_{1,\overline{cki}} = \rho_{\text{up}}\lambda_{\text{up}}\Big[-\Big(k_{xx}\frac{\partial\varphi}{\partial x} + k_{xy}\frac{\partial\varphi}{\partial y}\Big)(y_c - y_{ki}) - \Big(k_{yx}\frac{\partial\varphi}{\partial x} + k_{yy}\frac{\partial\varphi}{\partial y}\Big)(x_{ki} - x_c)\Big]. \tag{1.25c}$$

Note that the first equality in (1.25) states the fact that any flux flowing out of one control volume equals the flux flowing into another control volume through their common boundary within a triangular element. Thus, the CVM is a flux continuous numerical method.

1.6.4 Upstream weighting

The upstream weighted properties in the CVM are determined by the flux-based upstream weighting scheme (unlike the CVFE where the discretized equations are reformulated to resemble the finite difference formulation and a potential-based approach is used).

The concept of flux-based upstream weighting in the CVM is better explained by examples. Figure 1.5 shows an example triangle with constant flux across the

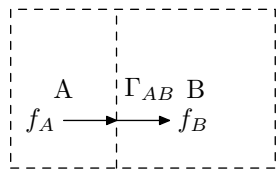

Figure 1.6: An illustration of the concept of flux continuity.

three control volume boundaries. The upstream properties can then be determined for each flux as follows.

1. For $f_{\overline{cij}}$, up $= j$;
2. for $f_{\overline{cjk}}$, up $= k$;
3. for $f_{\overline{cki}}$, up $= k$.

During the programming implementation, the upstream direction is determined by the sign of the flux. Recall the definition of f_i; when $f_i > 0$ the flux is flowing out of the control volume i. Therefore, the flux-based upstream operator up(ij) for the CVM is defined as

$$\mathrm{up}(ij) = \begin{cases} i & \text{if } f_{i,\overline{cij}} > 0, \\ j & \text{if } f_{j,\overline{cij}} > 0. \end{cases} \tag{1.26}$$

Remark 1.1 *For simplicity, the same symbol* up(ij) *is used for the upstream operator in* (1.13) *and* (1.26)*. There should be no confusion about which equation to use when* up(ij) *is encountered.*

1.7 Flux continuity

The term flux continuity for a control volume based method is defined as follows.

Definition 1.1 *A control volume based numerical method is flux continuous if and only if flux is defined on the control volume boundaries, and the flux flowing out of a control volume is exactly the same as the flux flowing into another control volume through their common boundary.*

This concept is shown in Figure 1.6 where f_A is the flux flowing out of control volume A and f_B is the flux flowing into control volume B through their common boundary Γ_{AB}. For the numerical method to be flux continuous we require

$$f_A + f_B = 0. \tag{1.27}$$

In the remainder of this section we show that the CVM is flux continuous and prove that the CVFE is not flux continuous.

1.7.1 CVM

Equation (1.25) provides expressions for pairs of fluxes entering and leaving control volumes through identified boundaries. Examinzation of this equation shows that the fluxes entering and leaving a boundary add to zero, thus satisfying the definition of flux continuity.

1.7.2 FEM and CVFE

The FEM and CVFE are considered together because they yield the same discretized equations. To prove that these two methods are not flux continuous, a general case is studied to show that the CVFE formulation satisfies Definition 1.1 if and only if the phase potentials at every node are equal. Consequentlly, when there is finite flux, the CVFE is not flux continuous.

Theorem 1.1 *Consider a triangular element $t_m \in \mathcal{T}$ with arbitrary shape and orientation. The CVFE fluxes between the control volumes $\{b_i, b_j, b_k\} \in \mathcal{B}$ in t_m are continuous if and only if the phase potential at three triangle vertices are equal, that is $\varphi_i = \varphi_j = \varphi_k$.*

Proof 1 *For simplicity, this proof is show for the case where*

$$\underline{k} = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix} \in \mathcal{P}_+^0(\mathcal{T}).$$

As discussed in §1.5, the CVFE formulation was rearranged for upstream implementation. To study the flux continuity property of this method, (1.11) *must be returned to its original flux-significant formulation. Starting with the flux portion of* (1.11)*, the flux into and out of b_i is separated into x- and y-components.*

$$\begin{aligned} f_i = A\Big[& k_x \frac{\partial L_i}{\partial x}\Big(\big(\rho\lambda\big)_{\mathrm{up}(ij)} \frac{\partial L_j}{\partial x}(\varphi_j - \varphi_i) + \big(\rho\lambda\big)_{\mathrm{up}(ik)} \frac{\partial L_k}{\partial x}(\varphi_k - \varphi_i)\Big) \\ & + k_y \frac{\partial L_i}{\partial y}\Big(\big(\rho\lambda\big)_{\mathrm{up}(ij)} \frac{\partial L_j}{\partial y}(\varphi_j - \varphi_i) + \big(\rho\lambda\big)_{\mathrm{up}(ik} \frac{\partial L_k}{\partial y}(\varphi_k - \varphi_i)\Big)\Big], \end{aligned}$$

For each component, the fluxes through different straight line boundaries are further separated. Referring to Figure 1.4, since

$$\frac{\partial L_i}{\partial x} = \frac{y_j - y_k}{2A} = \frac{y_{ij} - y_{ki}}{A} = \frac{y_{ij} - y_c}{A} + \frac{y_c - y_{ki}}{A}, \tag{1.28a}$$

$$\frac{\partial L_i}{\partial y} = \frac{x_k - x_j}{2A} = \frac{x_{ki} - x_{ij}}{A} = \frac{x_{ki} - x_c}{A} + \frac{x_c - x_{ij}}{A}, \tag{1.28b}$$

the total flux can be written as

$$f_i = f_{ix,\overline{cij}} + f_{ix,\overline{cki}} + f_{iy,\overline{cij}} + f_{iy,\overline{cki}}, \tag{1.29}$$

which is the summation of the x-direction flux across $\overline{cij}$, $\overline{cki}$ and y-direction flux across $\overline{cij}$, $\overline{cki}$, respectively. The flux terms are defined by

$$
\begin{aligned}
f_{ix,\overline{cij}} &= k_x(y_{ij} - y_c)\Big[\big(\rho\lambda\big)_{\mathrm{up}(ij)}\frac{\partial L_j}{\partial x}(\varphi_j - \varphi_i) + \big(\rho\lambda\big)_{\mathrm{up}(ik)}\frac{\partial L_k}{\partial x}(\varphi_k - \varphi_i)\Big]\\
f_{ix,\overline{cki}} &= k_x(y_c - y_{ki})\Big[\big(\rho\lambda\big)_{\mathrm{up}(ij)}\frac{\partial L_j}{\partial x}(\varphi_j - \varphi_i) + \big(\rho\lambda\big)_{\mathrm{up}(ik)}\frac{\partial L_k}{\partial x}(\varphi_k - \varphi_i)\Big]\\
f_{iy,\overline{cij}} &= k_y(x_c - x_{ij})\Big[\big(\rho\lambda\big)_{\mathrm{up}(ij)}\frac{\partial L_j}{\partial y}(\varphi_j - \varphi_i) + \big(\rho\lambda\big)_{\mathrm{up}(ik)}\frac{\partial L_k}{\partial y}(\varphi_k - \varphi_i)\Big]\\
f_{iy,\overline{cki}} &= k_y(x_{ki} - x_c)\Big[\big(\rho\lambda\big)_{\mathrm{up}(ij)}\frac{\partial L_j}{\partial y}(\varphi_j - \varphi_i) + \big(\rho\lambda\big)_{\mathrm{up}(ik)}\frac{\partial L_k}{\partial y}(\varphi_k - \varphi_i)\Big].
\end{aligned}
$$

Similar x- and y-direction fluxes can be derived for b_j and b_k and are listed here. For b_j,

$$f_j = f_{jx,\overline{cjk}} + f_{jx,\overline{cij}} + f_{jy,\overline{cjk}} + f_{jy,\overline{cij}}, \tag{1.30}$$

where

$$
\begin{aligned}
f_{jx,\overline{cjk}} &= k_x(y_{jk} - y_c)\Big[\big(\rho\lambda\big)_{\mathrm{up}(ij)}\frac{\partial L_i}{\partial x}(\varphi_i - \varphi_j) + \big(\rho\lambda\big)_{\mathrm{up}(jk)}\frac{\partial L_k}{\partial x}(\varphi_k - \varphi_j)\Big]\\
f_{jx,\overline{cij}} &= k_x(y_c - y_{ij})\Big[\big(\rho\lambda\big)_{\mathrm{up}(ij)}\frac{\partial L_i}{\partial x}(\varphi_i - \varphi_j) + \big(\rho\lambda\big)_{\mathrm{up}(jk)}\frac{\partial L_k}{\partial x}(\varphi_k - \varphi_j)\Big]\\
f_{jy,\overline{cjk}} &= k_y(x_c - x_{jk})\Big[\big(\rho\lambda\big)_{\mathrm{up}(ij)}\frac{\partial L_i}{\partial y}(\varphi_i - \varphi_j) + \big(\rho\lambda\big)_{\mathrm{up}(jk)}\frac{\partial L_k}{\partial y}(\varphi_k - \varphi_j)\Big]\\
f_{jy,\overline{cij}} &= k_y(x_{ij} - x_c)\Big[\big(\rho\lambda\big)_{\mathrm{up}(ij)}\frac{\partial L_i}{\partial y}(\varphi_i - \varphi_j) + \big(\rho\lambda\big)_{\mathrm{up}(jk)}\frac{\partial L_k}{\partial y}(\varphi_k - \varphi_j)\Big].
\end{aligned}
$$

For b_k,

$$f_k = f_{kx,\overline{cki}} + f_{kx,\overline{cjk}} + f_{ky,\overline{cki}} + f_{ky,\overline{cjk}}, \tag{1.31}$$

where

$$
\begin{aligned}
f_{kx,\overline{cki}} &= k_x(y_{ki} - y_c)\Big[\big(\rho\lambda\big)_{\mathrm{up}(ki)}\frac{\partial L_i}{\partial x}(\varphi_i - \varphi_k) + \big(\rho\lambda\big)_{\mathrm{up}(kj)}\frac{\partial L_k}{\partial x}(\varphi_j - \varphi_k)\Big]\\
f_{kx,\overline{cjk}} &= k_x(y_c - y_{jk})\Big[\big(\rho\lambda\big)_{\mathrm{up}(ki)}\frac{\partial L_i}{\partial x}(\varphi_i - \varphi_k) + \big(\rho\lambda\big)_{\mathrm{up}(kj)}\frac{\partial L_k}{\partial x}(\varphi_j - \varphi_k)\Big]\\
f_{ky,\overline{cki}} &= k_y(x_c - x_{ki})\Big[\big(\rho\lambda\big)_{\mathrm{up}(ki)}\frac{\partial L_i}{\partial y}(\varphi_i - \varphi_k) + \big(\rho\lambda\big)_{\mathrm{up}(kj)}\frac{\partial L_k}{\partial y}(\varphi_j - \varphi_k)\Big]\\
f_{ky,\overline{cjk}} &= k_y(x_{jk} - x_c)\Big[\big(\rho\lambda\big)_{\mathrm{up}(ki)}\frac{\partial L_i}{\partial y}(\varphi_i - \varphi_k) + \big(\rho\lambda\big)_{\mathrm{up}(kj)}\frac{\partial L_k}{\partial y}(\varphi_j - \varphi_k)\Big].
\end{aligned}
$$

For the CVFE formulation to be flux continuous, we require the x- and y-direction fluxes through each interface to satisfy Definition 1.1. Therefore, the CVFE is flux continuous if and only if the following equations are all true.

$$f_{iu,\overline{cij}} + f_{ju,\overline{cij}} = 0, \tag{1.32a}$$

$$f_{ju,\overline{cjk}} + f_{ku,\overline{cjk}} = 0, \tag{1.32b}$$

$$f_{ku,\overline{cki}} + f_{iu,\overline{cki}} = 0 \quad \textit{where} \quad u = \{x, y\}. \tag{1.32c}$$

Consider (1.32a).

$$
\begin{aligned}
0 = k_x(y_{ij} - y_c)\Big[&\big(\rho\lambda\big)_{\mathrm{up}(ij)}\big(\frac{\partial L_i}{\partial x} + \frac{\partial L_j}{\partial x}\big)(\varphi_j - \varphi_i) \\
&+ \big(\rho\lambda\big)_{\mathrm{up}(ik)}\frac{\partial L_k}{\partial x}(\varphi_k - \varphi_i) + \big(\rho\lambda\big)_{\mathrm{up}(jk)}\frac{\partial L_k}{\partial x}(\varphi_j - \varphi_k)\Big]
\end{aligned} \tag{1.33a}
$$

$$
\begin{aligned}
0 = k_y(x_c - x_{ij})\Big[&\big(\rho\lambda\big)_{\mathrm{up}(ij)}\big(\frac{\partial L_i}{\partial y} + \frac{\partial L_j}{\partial y}\big)(\varphi_j - \varphi_i) \\
&+ \big(\rho\lambda\big)_{\mathrm{up}(ik)}\frac{\partial L_k}{\partial y}(\varphi_k - \varphi_i) + \big(\rho\lambda\big)_{\mathrm{up}(jk)}\frac{\partial L_k}{\partial y}(\varphi_j - \varphi_k)\Big]
\end{aligned} \tag{1.33b}
$$

The only possibility for (1.33) *to be true for any regular shape and any orientation of triangular element is when*

$$\varphi_k \geq \varphi_i = \varphi_j. \tag{1.34}$$

The same argument can be applied to (1.32b) *and* (1.32c), *and we have*

$$\varphi_i \geq \varphi_j = \varphi_k, \tag{1.35}$$

$$\varphi_j \geq \varphi_i = \varphi_k, \tag{1.36}$$

respectively. From (1.34), (1.35), *and* (1.36), *we conclude that the CVFE is flux continuous only if* $\varphi_i = \varphi_j = \varphi_k$. *On the other hand, if* $\varphi_i = \varphi_j = \varphi_k$ *then* (1.32) *is true and the CVFE is flux continuous. Therefore, we have proved that the CVFE is flux continuous if and only if the phase potentials at three vertices are equal.*

Remark 1.2 *The only difference between the CVFE and the CVM is the choice of upstream properties. If* (1.32a) *is derived using the CVM, all the upstream properties would have been assigned to the values of the same control volume, and for* $u = x$

$$
\begin{aligned}
&f_{ix,\overline{cij}} + f_{jx,\overline{cij}} \\
&= k_x(y_{ij} - y_c)\big(\rho\lambda\big)_{\mathrm{up}}\Big[\big(\frac{\partial L_i}{\partial x} + \frac{\partial L_j}{\partial x}\big)(\varphi_j - \varphi_i) + \frac{\partial L_k}{\partial x}(\varphi_k - \varphi_i) + \frac{\partial L_k}{\partial x}(\varphi_j - \varphi_k)\Big] \\
&= 0,
\end{aligned}
$$

since $\partial L_i/\partial x + \partial L_j/\partial x + \partial L_k/\partial x = 0$.

1.8 Mass conservation

The governing equations for oil reservoir simulations are basically mass conservation equations. It is, therefore, very important to verify that the numerical methods employed in solving the equations are both globally and locally mass conservative.

It is well known that both the FEM and CVFE are globally mass conservative. The CVM derived in this paper is also globally mass conservative because the solution is obtained by forcing the set of residual functions to be as close to zero as possible.

It is widely believed that the FEM is not a locally mass conservative method. On the other hand, the CVFE is considered to be locally mass conservative. However, both the methods yield exactly the same discretized equations. In view of this, we first recall the definition of local mass conservation [22] and examine the CVFE, CVM and FEM methods.

Definition 1.2 *A control volume based numerical method is locally mass conservative if and only if flux is defined on the control volume interfaces, and the total fluxes flowing into and out of a control volume is exactly balanced by the accumulation term and the source terms.*

Consider the CVFE and CVM. Both methods solve the approximated solution by forcing every residual function to be zero. Taking a close look at their residual functions, (1.14), it is seen that Definition 1.2 is satisfied for every control volume. Consequentlly, both methods are locally mass conservative.

When the local mass conservation property of the FEM is considered on $\mathcal{T}$, the FEM is not conservative. This is because the flux is not defined on the edge of triangles. On the other hand, if the same control volumes, $\mathcal{B}$, as defined in the CVFE are considered for the FEM then the FEM is also locally mass conservative. This is a logical approach because the FEM implies the existance of $\mathcal{B}$ as discussed in §1.5.

1.9 Numerical experiments

First, a simple single element example is considered. The effect of different upstream weighting implementation is discussed. The implication of positive transmissibility condition [16] is also considered.

The second example considered is a five-spot injection problem. The results from the CVM and the CVFE are compared; the grid orientation effect is studied.

1.9.1 Single triangle example

The term *irreducible phase content* used in reservoir engineering is discussed first. Due to the nature of porous media, a fluid phase is only mobile when its saturation value is above its irreducible phase content S_{ir} in the porous medium. This property of the porous medium is reflected in the relative permeability function $k_r(S)$. The function $k_r(S)$ has the following properties

$$k_r(S) \begin{cases} = 0 & \text{if } S \leq S_{\text{ir}}, \\ > 0 & \text{if } S > S_{\text{ir}}. \end{cases}$$

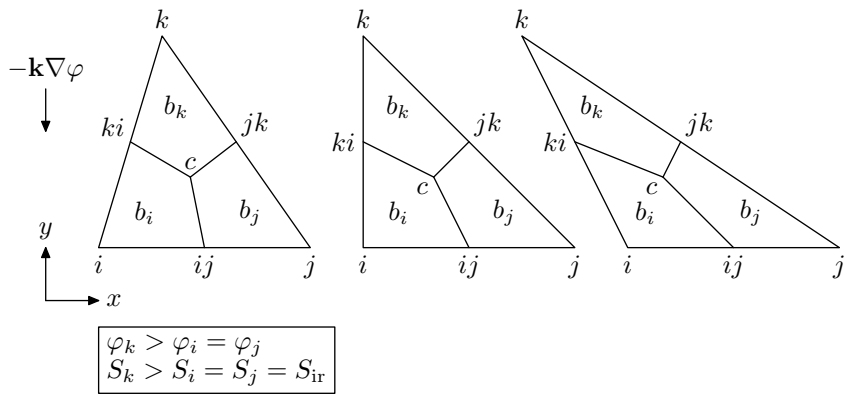

Figure 1.7: Acute, right-angled, and obtuse triangles used in the discussion of the consequences of applying either the potential- or flux-based upstream condition.

Therefore, a fluid phase is immobile if $S \leq S_{\mathrm{ir}}$ even when there exists a potential gradient.

The positive transmissibility condition for a single triangle is that all angles are equal to or less than $\pi/2$ [16]. Consider the acute triangle in Figure 1.7. Relative values of phase potentials and saturations are indicated in the figure. Assume that the permeability tensor is identity; then, the flux direction is pointed in the negative y-direction. The flux flowing out of b_j through $\overline{cij}$ in the y-direction should be zero because $S_j = S_{\mathrm{ir}}$. However, the actual flux calculated by the CVFE is

$$\text{CVFE:} \quad f_{jy,\overline{cij}} = k_y(x_{ij} - x_c)\big(\rho\lambda\big)_k \frac{x_j - x_i}{2A}(\varphi_k - \varphi_j) \neq 0.$$

It is clear that even if the positive transmissibility condition is satisfied, the CVFE still has unrealistic fluxes. In contrast, the same flux in the CVM is

$$\text{CVM:} \quad f_{jy,\overline{cij}} = \big(\rho\lambda\big)_j k_y \frac{\partial\varphi}{\partial y}(x_{ij} - x_c) = 0$$

because, $\lambda_j = 0$.

Consider the right-angled triangle in Figure 1.7. The potential and saturation conditions in Figure 1.7 require the flux through the boundary $\overline{cjk}$ should be nonzero. This means that the value of S_j should increase. The flux flowing into b_j through $\overline{cjk}$ in y-direction calculated by the CVFE is

$$\text{CVFE:} \quad f_{jy,\overline{cjk}} = k_y(x_c - x_{jk})\big(\rho\lambda\big)_k \frac{x_j - x_i}{2A}(\varphi_k - \varphi_j).$$

It is clear that $f_{jy,\overline{cij}} + f_{jy,\overline{cjk}} = 0$ because $(x_{ij} - x_c) = -(x_c - x_{jk})$. As a result, S_j remains unchanged. Thus, once again, it is possible to develop physically incorrect solutions using CVFE. It is for this reason that CVFE is considered a five-point

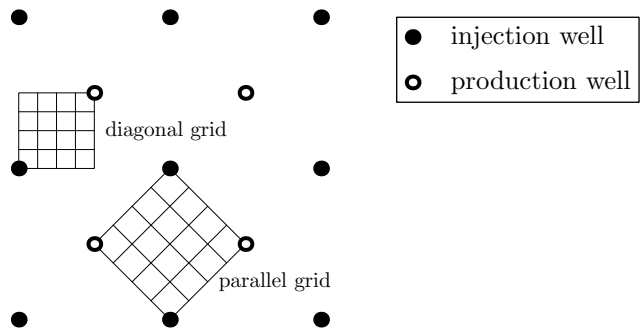

Figure 1.8: Five-spot injection prodcution pattern used as an example calculation problem.

stencil method [18] for the right-angled triangles, leading to grid orientation effects. As for the CVM, we have $f_{jy,\overline{cij}} + f_{jy,\overline{cjk}} < 0$ and S_j increases; therefore, a net flux exists in the diagonal direction.

Consider the obtuse triangle in Figure 1.7. For the CVFE formulation, as a result of violating the positive transmissibility condition, we have $f_{jy,\overline{cij}} + f_{jy,\overline{cjk}} > 0$, because $x_{ij} - x_{jk} > 0$. The phase saturation S_j is, therefore, reduced to a value less than S_{ir}, which is physically impossible. In the CVM, we have $f_{jy,\overline{cij}} = 0$ and $f_{jy,\overline{cjk}} < 0$; therefore, the value of S_j increases.

1.9.2 Five-spot injection problem

The test problem for the proposed CVM and the CVFE is a five-spot water injection problem. The arrangement of production and injection wells is shown in Figure 1.9.2. The distance between any two adjacent production wells is 2000 ft. One injection well is placed at the center of a square formed by four surrounding production wells. The wetting and non-wetting phases considered in this problem are water and oil, respectively. Oil and water are produced from production wells, and water is injected through injection wells. Two types of grids are used for testing the grid orientation effects. In the diagonal grid, the production and injection wells are connected through the diagonal of the grid. In the parallel grid, wells are connected through grid lines. These two subdomains within the larger five-spot pattern are shown in Figure 1.9.2. Each subdomain is discretized into 20 by 20 square blocks. Each square block is further divided into two triangles. It should be noted that the parallel grid domain is twice the size of the diagonal grid domain. The water migration patterns are expected to be identical between the injection and production wells in both cases. The oil production rates at different water injection stages are also expected to be the same for the two grid systems.

Reservoir rock and fluid properties are listed in Table 1.1. Initially, oil pressure is at 3000 psia and water saturation is 0.20. Because of the symmetric layout of the wells, no flow boundary conditions are used for the two subdomains in consideration. The total fluid rate for both injection and production wells is 40

Table 1.1: Rock and fluid properties used in the example problem.

Rock Properties	
k_x	200 md
k_y	200 md
ϕ	0.10=10%
P_c	0
$S_{o,\mathrm{ir}}$	0.2
$S_{w,\mathrm{ir}}$	0.2
k_{ro}	$(S_o - S_{o,\mathrm{ir}})^3$
k_{rw}	$(S_w - S_{w,\mathrm{ir}})^3$
Fluid Properties	
μ_o	10 cp
μ_w	1 cp
ρ_o	$(\frac{P-14.7}{89867.7} + 1)\rho_{o,\mathrm{STC}}$[1],[2]
ρ_w	$\rho_{w,\mathrm{STC}}$

[1]STC = stock tank condition
[2]P in psia

stock tank barrels per day (STB/day). Since the simulation domain is only a quarter of the five-spot pattern, the fluid rate is further divided by four and a rate of 10 STB/day is used for each well.

Consider the solutions of CVM and CVFE on the diagonal grid. Figure 1.9 shows the legend and Figure 1.10 shows the water saturation contours at different injection stages (different times) for these two methods on the diagonal grid. Close to the injection well, water tends to move evenly along the diagonal and grid lines for the CVM. On the contrary, water tends to move along the grid lines for the CVFE.

Figure 1.11 shows the water saturation contours at various pore volumes injected for the two solution methods on the parallel grid. For this grid, water migration patterns using the CVM and CVFE are different both around the injection and production wells.

Figure 1.12 shows the oil production rate at production wells for the two numerical methods on the two grids. It is evident that the water breakthrough times (the moment at which the fluid at the production well ceases to be pure oil) are different for the two methods. It is also clear that the CVFE solution for breakthrough times and oil rates for the two grid systems are quite different. The CVFE parallel grid solution provides the quickest breakthrough because of water movement predominantly along grid lines. In this case, the CVM solutions for the two grid systems are almost the same, thus, showing very little grid orientation effect.

1.10 Conclusions

The potential-based upstream weighting method, commonly used in control volume finite element (CVFE) numerical schemes for the solution of multiphase porous media problems results in solutions which are not flux continuous. A flux-based upstream weighting procedure which provides flux continuous solutions for control volume methods (CVM) is presented in this paper. The CVM is globally and locally mass conservative. The mass-lumping schemes employed in finite element (FEM) numerical methods for the solution of porous media flow problem imply the same set of control volumes as in CVFE. Both the CVFE and the FEM methods are locally and globally mass conservative. The CVM is not restricted by the positive transmissibility condition that limits the shape of the triangular elements used in CVFE. The upstream weighting scheme in CVFE can result in physically unrealistic fluxes even if the positive transmissibility condition is satisfied. The CVFE and the CVM yield different sets of solutions in a simple five-spot injection-production example. In the diagonal and the parallel grid orientations evaluated for this example, the breakthrough times and oil production rates are expected to be the same; these values calculated by the CVM are close while those from the CVFE are different.

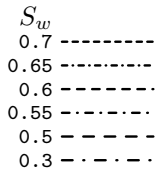

Figure 1.9: Legend for water contour plots.

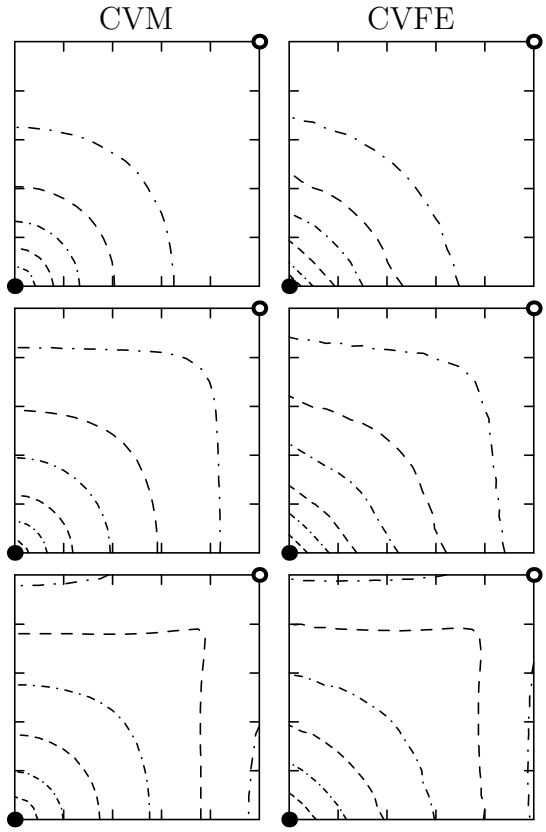

Figure 1.10: Comparison of the water saturation contours of the diagonal grid problem solved by the CVM (left) and the CVFE (right). From top to bottom are the contours at 0.1, 0.2 and 0.4 domain pore volume of water injected.

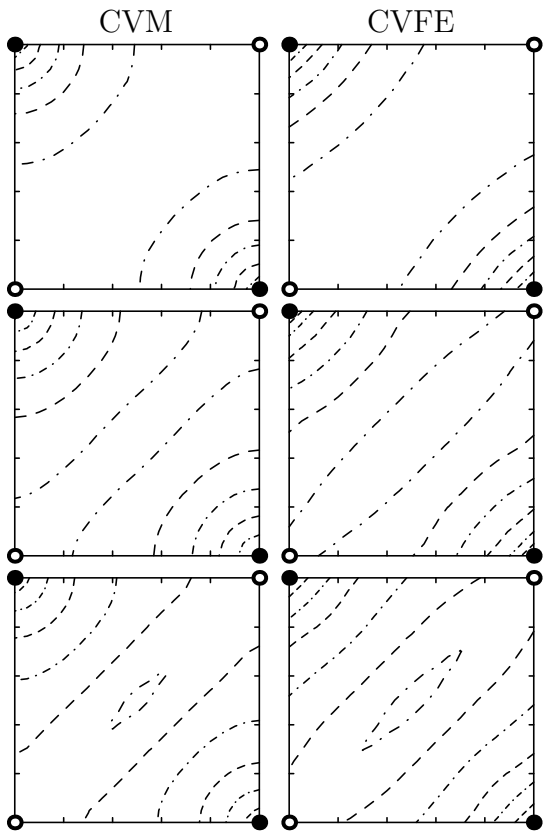

Figure 1.11: Comparison of the water saturation contours of the parallel grid problem solved by the CVM (left) and the CVFE (right). From top to bottom are the contours at 0.1, 0.2 and 0.4 domain pore volume of water injected.

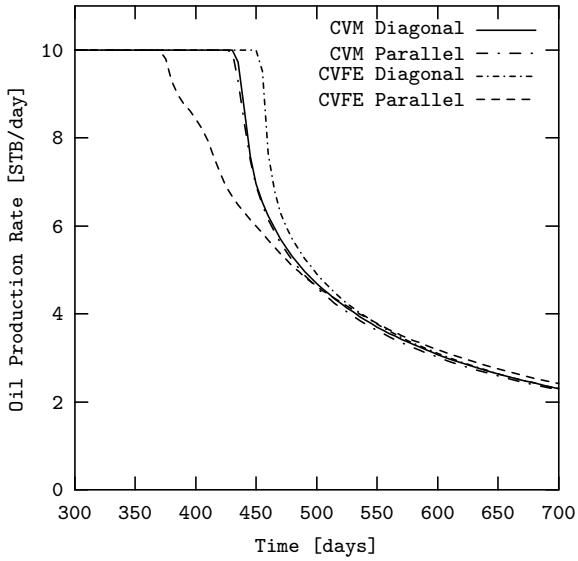

Figure 1.12: Comparison of the oil production rates for the CVM and the CVFE on the diagonal and the parallel grids.

Chapter 2

The Control Volume Finite Element Method

2.1 Synopsis

Reservoir simulation often requires representing complex, irregular domains and complicated fracture networks. The finite difference method is not capable of handling these complex features; finite element method is a good alternative for representing and modeling these systems.

The control volume finite element method (CVFE) is closely related to the finite element method (FEM). They both use the same types of interpolation functions for dependent variables. They differ in the way in which fluid flux between control volumes is calculated. In the FEM method, fluid potentials are approximated without the knowledge of fluxes between nodes; however, in the CVFE, fluid flux between nodes is calculated explicitly, and mass balance is formulated according to the flux.

The two-dimensional, two-phase CVFE was developed by Yi-kun Yang [23]. Mathematical formulation of CVFE for three-dimension is discussed in this chapter. Following is the outline of the development of CVFE. In section 2.2, the interpolation functions are derived. Based on the concept of control volume, the discretized residual equations for three-dimensional, two-phase system are formulated in section 2.3. The three-dimensional, three-phase formulations are discussed in section 2.4.

2.2 Area Coordinate System and Interpolation Functions

In this section, the lowest order of interpolation function from the Lagrange family–linear interpolation functions–for tetrahedral elements is discussed.

The position of any given point $\mathbf{x}$ in a tetrahedral element can be uniquely defined by the volumes enclosed with the vertices of the tetrahedron. As shown

in Figure 2.1, $\mathbf{x}$ is defined by

$$\mathbf{x} = (L_0, L_1, L_2) = (\frac{V_0}{V}, \frac{V_1}{V}, \frac{V_2}{V}). \tag{2.1}$$

Notice that

$$L_0 + L_1 + L_2 + L_3 = 1; \tag{2.2}$$

therefore L_3 can not be changed independently without disturbing the values of L_0, L_1 and L_2.

The linear interpolation functions for a given point $\mathbf{x}$ in a tetrahedral element are actually its coordinates. The value of h at $\mathbf{x} \in \Omega_e$ can be approximated using the values of h at the vertices and the interpolation functions

$$h(x) = \sum_{i=0}^{3} h_i L_i(\mathbf{x}). \tag{2.3}$$

Notice that h_i denotes the value of h at vertex i of the tetrahedron.

It can be seen that the value of ∇h needs to be calculated. In the case that $\Omega \subset \Re^3$, the values of $\frac{\partial h}{\partial x}$, $\frac{\partial h}{\partial y}$ and $\frac{\partial h}{\partial z}$ are sought. When h is written in the form of equation 2.3, it is clear that $\frac{\partial h}{\partial L_0}$, $\frac{\partial h}{\partial L_1}$ and $\frac{\partial h}{\partial L_2}$ can be evaluated much easier than $\frac{\partial h}{\partial x}$, $\frac{\partial h}{\partial y}$ and $\frac{\partial h}{\partial z}$. So $\frac{\partial h}{\partial L_0}$, $\frac{\partial h}{\partial L_1}$ and $\frac{\partial h}{\partial L_2}$ are calculated first, and then $\frac{\partial h}{\partial x}$, $\frac{\partial h}{\partial y}$ and $\frac{\partial h}{\partial z}$ are derived by matrix inversion.

Apply chain rule to $\frac{\partial h}{\partial x}$, $\frac{\partial h}{\partial y}$ and $\frac{\partial h}{\partial z}$:

$$\frac{\partial h}{\partial x} = \frac{\partial h}{\partial L_0}\frac{\partial L_0}{\partial x} + \frac{\partial h}{\partial L_1}\frac{\partial L_1}{\partial x} + \frac{\partial h}{\partial L_2}\frac{\partial L_2}{\partial x}, \tag{2.4a}$$

$$\frac{\partial h}{\partial y} = \frac{\partial h}{\partial L_0}\frac{\partial L_0}{\partial y} + \frac{\partial h}{\partial L_1}\frac{\partial L_1}{\partial y} + \frac{\partial h}{\partial L_2}\frac{\partial L_2}{\partial y}, \tag{2.4b}$$

$$\frac{\partial h}{\partial z} = \frac{\partial h}{\partial L_0}\frac{\partial L_0}{\partial z} + \frac{\partial h}{\partial L_1}\frac{\partial L_1}{\partial z} + \frac{\partial h}{\partial L_2}\frac{\partial L_2}{\partial z}, \tag{2.4c}$$

Rewrite equation 2.4 in its matrix form:

$$\begin{Bmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial h}{\partial y} \\ \frac{\partial h}{\partial z} \end{Bmatrix} = \begin{bmatrix} \frac{\partial L_0}{\partial x} & \frac{\partial L_1}{\partial x} & \frac{\partial L_2}{\partial x} \\ \frac{\partial L_0}{\partial y} & \frac{\partial L_1}{\partial y} & \frac{\partial L_2}{\partial y} \\ \frac{\partial L_0}{\partial z} & \frac{\partial L_1}{\partial z} & \frac{\partial L_2}{\partial z} \end{bmatrix} \begin{Bmatrix} \frac{\partial h}{\partial L_0} \\ \frac{\partial h}{\partial L_1} \\ \frac{\partial h}{\partial L_2} \end{Bmatrix} = J^* \begin{Bmatrix} \frac{\partial h}{\partial L_0} \\ \frac{\partial h}{\partial L_1} \\ \frac{\partial h}{\partial L_2} \end{Bmatrix}, \tag{2.5}$$

where

$$J^* = \begin{bmatrix} \frac{\partial L_0}{\partial x} & \frac{\partial L_1}{\partial x} & \frac{\partial L_2}{\partial x} \\ \frac{\partial L_0}{\partial y} & \frac{\partial L_1}{\partial y} & \frac{\partial L_2}{\partial y} \\ \frac{\partial L_0}{\partial z} & \frac{\partial L_1}{\partial z} & \frac{\partial L_2}{\partial z} \end{bmatrix}. \tag{2.6}$$

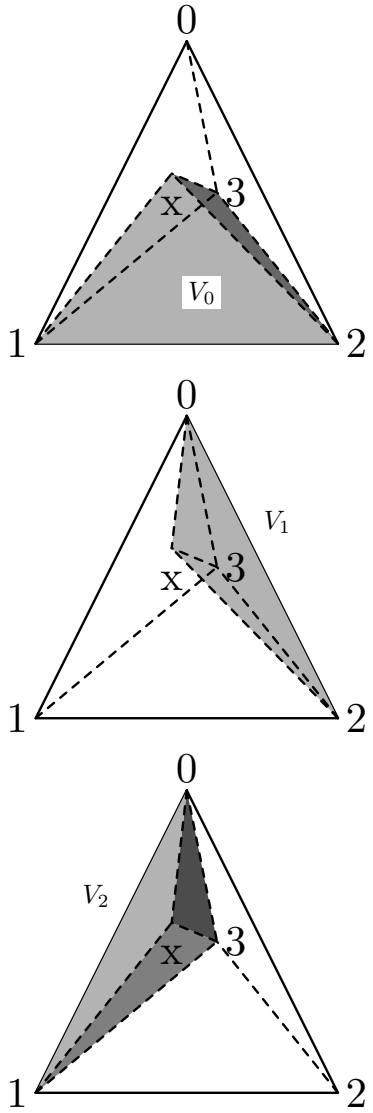

Figure 2.1: Definition of the natural coordinate of a tetrahedral element

Expand $\frac{\partial h}{\partial L_0}$, $\frac{\partial h}{\partial L_1}$ and $\frac{\partial h}{\partial L_2}$ in the same way as above:

$$\frac{\partial h}{\partial L_0} = \frac{\partial h}{\partial x}\frac{\partial x}{\partial L_0} + \frac{\partial h}{\partial y}\frac{\partial y}{\partial L_0} + \frac{\partial h}{\partial z}\frac{\partial z}{\partial L_0}, \tag{2.7a}$$

$$\frac{\partial h}{\partial L_1} = \frac{\partial h}{\partial x}\frac{\partial x}{\partial L_1} + \frac{\partial h}{\partial y}\frac{\partial y}{\partial L_1} + \frac{\partial h}{\partial z}\frac{\partial z}{\partial L_1}, \tag{2.7b}$$

$$\frac{\partial h}{\partial L_2} = \frac{\partial h}{\partial x}\frac{\partial x}{\partial L_2} + \frac{\partial h}{\partial y}\frac{\partial y}{\partial L_2} + \frac{\partial h}{\partial z}\frac{\partial z}{\partial L_2}, \tag{2.7c}$$

Again, rewrite equation 2.7 in its matrix form:

$$\begin{Bmatrix} \frac{\partial h}{\partial L_0} \\ \frac{\partial h}{\partial L_1} \\ \frac{\partial h}{\partial L_1} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial L_0} & \frac{\partial y}{\partial L_0} & \frac{\partial z}{\partial L_0} \\ \frac{\partial x}{\partial L_1} & \frac{\partial y}{\partial L_1} & \frac{\partial z}{\partial L_1} \\ \frac{\partial x}{\partial L_2} & \frac{\partial y}{\partial L_2} & \frac{\partial z}{\partial L_2} \end{bmatrix} \begin{Bmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial h}{\partial y} \\ \frac{\partial h}{\partial z} \end{Bmatrix} = J \begin{Bmatrix} \frac{\partial h}{\partial x} \\ \frac{\partial h}{\partial y} \\ \frac{\partial h}{\partial z} \end{Bmatrix}, \tag{2.8}$$

where

$$J = \begin{bmatrix} \frac{\partial x}{\partial L_0} & \frac{\partial y}{\partial L_0} & \frac{\partial z}{\partial L_0} \\ \frac{\partial x}{\partial L_1} & \frac{\partial y}{\partial L_1} & \frac{\partial z}{\partial L_1} \\ \frac{\partial x}{\partial L_2} & \frac{\partial y}{\partial L_2} & \frac{\partial z}{\partial L_2} \end{bmatrix}. \tag{2.9}$$

J^* is actually the inverse of J, that is

$$J^* = J^{-1}. \tag{2.10}$$

According to equation 2.10, J^* can be easily calculated once J is known. The calculation of J follows.

The coordinate of any point **x** (x,y,z) within a tetrahedral element can always be written in terms of its coordinates using equation 2.3:

$$x = \sum_{i=0}^{3} x_i L_i(x) = L_0(x)(x_0 - x_3) + L_1(x)(x_1 - x_3) + L_2(x)(x_2 - x_3) + x_3 \tag{2.11a}$$

$$y = \sum_{i=0}^{3} y_i L_i(y) = L_0(y)(y_0 - y_3) + L_1(y)(y_1 - y_3) + L_2(y)(y_2 - y_3) + y_3 \tag{2.11b}$$

$$z = \sum_{i=0}^{3} z_i L_i(z) = L_0(z)(z_0 - z_3) + L_1(z)(z_1 - y_3) + L_2(z)(z_2 - z_3) + z_3 \tag{2.11c}$$

Differentiate x, y and z with respect to L_0, L_1 and L_2

$$\frac{\partial x}{\partial L_0} = x_0 - x_3, \tag{2.12a}$$

$$\frac{\partial x}{\partial L_1} = x_1 - x_3, \tag{2.12b}$$

$$\frac{\partial x}{\partial L_2} = x_2 - x_3, \tag{2.12c}$$

$$\frac{\partial y}{\partial L_0} = y_0 - y_3, \tag{2.12d}$$

$$\frac{\partial y}{\partial L_1} = y_1 - y_3, \tag{2.12e}$$

$$\frac{\partial y}{\partial L_2} = y_2 - y_3, \tag{2.12f}$$

$$\frac{\partial z}{\partial L_0} = z_0 - z_3, \tag{2.12g}$$

$$\frac{\partial z}{\partial L_1} = z_1 - z_3, \tag{2.12h}$$

$$\frac{\partial z}{\partial L_2} = z_2 - z_3, \tag{2.12i}$$

Substitute equations 2.12 into equation 2.9 to obtain

$$J = \begin{bmatrix} \frac{\partial x}{\partial L_0} & \frac{\partial y}{\partial L_0} & \frac{\partial z}{\partial L_0} \\ \frac{\partial x}{\partial L_1} & \frac{\partial y}{\partial L_1} & \frac{\partial z}{\partial L_1} \\ \frac{\partial x}{\partial L_2} & \frac{\partial y}{\partial L_2} & \frac{\partial z}{\partial L_2} \end{bmatrix} = \begin{bmatrix} x_0 - x_3 & y_0 - y_3 & z_0 - z_3 \\ x_1 - x_3 & y_1 - y_3 & z_1 - z_3 \\ x_2 - x_3 & y_2 - y_3 & z_2 - z_3 \end{bmatrix}. \tag{2.13}$$

J^* can be written as

$$J^* = J^{-1} = \begin{bmatrix} \frac{\partial L_0}{\partial x} & \frac{\partial L_1}{\partial x} & \frac{\partial L_2}{\partial x} \\ \frac{\partial L_0}{\partial y} & \frac{\partial L_1}{\partial y} & \frac{\partial L_2}{\partial y} \\ \frac{\partial L_0}{\partial z} & \frac{\partial L_1}{\partial z} & \frac{\partial L_2}{\partial z} \end{bmatrix} \tag{2.14}$$

where

$$\frac{\partial L_0}{\partial x} = \frac{1}{|J|}[(y_1 - y_3)(z_2 - z_3) - (y_2 - y_3)(z_1 - z_3)], \tag{2.15a}$$

$$\frac{\partial L_1}{\partial x} = \frac{1}{|J|}[(y_2 - y_3)(z_0 - z_3) - (y_0 - y_3)(z_2 - z_3)], \tag{2.15b}$$

$$\frac{\partial L_2}{\partial x} = \frac{1}{|J|}[(y_0 - y_3)(z_1 - z_3) - (y_1 - y_3)(z_0 - z_3)], \tag{2.15c}$$

$$\frac{\partial L_0}{\partial y} = \frac{1}{|J|}[(x_2 - x_3)(z_1 - z_3) - (x_1 - x_3)(z_2 - z_3)], \tag{2.15d}$$

$$\frac{\partial L_1}{\partial y} = \frac{1}{|J|}[(x_0 - x_3)(z_2 - z_3) - (x_2 - x_3)(z_0 - z_3)], \tag{2.15e}$$

$$\frac{\partial L_2}{\partial y} = \frac{1}{|J|}[(x_1 - x_3)(z_0 - z_3) - (x_0 - x_3)(z_1 - z_3)], \tag{2.15f}$$

$$\frac{\partial L_0}{\partial z} = \frac{1}{|J|}[(x_1 - x_3)(y_2 - y_3) - (x_2 - x_3)(y_1 - y_3)], \tag{2.15g}$$

$$\frac{\partial L_1}{\partial z} = \frac{1}{|J|}[(x_2 - x_3)(y_0 - y_3) - (x_0 - x_3)(y_2 - y_3)], \tag{2.15h}$$

$$\frac{\partial L_2}{\partial z} = \frac{1}{|J|}[(x_0 - x_3)(y_1 - y_3) - (x_1 - x_3)(y_0 - y_3)]. \tag{2.15i}$$

$\frac{\partial L_3}{\partial x}$, $\frac{\partial L_3}{\partial y}$ and $\frac{\partial L_3}{\partial z}$ can also be obtained using equations 2.2 and 2.15.

$$\frac{\partial L_3}{\partial x} = -\frac{\partial L_0}{\partial x} - \frac{\partial L_1}{\partial x} - \frac{\partial L_2}{\partial x}, \tag{2.16a}$$

$$\frac{\partial L_3}{\partial y} = -\frac{\partial L_0}{\partial y} - \frac{\partial L_1}{\partial y} - \frac{\partial L_2}{\partial y}, \tag{2.16b}$$

$$\frac{\partial L_3}{\partial z} = -\frac{\partial L_0}{\partial z} - \frac{\partial L_1}{\partial z} - \frac{\partial L_2}{\partial z}, \tag{2.16c}$$

With all the necessary components derived so far, $\frac{\partial h}{\partial x}$, $\frac{\partial h}{\partial y}$ and $\frac{\partial h}{\partial z}$ can be calculated as follows. Substitute equation 2.3 into equations for $\frac{\partial h}{\partial x}$, $\frac{\partial h}{\partial y}$ and $\frac{\partial h}{\partial z}$

$$\frac{\partial h}{\partial x} = \frac{\partial L_0}{\partial x}h_0 + \frac{\partial L_1}{\partial x}h_1 + \frac{\partial L_2}{\partial x}h_2 + \frac{\partial L_3}{\partial x}h_3, \tag{2.17a}$$

$$\frac{\partial h}{\partial y} = \frac{\partial L_0}{\partial y}h_0 + \frac{\partial L_1}{\partial y}h_1 + \frac{\partial L_2}{\partial y}h_2 + \frac{\partial L_3}{\partial y}h_3, \tag{2.17b}$$

$$\frac{\partial h}{\partial z} = \frac{\partial L_0}{\partial z}h_0 + \frac{\partial L_1}{\partial z}h_1 + \frac{\partial L_2}{\partial z}h_2 + \frac{\partial L_3}{\partial z}h_3. \tag{2.17c}$$

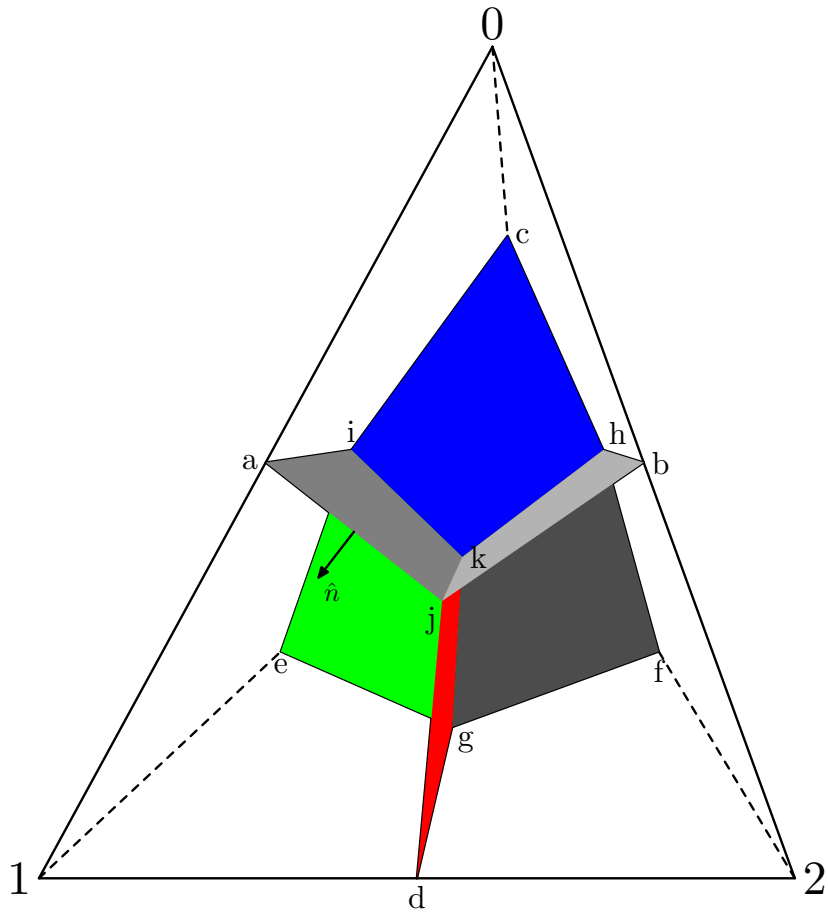

Figure 2.2: A tetrahedron element with associated control volumes

2.3 Three-Dimensional, Two-Phase Control Volume Formulation

The governing equation for two-phase flow may be generalized as follows:

$$-\nabla \cdot u_l = \frac{\partial}{\partial t}\left(\frac{1}{B_l}\phi_l S_l\right) + q_l \tag{2.18}$$

where l = o,w (oil and water phases, respectively). In subsequent development, the subscript will be omitted for abbreviation.

For the control volume formulation, both the fluid potential and saturation values are defined on tetrahedron vertices. The fluid potential value in a tetrahedron is interpolated as described in section 1. As for fluid saturation, it is defined as constant within each control volume.

Referring to Figure 2.2, only the residual function F_0 for the control volume surrounding node 0 is derived and similar procedures can be applied to obtain F_1, F_2 and F_3.

Start with equation 2.18 and integrate over V_0 in Figure 2.2:

$$F_0 = \int_{V0} \nabla \cdot \mathbf{V} + \frac{\partial C_0}{\partial t} d\mathbf{x} = 0. \tag{2.19}$$

Note that q is dropped in equation 2.19 since it is more convenient to deal with point source at global assembly stage. Applying divergence theorem to the flux term in equation 2.19 gives

$$\begin{aligned} F_0 &= \int_{aikj+bjkh+chki} \mathbf{V} \cdot \hat{n} ds + \int_{V_0} \frac{\partial C_0}{\partial t} d\mathbf{x} \\ &= \int_{aikj} \mathbf{V} \cdot \hat{n} ds + \int_{bjkh} \mathbf{V} \cdot \hat{n} ds + \int_{chki} \mathbf{V} \cdot \hat{n} ds + \int_{V_0} \frac{\partial C_0}{\partial t} d\mathbf{x} \end{aligned} \tag{2.20}$$

where $\hat{n}$ is the outward normal of the corresponding boundary as shown in Figure 2.2. Let

$$f_0 = f_{0,aikj} + f_{0,bjkh} + f_{0,chki}, \tag{2.21}$$

where

$$f_{0,aikj} = \int_{aikj} \mathbf{V} \cdot \hat{n} ds, \tag{2.22a}$$

$$f_{0,bjkh} = \int_{bjkh} \mathbf{V} \cdot \hat{n} ds, \tag{2.22b}$$

$$f_{0,chki} = \int_{chki} \mathbf{V} \cdot \hat{n} ds, \tag{2.22c}$$

then equation 2.20 becomes

$$F_0 = f_0 + \int_{V_0} \frac{\partial C_0}{\partial t} d\mathbf{x}. \tag{2.23a}$$

For F_1, F_2, and F_3,

$$F_1 = f_1 + \int_{V_1} \frac{\partial C_1}{\partial t} d\mathbf{x}, \tag{2.23b}$$

$$F_2 = f_2 + \int_{V_2} \frac{\partial C_2}{\partial t} d\mathbf{x}, \tag{2.23c}$$

$$F_3 = f_3 + \int_{V_3} \frac{\partial C_3}{\partial t} d\mathbf{x}. \tag{2.23d}$$

where

$$f_1 = f_{1,ajki} + f_{1,dgkj} + f_{1,eikj}, \tag{2.24}$$

$$f_2 = f_{2,bhkj} + f_{2,djkg} + f_{2,fgkh}, \tag{2.25}$$

$$f_3 = f_{3,chki} + f_{3,ejki} + f_{3,fhkg}. \tag{2.26}$$

Notice f_0 in equation 2.23 represents the flux portion of the local residual function.

The flux term is considered below and the accumulation term will be discussed later.

From Darcy's law,

$$\mathbf{V} = -\frac{k_r \rho}{B\mu} g k \nabla h. \tag{2.27}$$

where

$$h = \frac{P}{\rho g} + z. \tag{2.28}$$

In three-dimensional space,

$$\begin{aligned} f_{0,aikj} &= \int_{aikj} \mathbf{V} \cdot \hat{n} ds \\ &= \int_{aikj} (V_x \hat{i} + V_y \hat{j} + V_z \hat{k}) \cdot (n_x \hat{i} + n_y \hat{j} + n_z \hat{k}) ds \\ &= \int_{aikj} (V_x n_x + V_y n_y + V_z n_z) ds, \end{aligned} \tag{2.29}$$

where

$$V_x = -\left(T_{xx} \frac{\partial h}{\partial x} + T_{xy} \frac{\partial h}{\partial y} + T_{xz} \frac{\partial h}{\partial z} \right), \tag{2.30a}$$

$$V_y = -\left(T_{yx} \frac{\partial h}{\partial x} + T_{yy} \frac{\partial h}{\partial y} + T_{yz} \frac{\partial h}{\partial z} \right), \tag{2.30b}$$

$$V_z = -\left(T_{zx} \frac{\partial h}{\partial x} + T_{zy} \frac{\partial h}{\partial y} + T_{zz} \frac{\partial h}{\partial z} \right). \tag{2.30c}$$

To obtain the outward normal $\hat{n}$ of plane aikj, right-handedness is applied

$$\hat{n} = \frac{\vec{ai} \times \vec{aj}}{|\vec{ai} \times \vec{aj}|} = \frac{\vec{a3} \times \vec{a2}}{|\vec{a3} \times \vec{a2}|} \tag{2.31}$$

where the coordinate of $a = (\frac{x_0 + x_1}{2}, \frac{y_0 + y_1}{2}, \frac{z_0 + z_1}{2})$.

Let $\vec{a3} = (x_3 - x_a, y_3 - y_a, z_3 - z_a)$ and $\vec{a2} = (x_2 - x_a, y_2 - y_a, z_2 - z_a)$, for $\hat{n}$ of plane aikj

$$\begin{aligned} \hat{n} = &\frac{(y_3 - y_a)(z_2 - z_a) - (y_2 - y_a)(z_3 - z_a)}{|\vec{a3} \times \vec{a2}|} \vec{i} \\ &+ \frac{(x_2 - x_a)(z_3 - z_a) - (x_3 - x_a)(z_2 - z_a)}{|\vec{a3} \times \vec{a2}|} \vec{j} \\ &+ \frac{(x_3 - x_a)(y_2 - y_a) - (x_2 - x_a)(y_3 - y_a)}{|\vec{a3} \times \vec{a2}|} \vec{k}. \end{aligned} \tag{2.32}$$

The flux across aikj in Figure 2.2 can be expanded by substituting equations 2.30 and 2.32 into 2.29

$$
\begin{aligned}
&f_{0,aikj} \\
&= \int_{aikj} -\left(T_{xx}\frac{\partial h}{\partial x} + T_{xy}\frac{\partial h}{\partial y} + T_{xz}\frac{\partial h}{\partial z}\right)\frac{(y_3-y_a)(z_2-z_a)-(y_2-y_a)(z_3-z_a)}{|\vec{a3}\times\vec{a2}|} \\
&\quad -\left(T_{yx}\frac{\partial h}{\partial x} + T_{yy}\frac{\partial h}{\partial y} + T_{yz}\frac{\partial h}{\partial z}\right)\frac{(x_2-x_a)(z_3-z_a)-(x_3-x_a)(z_2-z_a)}{|\vec{a3}\times\vec{a2}|} \\
&\quad -\left(T_{zx}\frac{\partial h}{\partial x} + T_{zy}\frac{\partial h}{\partial y} + T_{zz}\frac{\partial h}{\partial z}\right)\frac{(x_3-x_a)(y_2-y_a)-(x_2-x_a)(y_3-y_a)}{|\vec{a3}\times\vec{a2}|}ds.
\end{aligned}
\tag{2.33}
$$

Let the formation volume factor B and viscosity μ in T be the average value of node 0 and 1

$$B = B_{01} = \frac{B(P_0)+B(P_1)}{2} \tag{2.34}$$

$$\mu = \mu_{01} = \frac{\mu(P_0)+\mu(P_1)}{2} \tag{2.35}$$

Here, P_i is the pressure at node i, and k_r is replaced with k_{r01}. This means k_r is determined by the direction of the flux between node 0 and 1. The area of the contacting domain between control volume 0 and 1 is

$$\int_{aikj} ds = S_{aikj} = \frac{1}{6}S_{\triangle 23a} = \frac{1}{6}\frac{1}{2}|\vec{a3}\times\vec{a2}| = \frac{1}{12}|\vec{a3}\times\vec{a2}|. \tag{2.36}$$

After taking the integrand out and canceling $|\vec{a3}\times\vec{a2}|$ with $\int_{aikj} ds$, the flux becomes

$$
\begin{aligned}
&f_{0,aikj} \\
&= \frac{1}{12}\frac{k_{r01}\rho_{01}}{B_{01}\mu_{01}}g \\
&\quad \Bigg\{ \left(k_{xx}\frac{\partial h}{\partial x} + k_{xy}\frac{\partial h}{\partial y} + k_{xz}\frac{\partial h}{\partial z}\right)[(y_2-y_a)(z_3-z_a)-(y_3-y_a)(z_2-z_a)] \\
&\quad + \left(k_{yx}\frac{\partial h}{\partial x} + k_{yy}\frac{\partial h}{\partial y} + k_{yz}\frac{\partial h}{\partial z}\right)[(x_3-x_a)(z_2-z_a)-(x_2-x_a)(z_3-z_a)] \\
&\quad + \left(k_{zx}\frac{\partial h}{\partial x} + k_{zy}\frac{\partial h}{\partial y} + k_{zz}\frac{\partial h}{\partial z}\right)[(x_2-x_a)(y_3-y_a)-(x_3-x_a)(y_2-y_a)]\Bigg\}.
\end{aligned}
\tag{2.37}
$$

The k_{r01} in equation 2.37 is determined by using upstream weighting. To apply this concept, the sign of $f_{0,aikj}$ is used to determine the upstream node. Note that the value of $\frac{1}{12}\frac{k_{r01}\rho_{01}}{B_{01}\mu_{01}}g$ is positive, and only the values in the bracket in equation 2.37 can be negative. Therefore the sign of $f_{0,aikj}$ is only determined by the terms

in the bracket. Positive flux means flux flowing out of the control volume. Apply the upstream condition to $f_{0,aikj}$.

- for $f_{0,aikj} > 0$, $k_{r01} = k_{r0} = k_r(S_0)$
- for $f_{0,aikj} < 0$, $k_{r01} = k_{r1} = k_r(S_1)$

Repeating the same procedure, the fluxes across bjkh, chki, dgkj, eikj and fgkh can be derived. The six fluxes across the six boundaries are summarized here

$$
\begin{aligned}
&f_{0,aikj} \\
&= -f_{1,ajki} \\
&= \frac{1}{12}\frac{k_{r01}\rho_{01}}{B_{01}\mu_{01}} g \\
&\quad \left\{ \left(k_{xx}\frac{\partial h}{\partial x} + k_{xy}\frac{\partial h}{\partial y} + k_{xz}\frac{\partial h}{\partial z} \right) [(y_2 - y_a)(z_3 - z_a) - (y_3 - y_a)(z_2 - z_a)] \right. \\
&\quad + \left(k_{yx}\frac{\partial h}{\partial x} + k_{yy}\frac{\partial h}{\partial y} + k_{yz}\frac{\partial h}{\partial z} \right) [(x_3 - x_a)(z_2 - z_a) - (x_2 - x_a)(z_3 - z_a)] \\
&\quad \left. + \left(k_{zx}\frac{\partial h}{\partial x} + k_{zy}\frac{\partial h}{\partial y} + k_{zz}\frac{\partial h}{\partial z} \right) [(x_2 - x_a)(y_3 - y_a) - (x_3 - x_a)(y_2 - y_a)] \right\}.
\end{aligned}
\tag{2.38a}
$$

$$
\begin{aligned}
&f_{0,bjkh} \\
&= -f_{2,bhkj} \\
&= \frac{1}{12}\frac{k_{r02}\rho_{02}}{B_{02}\mu_{02}} g \\
&\quad \left\{ \left(k_{xx}\frac{\partial h}{\partial x} + k_{xy}\frac{\partial h}{\partial y} + k_{xz}\frac{\partial h}{\partial z} \right) [(y_3 - y_b)(z_1 - z_b) - (y_1 - y_b)(z_3 - z_b)] \right. \\
&\quad + \left(k_{yx}\frac{\partial h}{\partial x} + k_{yy}\frac{\partial h}{\partial y} + k_{yz}\frac{\partial h}{\partial z} \right) [(x_1 - x_b)(z_3 - z_b) - (x_3 - x_b)(z_1 - z_b)] \\
&\quad \left. + \left(k_{zx}\frac{\partial h}{\partial x} + k_{zy}\frac{\partial h}{\partial y} + k_{zz}\frac{\partial h}{\partial z} \right) [(x_3 - x_b)(y_1 - y_b) - (x_1 - x_b)(y_3 - y_b)] \right\}.
\end{aligned}
\tag{2.38b}
$$

$$
\begin{aligned}
&f_{3,cikh} \\
&= -f_{0,chki} \\
&= \frac{1}{12}\frac{k_{r30}\rho_{30}}{B_{30}\mu_{30}} g \\
&\quad \left\{ \left(k_{xx}\frac{\partial h}{\partial x} + k_{xy}\frac{\partial h}{\partial y} + k_{xz}\frac{\partial h}{\partial z} \right) [(y_2 - y_c)(z_1 - z_c) - (y_1 - y_c)(z_2 - z_c)] \right. \\
&\quad + \left(k_{yx}\frac{\partial h}{\partial x} + k_{yy}\frac{\partial h}{\partial y} + k_{yz}\frac{\partial h}{\partial z} \right) [(x_1 - x_c)(z_2 - z_c) - (x_2 - x_c)(z_1 - z_c)] \\
&\quad \left. + \left(k_{zx}\frac{\partial h}{\partial x} + k_{zy}\frac{\partial h}{\partial y} + k_{zz}\frac{\partial h}{\partial z} \right) [(x_2 - x_c)(y_1 - y_c) - (x_1 - x_c)(y_2 - y_c)] \right\}.
\end{aligned}
\tag{2.38c}
$$

$$
\begin{aligned}
&f_{1,dgkj} \\
&= -f_{2,djkg} \\
&= \frac{1}{12}\frac{k_{r12}\rho_{12}}{B_{12}\mu_{12}}g \\
&\quad \Bigg\{ \left(k_{xx}\frac{\partial h}{\partial x} + k_{xy}\frac{\partial h}{\partial y} + k_{xz}\frac{\partial h}{\partial z}\right)[(y_0 - y_d)(z_3 - z_d) - (y_3 - y_d)(z_0 - z_d)] \\
&\quad + \left(k_{yx}\frac{\partial h}{\partial x} + k_{yy}\frac{\partial h}{\partial y} + k_{yz}\frac{\partial h}{\partial z}\right)[(x_3 - x_d)(z_0 - z_d) - (x_0 - x_d)(z_3 - z_d)] \\
&\quad + \left(k_{zx}\frac{\partial h}{\partial x} + k_{zy}\frac{\partial h}{\partial y} + k_{zz}\frac{\partial h}{\partial z}\right)[(x_0 - x_d a)(y_3 - y_d) - (x_3 - x_d)(y_0 - y_d)]\Bigg\}.
\end{aligned}
\tag{2.38d}
$$

$$
\begin{aligned}
&f_{1,egki} \\
&= -f_{3,eikg} \\
&= \frac{1}{12}\frac{k_{r13}\rho_{13}}{B_{13}\mu_{13}}g \\
&\quad \Bigg\{ \left(k_{xx}\frac{\partial h}{\partial x} + k_{xy}\frac{\partial h}{\partial y} + k_{xz}\frac{\partial h}{\partial z}\right)[(y_2 - y_e)(z_0 - z_e) - (y_0 - y_e)(z_2 - z_e)] \\
&\quad + \left(k_{yx}\frac{\partial h}{\partial x} + k_{yy}\frac{\partial h}{\partial y} + k_{yz}\frac{\partial h}{\partial z}\right)[(x_0 - x_e)(z_2 - z_e) - (x_2 - x_e)(z_0 - z_e)] \\
&\quad + \left(k_{zx}\frac{\partial h}{\partial x} + k_{zy}\frac{\partial h}{\partial y} + k_{zz}\frac{\partial h}{\partial z}\right)[(x_2 - x_e)(y_0 - y_e) - (x_0 - x_e)(y_2 - y_e)]\Bigg\}.
\end{aligned}
\tag{2.38e}
$$

$$
\begin{aligned}
&f_{2,fgkh} \\
&= -f_{3,fhkg} \\
&= \frac{1}{12}\frac{k_{r23}\rho_{23}}{B_{23}\mu_{23}}g \\
&\quad \Bigg\{ \left(k_{xx}\frac{\partial h}{\partial x} + k_{xy}\frac{\partial h}{\partial y} + k_{xz}\frac{\partial h}{\partial z}\right)[(y_0 - y_f)(z_1 - z_f) - (y_1 - y_f)(z_0 - z_f)] \\
&\quad + \left(k_{yx}\frac{\partial h}{\partial x} + k_{yy}\frac{\partial h}{\partial y} + k_{yz}\frac{\partial h}{\partial z}\right)[(x_1 - x_f)(z_0 - z_f) - (x_0 - x_f)(z_1 - z_f)] \\
&\quad + \left(k_{zx}\frac{\partial h}{\partial x} + k_{zy}\frac{\partial h}{\partial y} + k_{zz}\frac{\partial h}{\partial z}\right)[(x_0 - x_f)(y_1 - y_f) - (x_1 - x_f)(y_0 - y_f)]\Bigg\}.
\end{aligned}
\tag{2.38f}
$$

Note that the first equalities in equations 2.38 are indicative of the fact that any flux flowing out of one control volume must equal the flux into another control volume through their common control volume boundary. Thus the CVFE is a locally mass(flux) conservative method.

The accumulation term is discussed now. The definition of C is defined as $C_i = \phi\frac{S_i}{B_i}$. The porosity ϕ and saturation S are constant within the control volume

and B is calculated using the nodal pressure $B_i = B(P_i)$. As a result, C_i is constant with respect to V_i. Therefore, the accumulation terms in equations 2.23 can be written as

$$\begin{aligned}\int_{V_i} \frac{\partial C_i}{\partial t} dx &= \frac{\partial C_i}{\partial t} \int_{V_i} dx \\ &= \frac{\partial C_i}{\partial t} V_i = \frac{\partial C_i}{\partial t} \frac{V}{4}.\end{aligned} \tag{2.39}$$

where V_i is the volume for control volume i and V is the total volume of the tetrahedron.

To derive the final residual functions for all four control volumes, equations 2.38 and 2.39 are substituted into equations 2.6. After applying implicit Euler time discretization, equation 2.23 becomes

$$F_0^{n+1} = f_{0,aikj}^{n+1} + f_{0,bjkh}^{n+1} + f_{0,chki}^{n+1} + \frac{V}{3\triangle t}(C_0^{n+1} - C_0^n), \tag{2.40a}$$

$$F_1^{n+1} = f_{1,ajki}^{n+1} + f_{1,dgkj}^{n+1} + f_{1,eikj}^{n+1} + \frac{V}{3\triangle t}(C_1^{n+1} - C_1^n), \tag{2.40b}$$

$$F_2^{n+1} = f_{2,bhkj}^{n+1} + f_{2,djkg}^{n+1} + f_{2,fgkh}^{n+1} + \frac{V}{3\triangle t}(C_2^{n+1} - C_2^n), \tag{2.40c}$$

$$F_3^{n+1} = f_{3,chki}^{n+1} + f_{3,ejki}^{n+1} + f_{3,fhkg}^{n+1} + \frac{V}{3\triangle t}(C_3^{n+1} - C_3^n). \tag{2.40d}$$

where the superscripts n+1 and n represent the time levels.

2.4 Three-Dimensional, Three-Phase Simulations

In three-phase simulation, oil, water and gas are present. The solubility of gas in oil as a function of pressure is denoted by the gas-oil ratio (GOR), Rs. Gas emerges from solution when the reservoir pressure falls below the oil/gas bubble point pressure at the given temperature. At this time, gas remains distributed between the oil phase and a free gas phase. Let the subscripts o, w and g represent oil, water and gas components respectively; then the conservation equations for three phases are

$$-\nabla \cdot u_o = \frac{\partial}{\partial t}\left(\frac{1}{B_o}\phi S_o\right) + q_o, \tag{2.41}$$

$$-\nabla \cdot u_w = \frac{\partial}{\partial t}\left(\frac{1}{B_w}\phi S_w\right) + q_w, \tag{2.42}$$

$$-\nabla \cdot (R_s u_o + u_g) = \frac{\partial}{\partial t}\left(\phi \frac{R_s}{B_o} S_o + \phi \frac{S_g}{B_g}\right) + R_s q_o + q_{fg}. \tag{2.43}$$

The flow equations 2.42 and 2.43 for oil and water are the same as the equations for two phases, which are well discussed in the previous sections. In this section only the gas flux term is considered.

$$F = \int (R_s u_o + u_g) \cdot \hat{n} ds = \int V \cdot \hat{n} ds. \tag{2.44}$$

where

$$V = R_s u_o + u_g. \tag{2.45}$$

Referring to Figure 2.2, the gas flux across aikj is

$$\begin{aligned} f_{0,aikj} &= \int_{aikj} \mathbf{V} \cdot \hat{n} ds \\ &= \int_{aikj} (V_x \hat{i} + V_y \hat{j} + V_z \hat{k}) \cdot (n_x \hat{i} + n_y \hat{j} + n_z \hat{k}) ds \\ &= \int_{aikj} (V_x n_x + V_y n_y + V_z n_z) ds. \end{aligned} \tag{2.46}$$

The volumetric flux is computed by Darcy's law:

$$u_l = -\frac{k_{rl}\rho_l}{B_l \mu_l} gk \nabla h_l. \tag{2.47}$$

where k is a tensor. The head h is defined as

$$h = \frac{P}{\rho g} + z. \tag{2.48}$$

Substituting equation 2.47 into equation 2.45, then the flux V is

$$V = -(R_s T_o \nabla h_o + T_g \nabla h_g). \tag{2.49}$$

where

$$T_l = \frac{k_{rl}\rho_l}{B_l \mu_l} gk. \tag{2.50}$$

In three dimensional space,

$$\begin{aligned} V_x = &- R_s \left(T_{oxx} \frac{\partial h_o}{\partial x} + T_{oxy} \frac{\partial h_o}{\partial y} + T_{oxz} \frac{\partial h_o}{\partial z} \right) \\ &- \left(T_{gxx} \frac{\partial h_g}{\partial x} + T_{gxy} \frac{\partial h_g}{\partial y} + T_{gxz} \frac{\partial h_g}{\partial z} \right), \end{aligned} \tag{2.51a}$$

$$\begin{aligned} V_y = &- R_s \left(T_{oyx} \frac{\partial h_o}{\partial x} + T_{oyy} \frac{\partial h_o}{\partial y} + T_{oyz} \frac{\partial h_o}{\partial z} \right) \\ &- \left(T_{gyx} \frac{\partial h_g}{\partial x} + T_{gyy} \frac{\partial h_g}{\partial y} + T_{gyz} \frac{\partial h_g}{\partial z} \right), \end{aligned} \tag{2.51b}$$

$$
\begin{aligned}
V_z = &- R_s \left(T_{ozx} \frac{\partial h_o}{\partial x} + T_{ozy} \frac{\partial h_o}{\partial y} + T_{ozz} \frac{\partial h_o}{\partial z} \right) \\
&- \left(T_{gzx} \frac{\partial h_g}{\partial x} + T_{gzy} \frac{\partial h_g}{\partial y} + T_{gzz} \frac{\partial h_g}{\partial z} \right).
\end{aligned} \tag{2.51c}
$$

The gas flux across aikj in Figure 2.2 is expanded by substituting equations 2.32 and 2.51 into 2.46

$$
\begin{aligned}
&f_{0,aikj} \\
&= \int_{aikj} - \left[R_s \left(T_{oxx} \frac{\partial h_o}{\partial x} + T_{oxy} \frac{\partial h_o}{\partial y} + T_{oxz} \frac{\partial h_o}{\partial z} \right) - \left(T_{gxx} \frac{\partial h_g}{\partial x} + T_{gxy} \frac{\partial h_g}{\partial y} + T_{gxz} \frac{\partial h_g}{\partial z} \right) \right] \\
&\quad \frac{(y_3 - y_a)(z_2 - z_a) - (y_2 - y_a)(z_3 - z_a)}{|\vec{a3} \times \vec{a2}|} \\
&\quad - \left[R_s \left(T_{oyx} \frac{\partial h_o}{\partial x} + T_{oyy} \frac{\partial h_o}{\partial y} + T_{oyz} \frac{\partial h_o}{\partial z} \right) - \left(T_{gyx} \frac{\partial h_g}{\partial x} + T_{gyy} \frac{\partial h_g}{\partial y} + T_{gyz} \frac{\partial h_g}{\partial z} \right) \right] \\
&\quad \frac{(x_2 - x_a)(z_3 - z_a) - (x_3 - x_a)(z_2 - z_a)}{|\vec{a3} \times \vec{a2}|} \\
&\quad - \left[R_s \left(T_{ozx} \frac{\partial h_o}{\partial x} + T_{ozy} \frac{\partial h_o}{\partial y} + T_{ozz} \frac{\partial h_o}{\partial z} \right) - \left(T_{gzx} \frac{\partial h_g}{\partial x} + T_{gzy} \frac{\partial h_g}{\partial y} + T_{gzz} \frac{\partial h_g}{\partial z} \right) \right] \\
&\quad \frac{(x_3 - x_a)(y_2 - y_a) - (x_2 - x_a)(y_3 - y_a)}{|\vec{a3} \times \vec{a2}|} ds.
\end{aligned} \tag{2.52}
$$

After taking the integrand out and canceling $|\vec{a3} \times \vec{a2}|$ with $\int_{aikj} ds$, the flux

becomes

$$
\begin{aligned}
&f_{0,aikj} \\
&= \frac{1}{12} R_s \frac{k_{ro01}\rho_{o01}}{B_{o01}\mu_{o01}} g \\
&\quad \left\{ \left(k_{oxx}\frac{\partial h_o}{\partial x} + k_{oxy}\frac{\partial h_o}{\partial y} + k_{oxz}\frac{\partial h_o}{\partial z} \right) [(y_3 - y_a)(z_2 - z_a) - (y_2 - y_a)(z_3 - z_a)] \right. \\
&\quad + \left(k_{oyx}\frac{\partial h_o}{\partial x} + k_{oyy}\frac{\partial h_o}{\partial y} + k_{oyz}\frac{\partial h_o}{\partial z} \right) [(x_2 - x_a)(z_3 - z_a) - (x_3 - x_a)(z_2 - z_a)] \\
&\quad \left. + \left(k_{ozx}\frac{\partial h_o}{\partial x} + k_{ozy}\frac{\partial h_o}{\partial y} + k_{ozz}\frac{\partial h_o}{\partial z} \right) [(x_3 - x_a)(y_2 - y_a) - (x_2 - x_a)(y_3 - y_a)] \right\} \\
&\quad + \frac{1}{12} \frac{k_{rg01}\rho_{g01}}{B_{g01}\mu_{g01}} g \\
&\quad \left\{ \left(k_{gxx}\frac{\partial h_g}{\partial x} + k_{gxy}\frac{\partial h_g}{\partial y} + k_{gxz}\frac{\partial h_g}{\partial z} \right) [(y_3 - y_a)(z_2 - z_a) - (y_2 - y_a)(z_3 - z_a)] \right. \\
&\quad + \left(k_{gyx}\frac{\partial h_g}{\partial x} + k_{gyy}\frac{\partial h_g}{\partial y} + k_{gyz}\frac{\partial h_g}{\partial z} \right) [(x_2 - x_a)(z_3 - z_a) - (x_3 - x_a)(z_2 - z_a)] \\
&\quad \left. + \left(k_{gzx}\frac{\partial h_g}{\partial x} + k_{gzy}\frac{\partial h_g}{\partial y} + k_{gzz}\frac{\partial h_o}{\partial z} \right) [(x_3 - x_a)(y_2 - y_a) - (x_2 - x_a)(y_3 - y_a)] \right\}.
\end{aligned}
\tag{2.53}
$$

Repeating the same procedure, the gas fluxes across bjkh, chki, dgkj, eikj and fgkh can be derived.

Chapter 3

The Mixed Finite Element Method

3.1 Introduction

The mixed finite element (MFE) method was developed by Raviart and Thomas [24]. To understand this method, consider a second order elliptic model problem on a bounded domain Ω with a Lipshitz continuous boundary Γ:

$$-\nabla \cdot \underline{u} = f \qquad \text{in} \quad \Omega \tag{3.1}$$

$$\underline{u} = \nabla p \qquad \text{in} \quad \Omega \tag{3.2}$$

$$p = 0 \qquad \text{on} \quad \Gamma \tag{3.3}$$

The variational form of equations (3.1) and (3.2) can be written as

$$\int_\Omega w\,(\nabla \cdot \underline{u} + f)\,dx = 0 \quad \forall\, w \in L^2(\Omega), \tag{3.4}$$

$$\int_\Omega \underline{v} \cdot \underline{u}\,dx + \int_\Omega p\,\nabla \cdot \underline{v}\,dx = 0 \quad \forall\, \underline{v} \in \underline{H}(\text{div};\Omega), \tag{3.5}$$

respectively.

Raviart and Thomas [24] proved that equations (3.4) and (3.5) have a unique solution

$$(\underline{u}, p) \in \underline{H}(\text{div};\Omega) \times L^2(\Omega). \tag{3.6}$$

3.1.1 The lowest order Raviart–Thomas space

The zero order Raviart–Thomas space (RT_0) and triangular elements are used in this research work. The reason for not using higher order Raviart–Thomas space is because of the requirement of upstream weighting, and the reason for using triangular elements is because of the complexity of the geometrical domain.

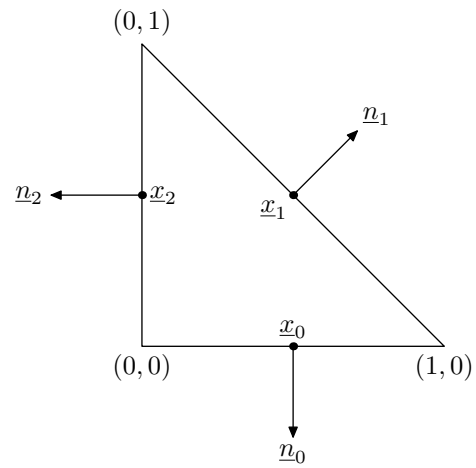

Figure 3.1: Unit outward normals on triangle's three edges.

RT_0 is a special solution space and is normally not well known to people with an engineering mathematical background. The definition of this space is explained, and the properties of this space is discussed.

RT_0 defines a vector space in multidimensional domains. In triangular elements, RT_0 is defined as

$$\underline{v} = (a + c\,x, b + c\,y) \tag{3.7}$$

where the underline denotes that v is a vector, and a, b and c are real numbers. It is more convenient to write $\underline{v}$ as a combination of three orthogonal (independent) bases in practice.

$$\underline{v} = v_0\,\underline{v}_0 + v_1\,\underline{v}_1 + v_2\,\underline{v}_2 \tag{3.8}$$

where $\underline{v}_0$, $\underline{v}_1$, and $\underline{v}_2$ are the orthogonal bases, and are defined by equation (3.7).

$$\underline{v}_i = (a_i + c_i x, b_i + c_i y). \tag{3.9}$$

These bases can be found by solving the following equations

$$(\underline{v}_i \cdot \underline{n}_j)\big|_{\underline{x}_j} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases} \quad \text{for } i, j = 0, 1, 2 \tag{3.10}$$

where $\underline{x}_j$ is the midpoint of triangle's side j, and $\underline{n}_j$ is the unit outward normal at $\underline{x}_j$ as shown in Figure 3.1.

The three bases for the triangle shown in Figure 3.1 are

$$\underline{v}_0 = (x, y - 1) \tag{3.11}$$

$$\underline{v}_1 = (\sqrt{2}\,x, \sqrt{2}\,y) \tag{3.12}$$

$$\underline{v}_2 = (x - 1, y). \tag{3.13}$$

It can be shown that

$$\int_t \underline{v}_i \cdot \underline{v}_j \, d\underline{x} = \begin{cases} 2\,|t|/3 & \text{if } i = j, \\ 0 & \text{if } i \neq j \end{cases} \tag{3.14}$$

where t is the triangle in Figure 3.1, and $|t|$ is the area of t.

3.2 Application of the MFEM in reservoir simulation

The mixed finite element method has been applied to miscible flow problem [25] and incompressible two-phase flow problem [26]. A triangle-based mixed finite element - finite volume formulation was developed for compositional reservoir simulation [27]. This formulation uses total flux approximation and IMPES scheme. An expanded mixed finite element method has been developed for accurate and efficient treatment of irregular domains [28]. The expanded method involves a coordinate mapping into one regular computational grid. When the matrix permeability is full-tensor, this formulation has 9-points stencil in two-dimensional space and 19-points stencil in three-dimensional space.

3.3 Multiphase MFE formulation

The extension of the MFE method from the model elliptic problem, equations (3.1) to (3.3), to a time-dependent, multiphase flow problem is not a simple matter. The extension is discussed here.

3.3.1 Multiphase flow equations

The governing equation for multiphase subsurface flow problems is the phase volumetric balance equation:

$$-\nabla \cdot \underline{u} = \frac{\partial}{\partial t}\left(\frac{\phi S}{B}\right) + q. \tag{3.15}$$

The phase subscripts for each variable are omitted for brevity. The volumetric flux is computed by the multiphase Darcy's law:

$$\underline{u} = -\frac{k_r \rho}{B\mu}\, g\, \underline{\underline{k}} \nabla h \tag{3.16}$$

where the double underlines indicate that k is a tensor. The head is defined as

$$h = \frac{P}{\rho g} + z. \tag{3.17}$$

The relative permeability k_r in equation (3.16) could degenerate to zero causing $\underline{u} = 0$ even though the head gradient is not zero, therefore $\underline{u}$ cannot be used as the flux in the MFEM context. Instead, the modified head gradient is used

$$\underline{f} = -\underline{\underline{k}}\nabla h, \tag{3.18}$$

and the multiphase Darcy's law becomes

$$\underline{u} = \frac{k_r \rho}{B\mu}\, g\, \underline{f} \tag{3.19}$$

The variational form of (3.15) and (3.18) are

$$\int_\Omega w \nabla \cdot \underline{u}\, dx + \int_\Omega w \frac{\partial}{\partial t}\left(\frac{\phi S}{B}\right)\, dx + \int_\Omega wq\, dx = 0, \tag{3.20}$$

$$\int_\Omega \underline{v} \cdot \underline{f}\, dx = -\int_\Omega \underline{v} \cdot \underline{k} \cdot \nabla h\, dx, \tag{3.21}$$

respectively.

3.3.2 Discretization

Consider solving the multiphase equations on a polygonal domain Ω, and the domain is partitioned (meshed) into a set of regular triangles $\mathcal{T}$. The choice of w in equation (3.20) for a triangle T in $\mathcal{T}$ is

$$w(\underline{x}) = \begin{cases} 1 & \text{if } \underline{x} \in T, \\ 0 & \text{if } \underline{x} \notin T. \end{cases} \tag{3.22}$$

Substituting $w(\underline{x})$ into equation (3.20), the variational balance equation becomes an integral equation

$$\int_T \nabla \cdot \underline{u}\, d\underline{x} + \int_T \frac{\partial}{\partial t}\left(\frac{\phi S}{B}\right)\, d\underline{x} + \int_T q\, d\underline{x} = 0. \tag{3.23}$$

The choice of $\underline{v}$ in equation (3.21) for t is

$$\underline{v}_i(\underline{x}) = \begin{cases} \underline{v}_i(\underline{x}) & \text{if } \underline{x} \in T, \\ 0 & \text{if } \underline{x} \notin T \end{cases} \quad \text{for } i = 0, 1, 2. \tag{3.24}$$

After substituting equation (3.24) into equation (3.21), the variational flux equation becomes

$$\int_T \underline{v}_i \cdot \underline{f}\, d\underline{x} = -\int_T \underline{v}_i \cdot \underline{\underline{k}} \cdot \nabla h\, d\underline{x}, \quad \text{for } j = 0, 1, 2. \tag{3.25}$$

There will be a volumetric balance equation and three volumetric flux equations for each triangle in $\mathcal{T}$.

The solution space for h and S is piecewise constant, and the space for $\underline{f}$ is RT_0. Consequently, the integral equation (3.23) can be simplified to

$$\sum_{i=0}^{2}(\underline{u}\cdot\underline{n})\big|_{\underline{x}_i}\, l_i + \frac{|T|}{\Delta t}\left[\left(\frac{\phi S}{B}\right) - \left(\frac{\phi S}{B}\right)\right] + |T|\; q = 0 \tag{3.26}$$

where l_i is the length of side i, $|T|$ is the area of T. Replacing $\underline{f}$ on the left of equation (3.25) with equation (3.8), the equation is simplified to

$$\int_T \underline{v}_i \cdot \underline{f}\; d\underline{x} = \frac{2\,|T|}{3} f_i. \tag{3.27}$$

Applying integration by parts to the right of equation (3.25),

$$\begin{aligned} -\int_T \underline{v}_i \cdot \underline{\underline{k}}\,\nabla h \cdot d\underline{x} &= \int_T h\,\nabla\!\cdot\underline{q}_i\; d\underline{x} - \int_T \nabla\!\cdot(\underline{q}_i h)\; d\underline{x} \\ &= \int_T h\,\nabla\!\cdot\underline{q}_i\; d\underline{x} - \int_{\partial T} \overline{h}\;\underline{q}_i\cdot\underline{n}\; ds \\ &= h\,|T|\,\nabla\!\cdot\underline{q}_i - \sum_{j=0}^{2}\overline{h}_j\,(\underline{q}_j\cdot\underline{n}_j)\big|_{\underline{x}_j}\, l_j \end{aligned} \tag{3.28}$$

where

$$\underline{q}_i = \underline{v}_i \cdot \underline{\underline{k}} \tag{3.29}$$

and $\overline{h}_j$ is the phase head on T's side j. Equate equations (3.27) and (3.28).

$$\frac{2\,|T|}{3} f_i = h\,|T|\,\nabla\!\cdot\underline{q}_i - \sum_{j=0}^{2}\overline{h}_j\,(\underline{q}_j\cdot\underline{n}_j)\big|_{\underline{x}_j}\, l_j. \tag{3.30}$$

Therefore,

$$f_i = \frac{3}{2} h\nabla\!\cdot\underline{q}_i - \frac{3}{2\,|T|}\sum_{j=0}^{2}\overline{h}_j\,(\underline{q}_j\cdot\underline{n}_j)\big|_{\underline{x}_j}\, l_j. \tag{3.31}$$

The first term of equation (3.26) can be simplified to

$$\begin{aligned} \sum_{i=0}^{2}\underline{u}\big|_{\underline{x}_i}\cdot\underline{n}_i\, l_i &= \sum_{i=0}^{2}\frac{k_r\rho}{B\mu} g\;\underline{f}\big|_{\underline{x}_i}\cdot\underline{n}_i\, l_i \\ &= \sum_{i=0}^{2}\frac{k_r\rho}{B\mu} g\; f_i\, l_i \end{aligned} \tag{3.32}$$

The final volume balance equation for T is

$$\sum_{i=0}^{2}\frac{k_r\rho}{B\mu} g\; f_i\; l_i + \frac{|T|}{\Delta t}\left[\left(\frac{\phi S}{B}\right)^{n+1} - \left(\frac{\phi S}{B}\right)^{n}\right] + |T|\; q = 0 \tag{3.33}$$

where f_i is defined by equation (3.31).

3.3.3 Discretized multiphase flow equations

The discretized multiphase flow equations are derived in this section. The volumetric balance equation (3.15) is rewritten here for convenience.

$$-\nabla \cdot \underline{u} = \frac{\partial}{\partial t}\left(\frac{\phi S}{B}\right) + q. \tag{3.34}$$

Let the subscripts o, w and g represent oil, water and gas phases, respectively, then the three-phase volumetric balance equations are

$$-\nabla \cdot \underline{u}_o = \frac{\partial}{\partial t}\left(\frac{\phi S_o}{B_o}\right) + q_o, \tag{3.35}$$

$$-\nabla \cdot \underline{u}_w = \frac{\partial}{\partial t}\left(\frac{\phi S_w}{B_w}\right) + q_w, \tag{3.36}$$

$$-\nabla \cdot \underline{u}_g - R_s \nabla \cdot \underline{u}_o = \frac{\partial}{\partial t}\left(\frac{\phi S_g}{B_g} + R_s \frac{\phi S_o}{B_o}\right) + q_g + R_s q_o. \tag{3.37}$$

Referring to equation (3.33), the discretized three-phase equations for triangle T can be written as

$$0 = \sum_{i=0}^{2} \frac{k_{ro}\rho_o}{B_o \mu_o} g\ f_{io}\ l_i + \frac{|T|}{\Delta t}\left[\left(\frac{\phi S_o}{B_o}\right)^{n+1} - \left(\frac{\phi S_o}{B_o}\right)^{n}\right] + |T|\ q_o, \tag{3.38}$$

$$0 = \sum_{i=0}^{2} \frac{k_{rw}\rho_w}{B_w \mu_w} g\ f_{iw}\ l_i + \frac{|T|}{\Delta t}\left[\left(\frac{\phi S_w}{B_w}\right)^{n+1} - \left(\frac{\phi S_w}{B_w}\right)^{n}\right] + |T|\ q_w, \tag{3.39}$$

$$\begin{aligned} 0 = &\sum_{i=0}^{2} \frac{k_{rg}\rho_g}{B_g \mu_g} g\ f_{ig}\ l_i + \frac{|T|}{\Delta t}\left[\left(\frac{\phi S_g}{B_g}\right)^{n+1} - \left(\frac{\phi S_g}{B_g}\right)^{n}\right] + |T|\ q_g \\ &+ \sum_{i=0}^{2} R_s \frac{k_{ro}\rho_o}{B_o \mu_o} g\ f_{io}\ l_i + \frac{|T|}{\Delta t}\left[\left(R_s \frac{\phi S_o}{B_o}\right)^{n+1} - \left(R_s \frac{\phi S_o}{B_o}\right)^{n}\right] + |T|\ R_s q_o. \end{aligned} \tag{3.40}$$

The modified head gradients for each phase on each side of T are

$$f_{io} = \frac{3}{2} h_o \nabla \cdot \underline{q}_i - \frac{3}{2\,|T|} \sum_{j=0}^{2} \overline{h}_{jo} \left(\underline{q}_j \cdot \underline{n}_j\right)\Big|_{\underline{x}_j} l_j, \tag{3.41}$$

$$f_{iw} = \frac{3}{2} h_w \nabla \cdot \underline{q}_i - \frac{3}{2\,|T|} \sum_{j=0}^{2} \overline{h}_{jw} \left(\underline{q}_j \cdot \underline{n}_j\right)\Big|_{\underline{x}_j} l_j, \tag{3.42}$$

$$f_{ig} = \frac{3}{2} h_g \nabla \cdot \underline{q}_i - \frac{3}{2\,|T|} \sum_{j=0}^{2} \overline{h}_{jg} \left(\underline{q}_j \cdot \underline{n}_j\right)\Big|_{\underline{x}_j} l_j. \tag{3.43}$$

3.3.4 Flux continuity

Equations (3.41) to (3.43) are defined as the fluxes flowing out of a triangle through side i. Flux continuity between neighbor triangles was not imposed in those equations. Three additional balance equations for each edge are necessary to enforce flux continuity. Consider two triangles T_1 and T_2 sharing a common edge γ. The flux continuity is ensured by solving the following equations

$$0 = f_{\gamma o}^{T_1} + f_{\gamma o}^{T_2} \tag{3.44}$$

$$0 = f_{\gamma w}^{T_1} + f_{\gamma w}^{T_2} \tag{3.45}$$

$$0 = f_{\gamma g}^{T_1} + f_{\gamma g}^{T_2}. \tag{3.46}$$

The superscript indicates which triangle the flux is originated from. The first subscript indicates the edge these two fluxes are flowing through. The second subscript indicates phase.

Chapter 4

Verification

A synthesized oil reservoir was used to compare the UFES (Utah Finite Element Simulator) and Eclipse (a commercial oil reservoir simulator). The reservoir measures 1000 feet in the ease-west direction, 500 feet in the north-south direction, and is 50 feet thick (Figure 4.1). Two horizontal wells were modeled. The horizontal production well was placed in the top layer on the north side and the horizontal injection well was placed in the bottom layer on the south side.

Primary oil production was begun at a production liquid rate of 1000 stb/day and was continued for 90 days. At 90 days, the production liquid rate is reduced to 600 stb/day and water injection was started. At 3600 days, the injection and production were stopped. This scenario was constructed to capture the essentials of the three-phase flow processes; oil and gas flow, gas evolution, gas pressurization, and oil and water flow.

Two meshes were used to study the production process. The first mesh has only one layer, and therefore gravitational effect is inconsequential. The second mesh has two layers and gas segregation and water under-ride due to gravity are expected. These two case studies are described in the following sections.

4.1 One-layer case study

The simulation domain is discretized into 40×20×1 blocks for Eclipse. As for UFES, the domain is discretized into 1330 tetrahedrals all expanded over entire thickness. Plots of cumulative oil production versus time are presented in Figures 4.3 and 4.4. Figures 4.5 and 4.6 are the oil production rate curves. Water cut data are shown in Figures 4.7 and 4.8. Gas to oil ratios are shown in Figures 4.9 and 4.10.

4.2 Two-layer case study

The domain of interest was divided into two layers. The grid used for Eclipse is 20×10×2 blocks. The grid for the UFES has two layers and each layer is further

discretized into 300 tetrahedrals. Figures 4.12 and 4.13 present the cumulative oil production curve for this case study. The oil production rate plots are shown in Figures 4.14 and 4.15. Water cut plots are shown in Figures 4.16 and 4.17. Gas to oil ratios are presented in Figures 4.18 and 4.19.

4.3 Summary

It should be emphasized that these are fundamentally very different simulators. The finite-element simulator has been developed primarily for representing complex domains including faulted and fractured systems. It is also appropriate for modeling complex well systems (multilaterals, fish-bones, etc.) The well models for UFES were developed at the University of Utah. Considering the fundamentally different discretization, and different well models, the match between the three-phase UFES and Eclipse is excellent. A new three-dimensional, three-phase, finite-element simulator is available for use. We would like to study the application of the model to complex geologic systems.

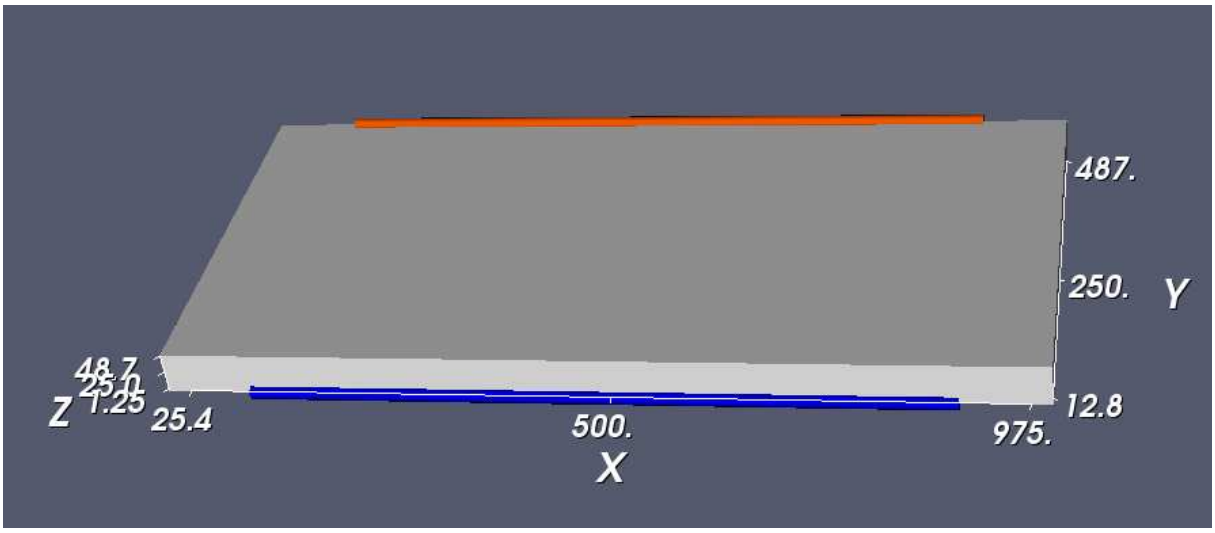

Figure 4.1: The domain measures 1000 ft in the x-direction, 500 ft in the y-direction and 50 ft in the z-direction. The horizontal production well is represented by the the orange line and the horizontal injection well is represented by the blue line. Wells are placed at the center of the domain along the x-direction, and each well measures 800 ft in length.

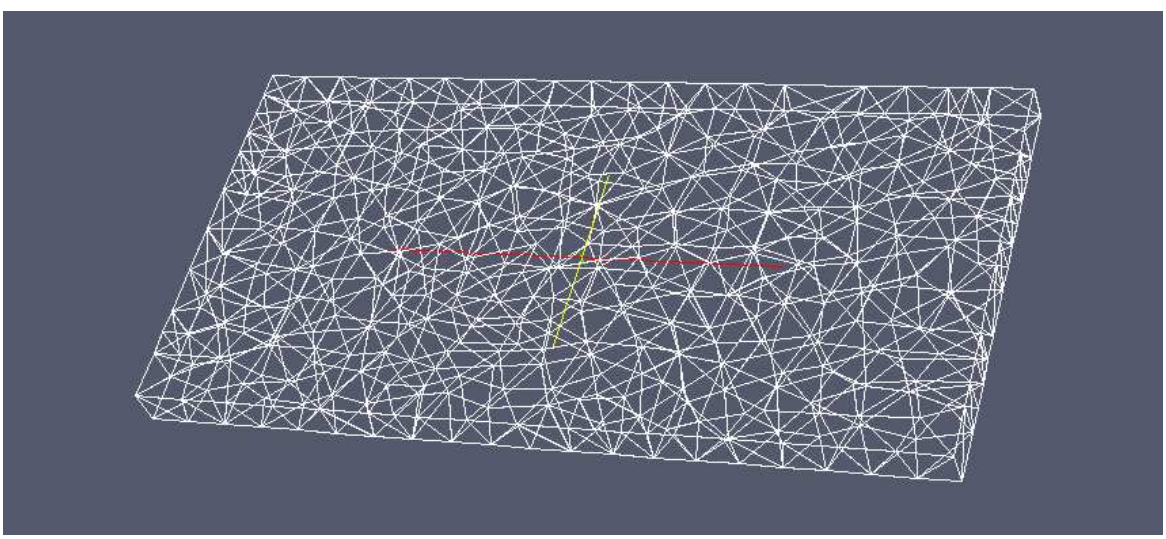

Figure 4.2: The mesh used for the one-layer case study for the UFES.

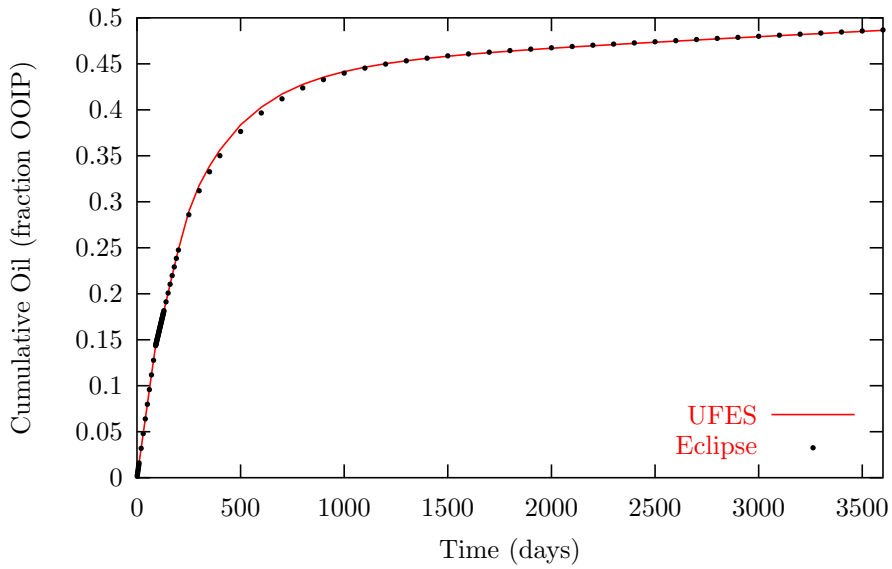

Figure 4.3: Cumulative oil production for the one-layer case study.

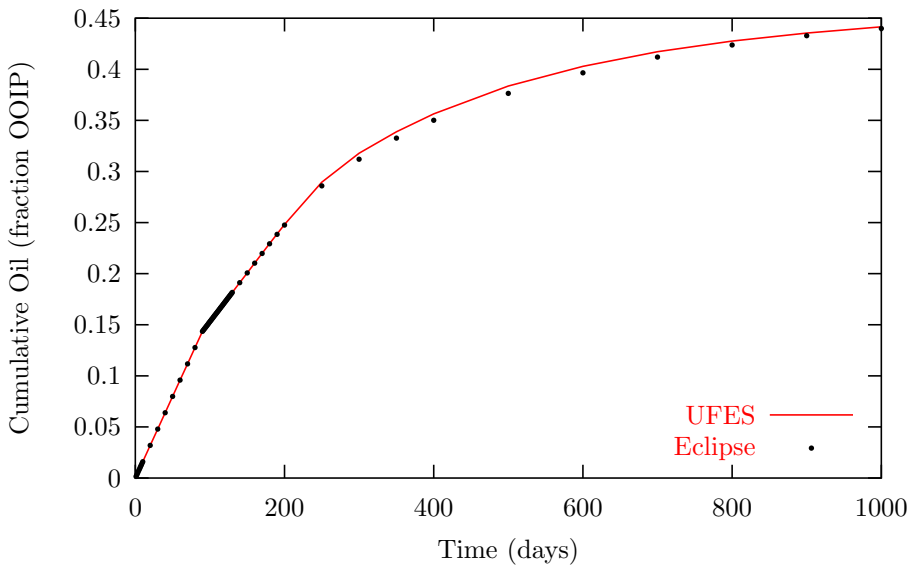

Figure 4.4: A more detailed cumulative oil production plot for the one-layer case study (between zero and 1000 days).

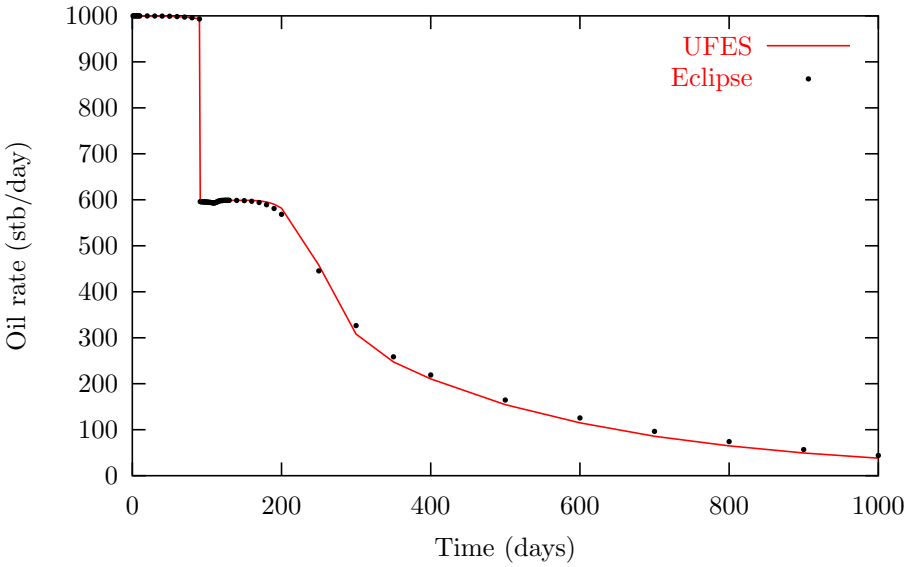

Figure 4.5: Oil production rate for the one-layer case study.

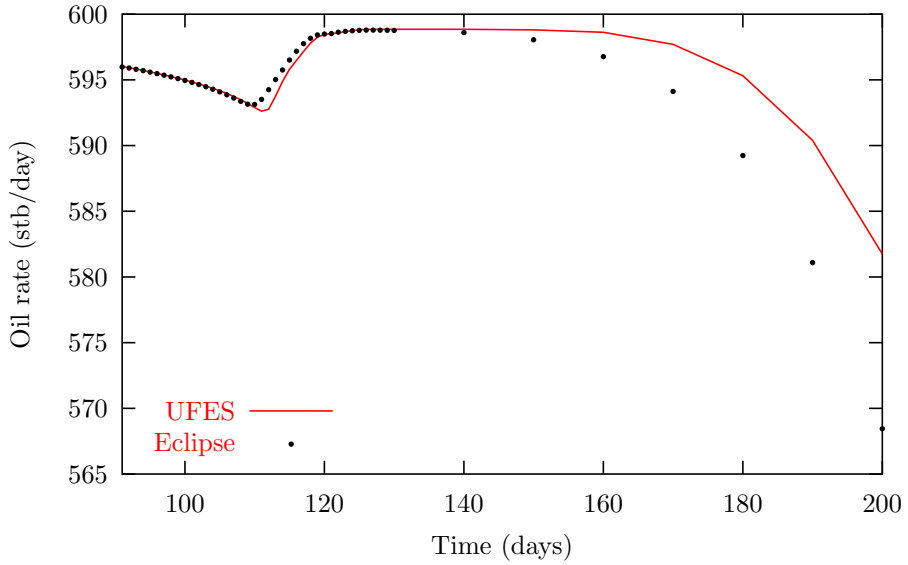

Figure 4.6: A more detailed oil production plot for the one-layer case study (between 91 and 200 days). Notice that the y-axis has a higher resolution.

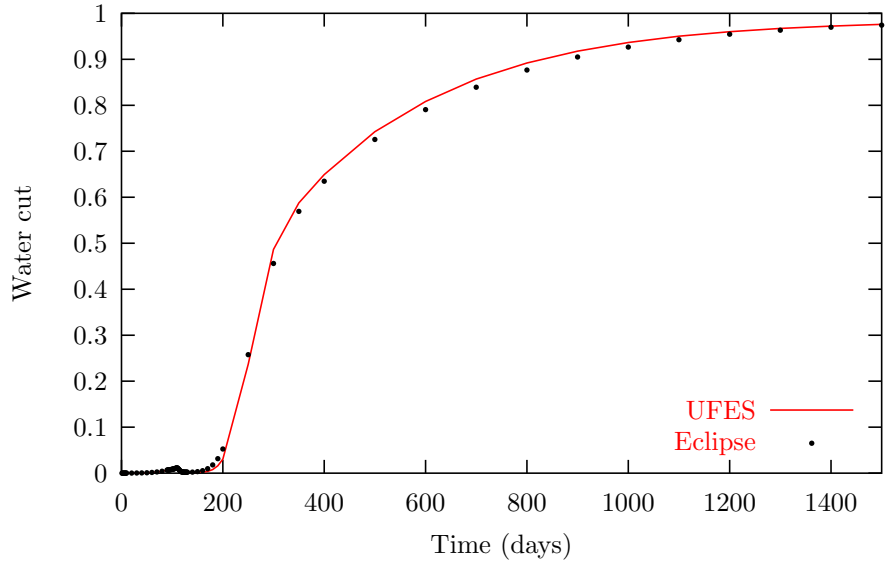

Figure 4.7: Water cut for the one-layer case study.

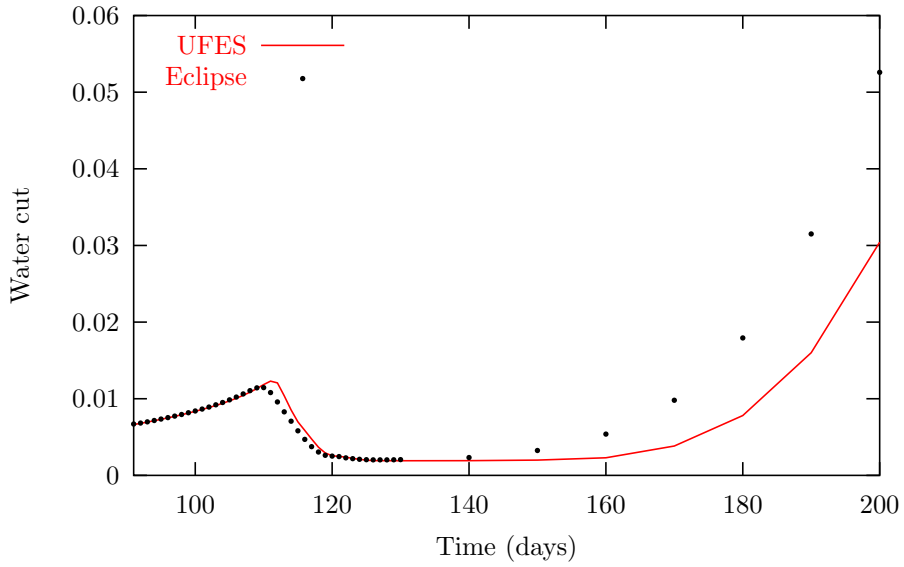

Figure 4.8: Water cut data between 91 and 200 days for the one-layer case study. Notice that the y-axis has a higher resolution

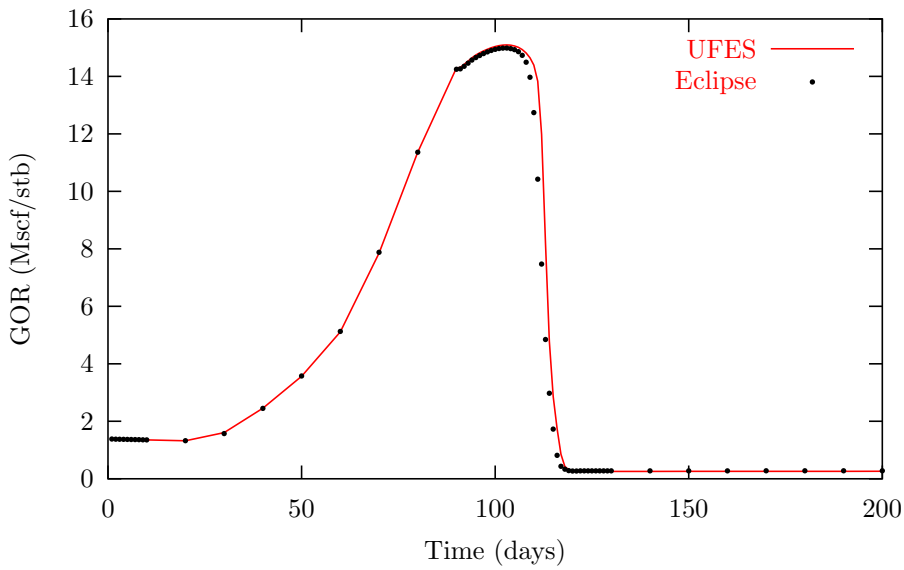

Figure 4.9: Gas oil ratio for the one-layer case study.

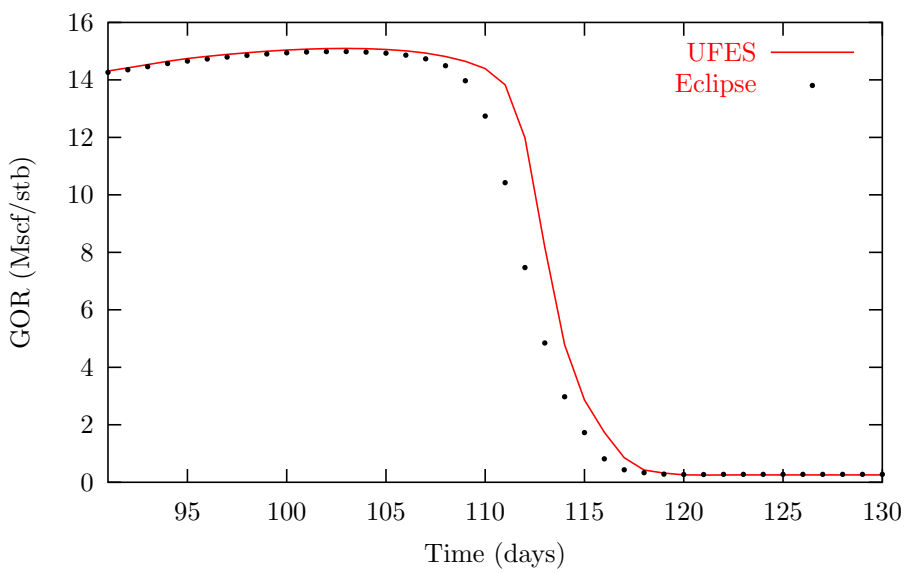

Figure 4.10: Gas oil ratio between 91 and 130 days for the one-layer case study.

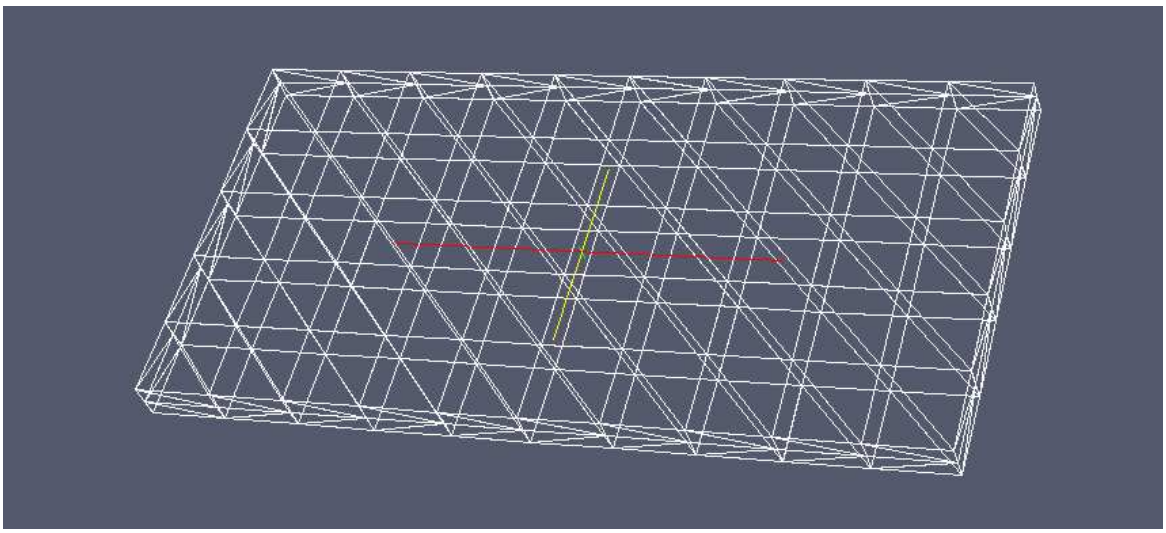

Figure 4.11: The mesh used for the two-layer case study for the UFES.

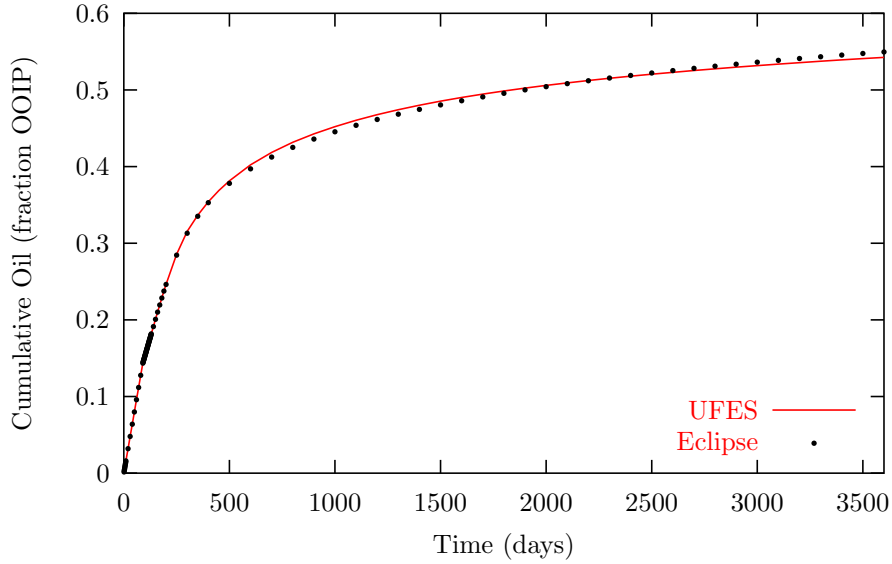

Figure 4.12: Cumulative oil production for the two-layer case study.

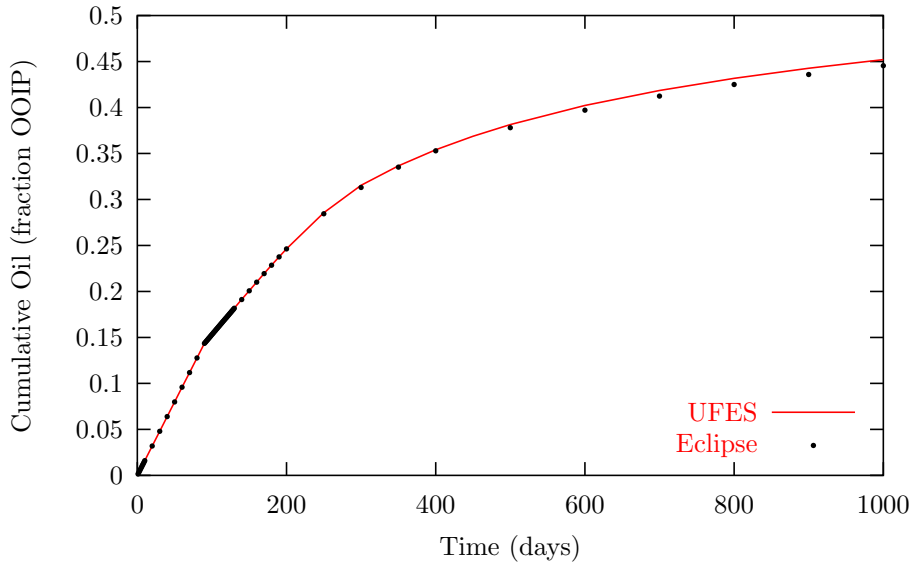

Figure 4.13: Cumulative oil production curve between 0 and 1000 days for the two-layer case study.

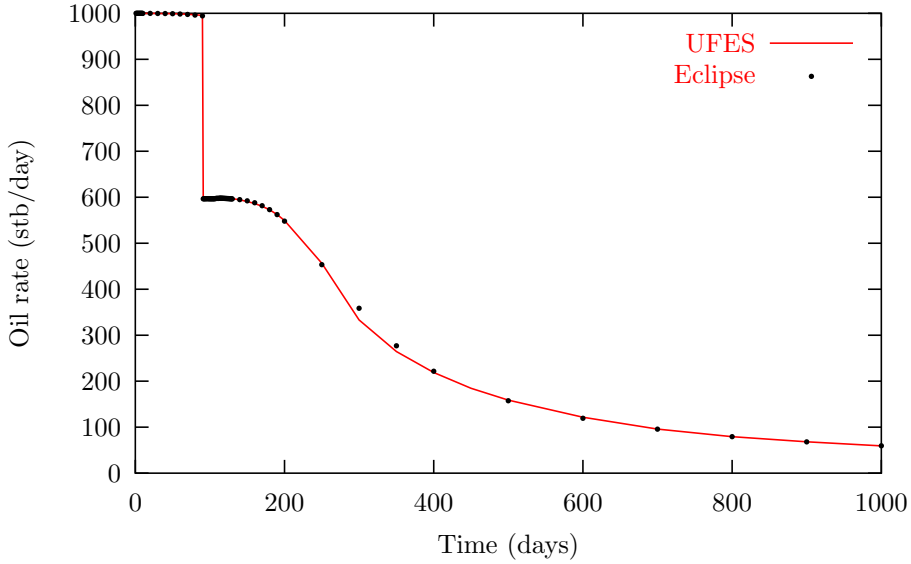

Figure 4.14: Oil production rate for the two-layer case study.

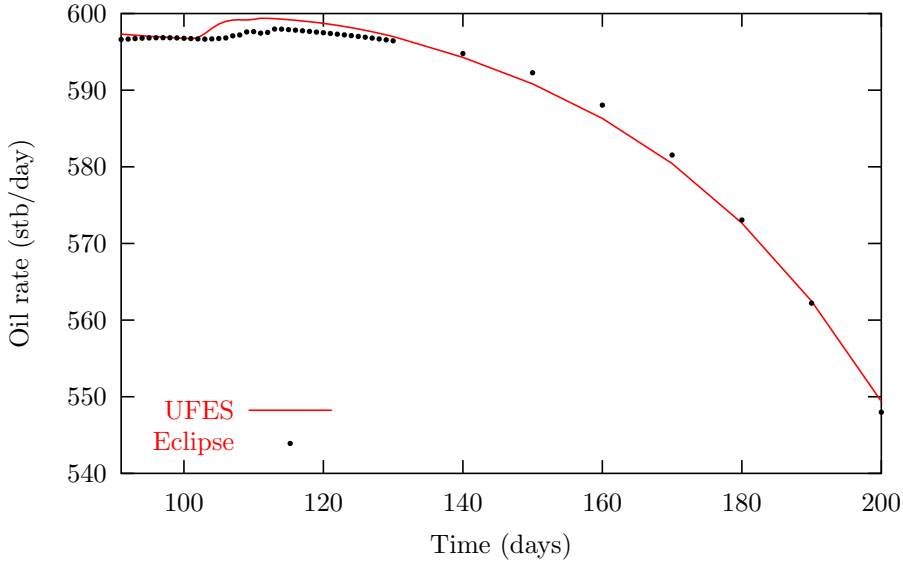

Figure 4.15: Oil production rate between 91 and 200 days for the two-layer case study.

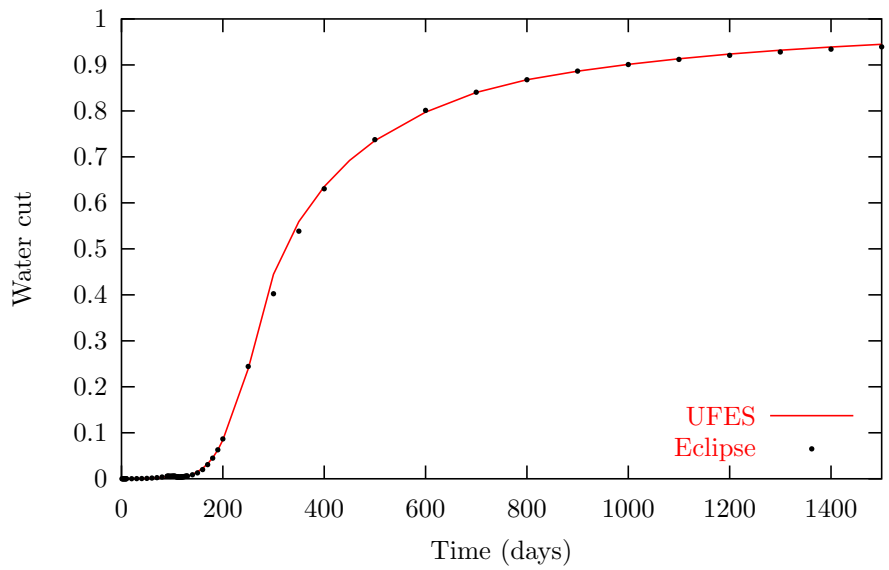

Figure 4.16: Water cut for the two-layer case study.

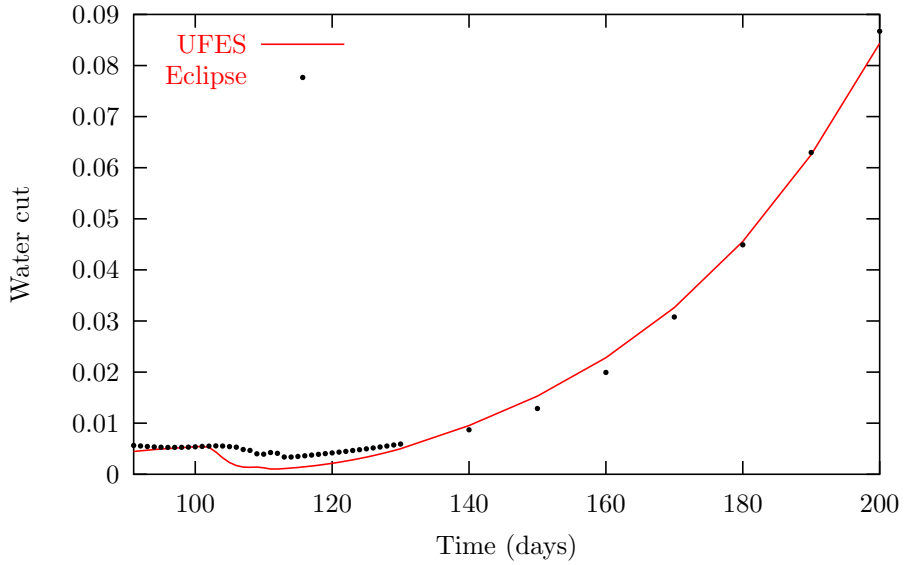

Figure 4.17: Water cut between 91 and 200 days for the two-layer case study.

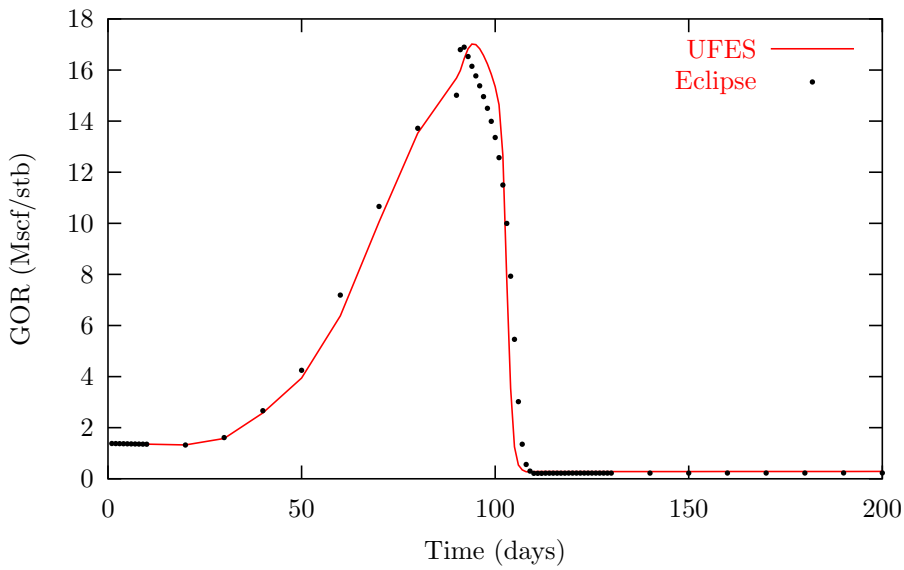

Figure 4.18: Gas oil ratio for the two-layer case study.

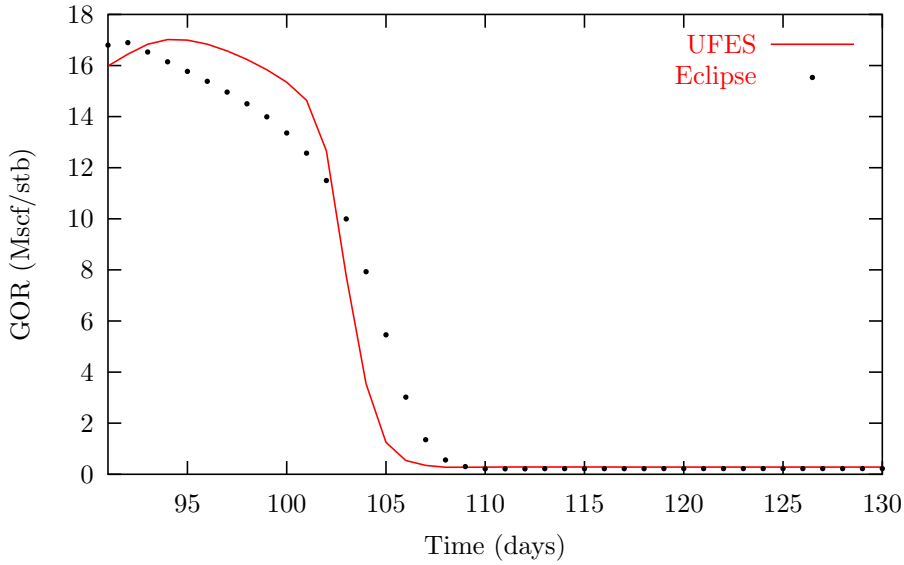

Figure 4.19: Gas oil ratio between 91 and 130 days for the two-layer case study.

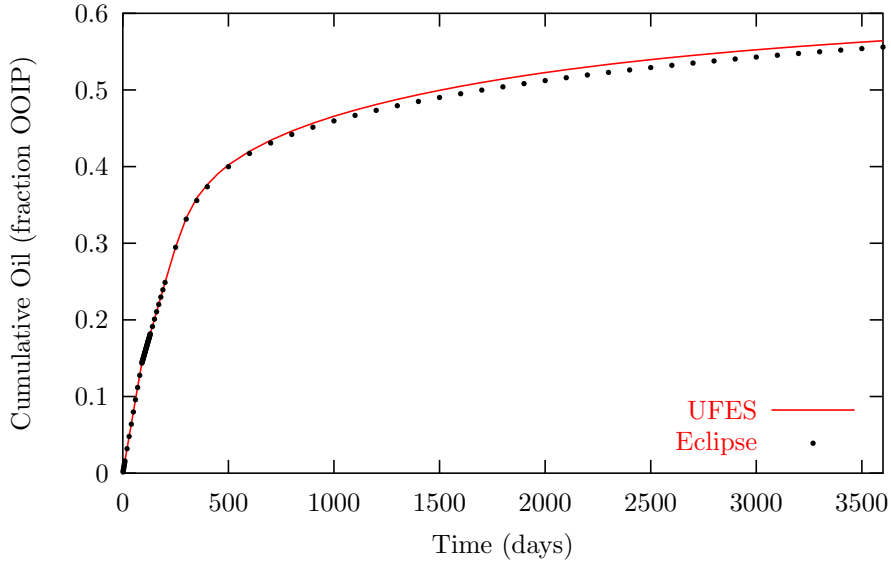

Figure 4.20: Cumulative oil production for the two-layer case study.

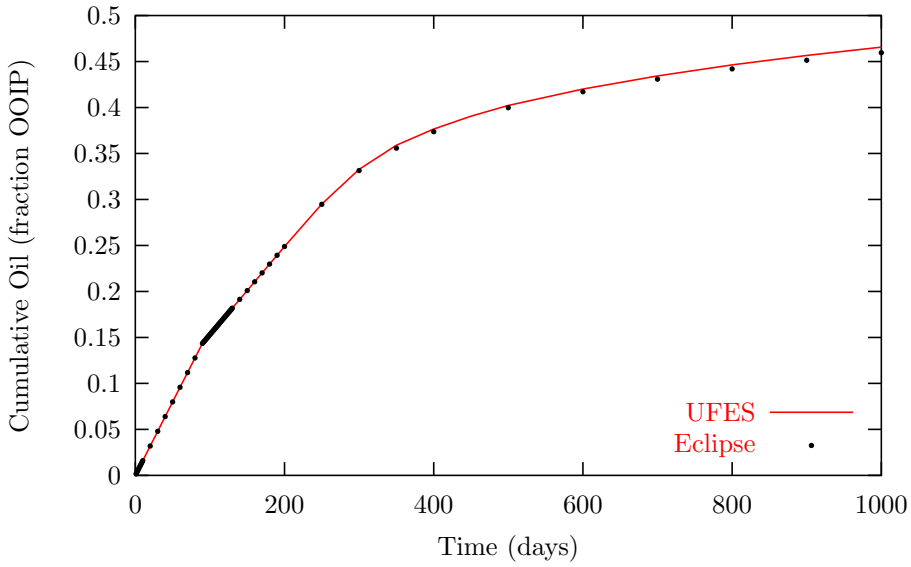

Figure 4.21: Cumulative oil production curve between 0 and 1000 days for the two-layer case study.

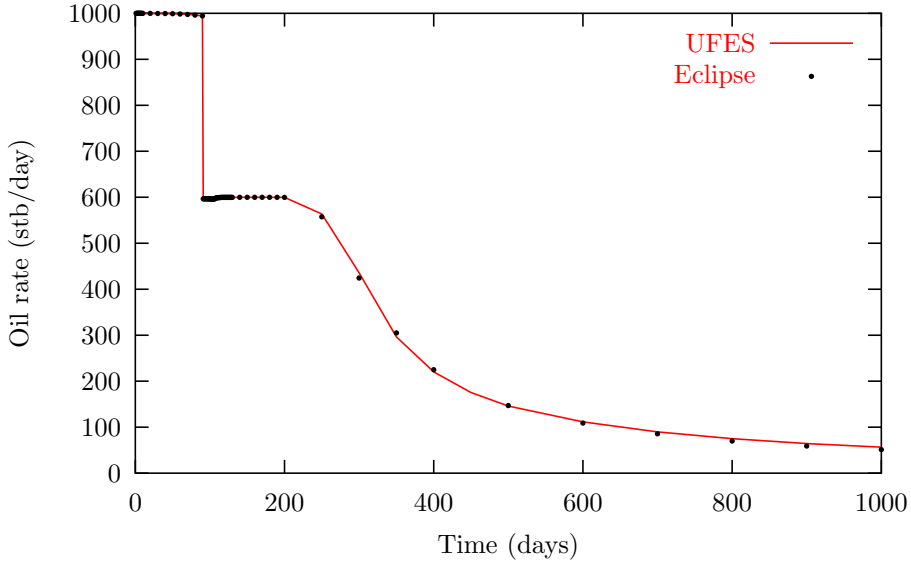

Figure 4.22: Oil production rate for the two-layer case study.

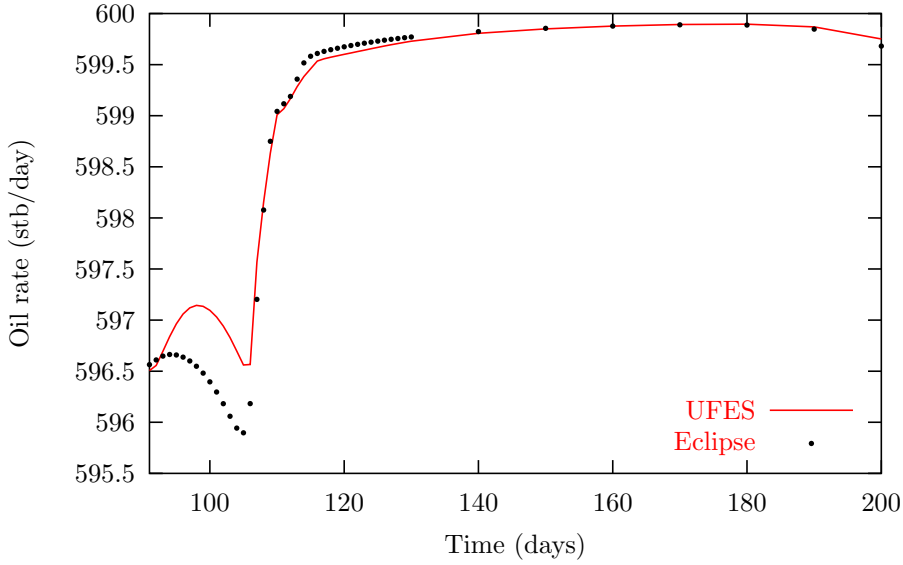

Figure 4.23: Oil production rate between 91 and 200 days for the two-layer case study.

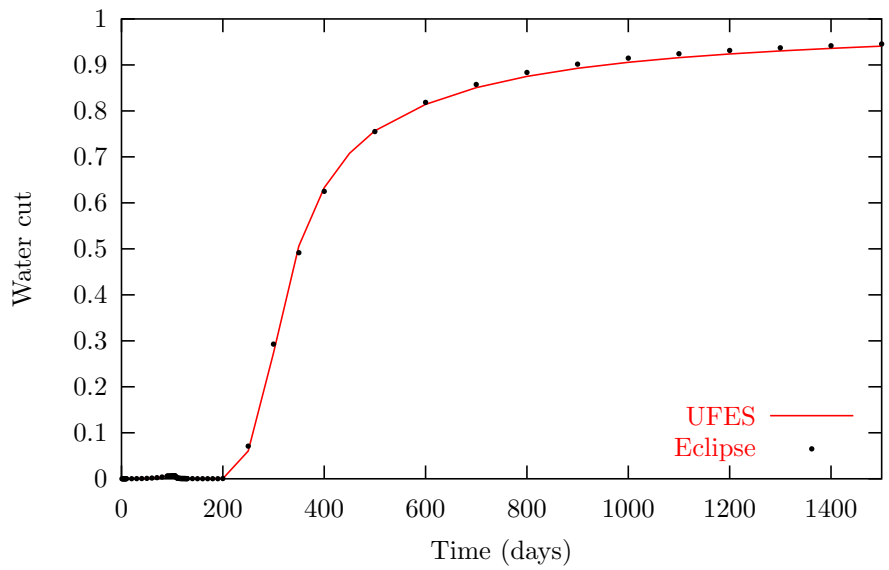

Figure 4.24: Water cut for the two-layer case study.

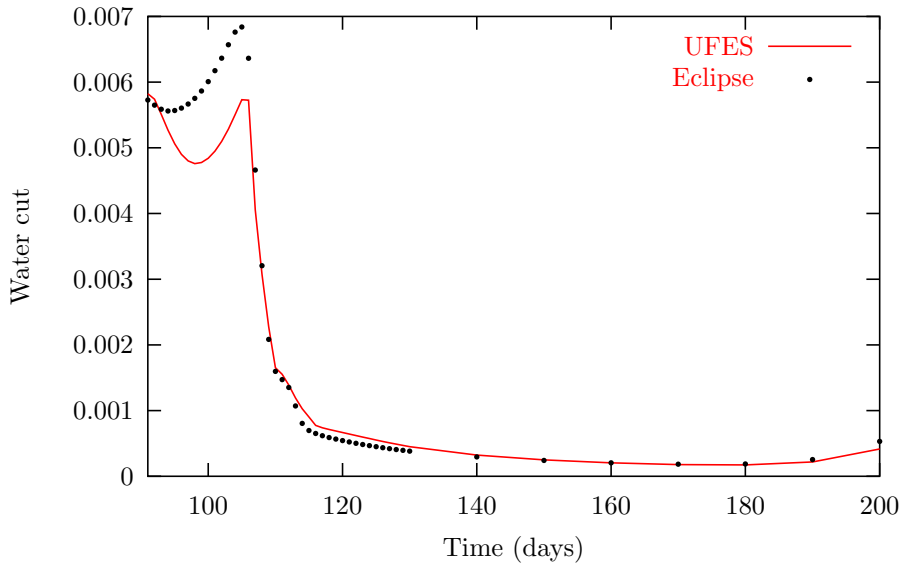

Figure 4.25: Water cut between 91 and 200 days for the two-layer case study.

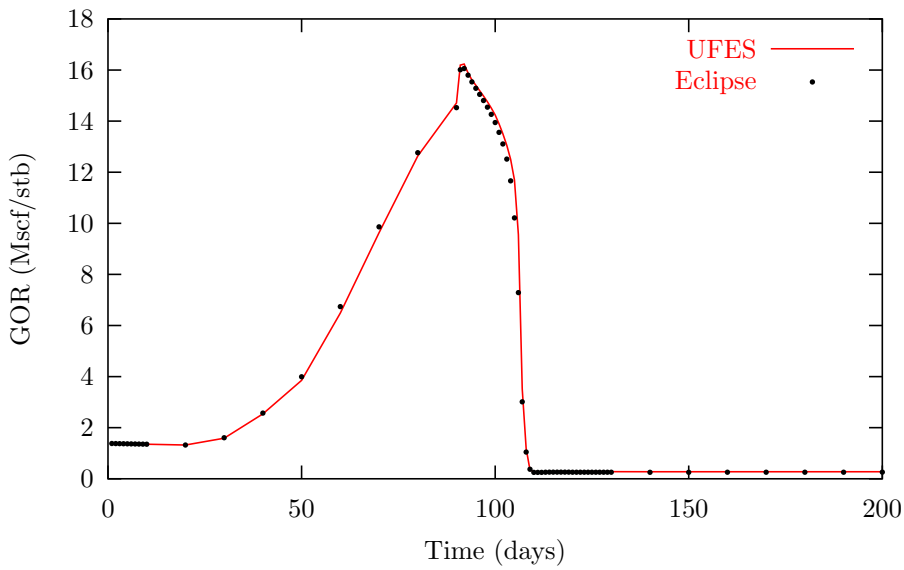

Figure 4.26: Gas oil ratio for the two-layer case study.

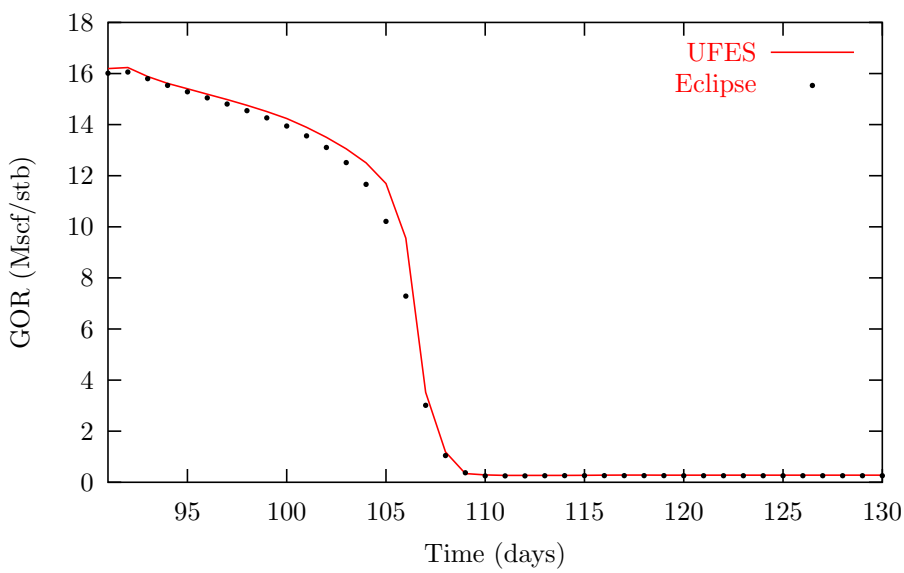

Figure 4.27: Gas oil ratio between 91 and 130 days for the two-layer case study.

Chapter 5

Fractured Reservoir Simulation

Naturally fractured oil reservoirs have quite different recovery characteristics compared to non-fractured reservoirs and therein consider the influence of fractures on fluid flow when production strategy is being planed is important.

There are currently four different fracture models being used in reservoir simulations: single continuum, dual porosity, dual permeability and discrete-fracture model. A discussion and comparison of these four models can be found in Kim [11]. In current research work, the focus is on the continuation of the discrete-fracture model development.

One version of a discrete-fracture model has been implemented by Kim [11] using the finite element method. The same set of governing equations are used for the fluid flow in matrix and in fracture. The final global stiffness matrix is obtained by superimposing the contribution from matrix and fracture. Validation of this model with single continuum model on a fine grid was attempted. Influence of different capillary pressures between fractures and matrix was studied. However, the superimposition was not explained in terms of the underlying physical implication.

A new version of a discrete-fracture model is developed in this research work. With the concept of control volumes and explicit calculation of fluxes, the discrete-fracture model can now be explained in a physically meaningful way.

A brief introduction to reservoir fractures is first presented in section 5.1. This is followed by the implementation of discrete-fracture model using the control volume method developed in Chapter 1. Simulations of a fractured offshore oil field are discussed.

5.1 Reservoir Rock Fractures

The formation, classification and evaluation of fractures are ongoing research topics themselves. Therefore, only a short introduction to these subjects are provided here for the completeness of fracture simulation.

A reservoir fracture is a macroscopic planar discontinuity in which a loss of cohesion of reservoir rock has taken place [29, 30]. A fracture can, therfore, be

considered as a rupture in reservoir rocks. Two of the most common types of fractures are faults and joints. A fault is a fracture along which one side has moved relative to the other [31]. A fracture is regarded as a joint when there is no noticeable displacement along the fracture [29]. Two dimensional representations of a fault and joints are shown in Figure 5.1 and Figure 5.2, respectively.

Two common causes of fractures are overburden pressure and tectonic forces. Figure 5.3 shows a fracture created by overbuden prssure and Figure 5.4 shows a pair of conjugated fractures created by tectonic forces.

Direct evaluation of fractures includes outcrop observation in the field and core examination in the laboratory [29]. Fracture opening (fracture width), fractrue filling and fracture orientation can be measured during the direct evaluation process. Indirect methods include well logs and seismic data. There are many well logging methods for the evaluation of fractures, for example, induction logs, the combination of sonic, neutron and density logs, gamma ray and borehole televiewer [30].

5.2 Fracture Information from the Field

Before the discussion of fractured reservoir simulation, it is necessary to understand what types of fracture information are available and are given to simulator developers. For simplicity, consider two-dimensional applications. The fracture information of an oil field given by geologists or reservoir engineers are in terms of straignt line segments. The permeability of each fracture is given or estimated from its width. It is not uncommon that the width of fractures are never known.

An example data set of a fractured domain Ω is given as follows. The domain with fractures is shown in Figure 5.5. The matrix permeability tensor is given as a function of the location, that is

$$\underline{k}^{\mathrm{m}} = \underline{k}^{\mathrm{m}}(\underline{x}) \tag{5.1}$$

where m stands for rock matrix or simply matrix and $\underline{x} \in \Omega$. Fracture properties are given in Table 5.1. The location and orientation of each fracture is defined

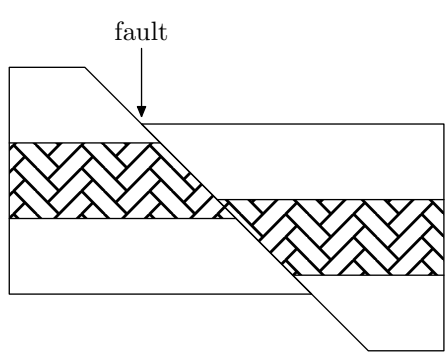

Figure 5.1: Cross-section view of a fault.

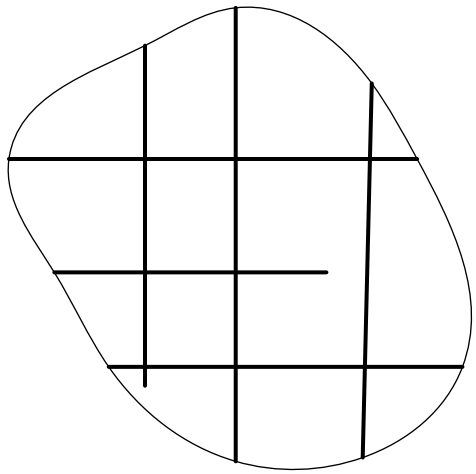

Figure 5.2: Areal view of some joints.

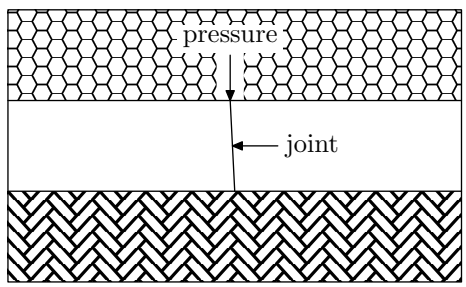

Figure 5.3: Cross-section view of the formation of a joint caused by overburden pressure.

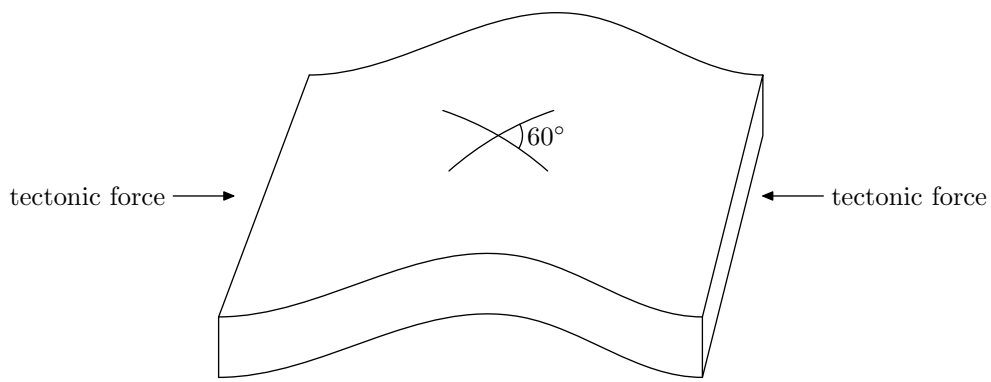

Figure 5.4: The formation of a pair of conjugated fractures due to tectonic force.

Table 5.1: Fracture properties available from oil field.

Fracture No.	Start	End	Permeability	width [1]
f_1	(x_1^s, y_1^s)	(x_1^e, y_1^e)	k_1	w_1
f_2	(x_2^s, y_2^s)	(x_2^e, y_2^e)	k_2	w_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

[1]The width of fractures is not always available.

by its starting and ending points. Fracture permeability k^{f} is given as a scalar which represents the permeability along the fracture line. The width of fractures is sometimes but not always available as shown in the last column of Table 5.1.

5.3 Single-Porosity Model

When the width information of fractures is available, the most straightforward and accurate approach to fracture simulation is the single porosity model. In this model, fractures are represented by very tiny or very thin two-dimensional elements.

Consider a square domain with one fracture as shown in Figure 5.6. To put the width of fractures into consideration, the line is replaced by a rectangle and is shown is Figure 5.7. It important to note that the width and the length of the fracture in Figure 5.7 are not in proportion; the width has been enlarged in order to be visible. In general the ratio of length to width of fractues is in the range of four to six orders. The triangular mesh of this single fracture domain is shown in Figure 5.8. It is clear that a trimendous amount of triangles is required by the single-porosity model even for a single, disproportional fracture. Consequentlly, the single-porosity model will need a lot more elements for a real fracture (a fracture that is in correct proportion) than the disproportional fracture.

5.4 Discrete-Fracture Model

In view of the limited fracture information that is available and the difficulty (requires too many elements) involved in the single porosity model, it is natural to consider the use of line elements to approximate fractures in two-dimensional space.

Consider the same square domain with one fracture as discussed in section 5.3. Without the introduction of fracture width, the domain is directly triangulated and the result is shown in Figure 5.9. Comparing Figure 5.8 with Figure 5.9, the number of triangular elements required by the single-porosity for a single disproportional fracture is about five times of that required by the discrete-fracture model. When the number of fracture increases and real fracture geometry is used

in the single-porosity model, the single-porosity approach becomes impractical because of the amount of nodes and elements involved.

5.5 Case Studies

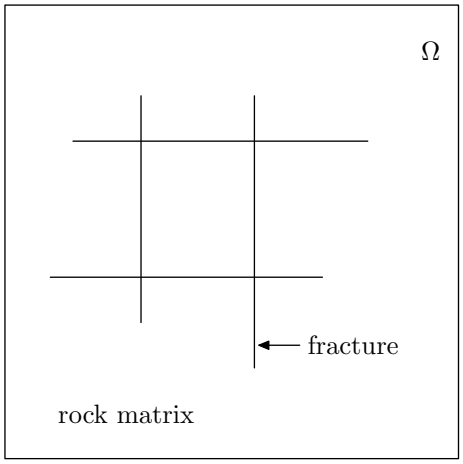

Figure 5.5: A fractured domain Ω.

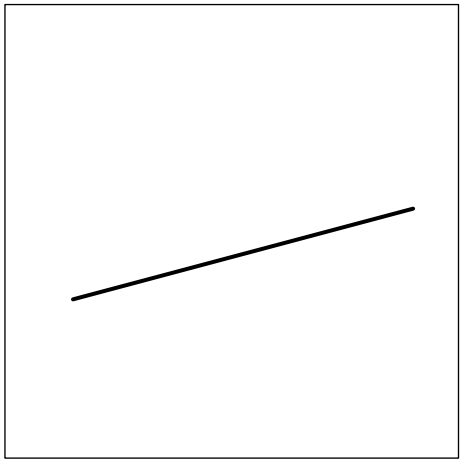

Figure 5.6: The original fractured domain.

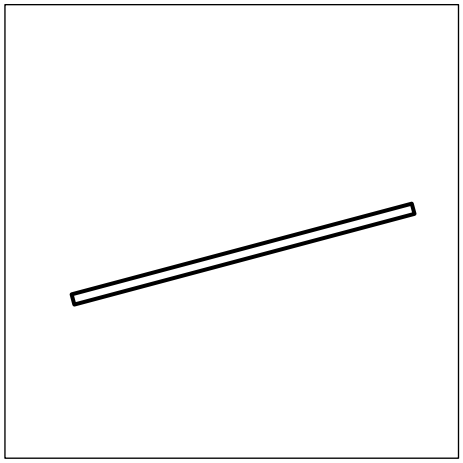

Figure 5.7: The fractured domain after the fracture is replaced by a rectangle; the width of the fracture has been enlarged for visibility.

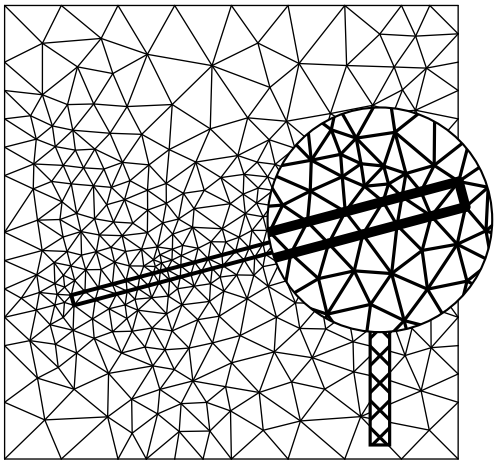

Figure 5.8: Triangular mesh of the domain using the single-porosity model; there are 415 nodes and 780 triangles in this particular mesh.

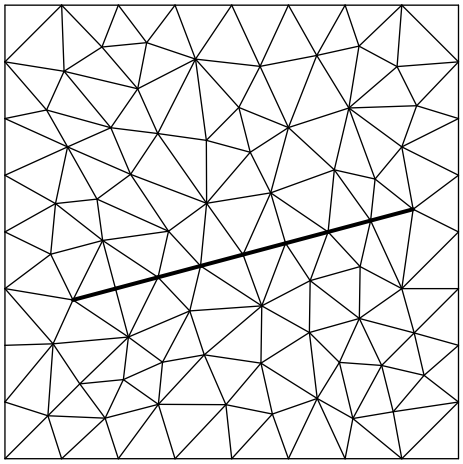

Figure 5.9: Triangular mesh of the domain using the discrete-fracture model; there are 97 nodes and 160 triangles in this particular mesh.

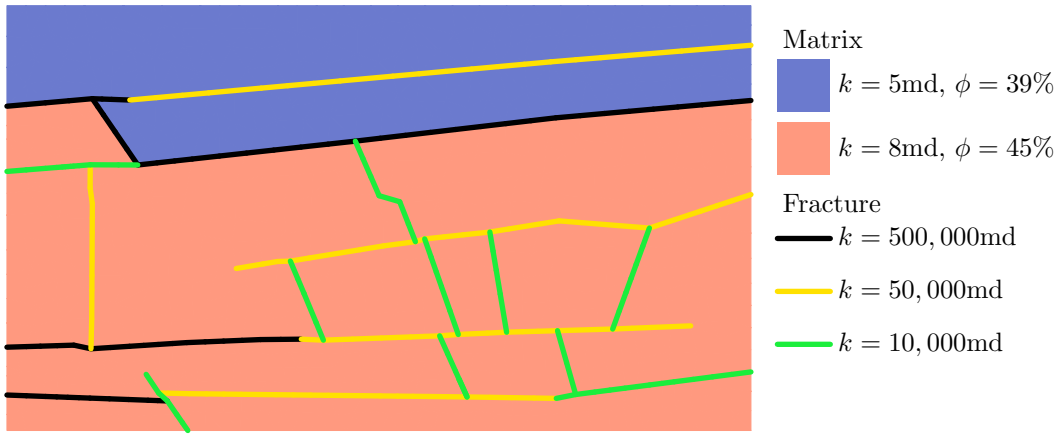

Figure 5.10: The fractured domain. Different matrix and fracture properties are shown in the legend.

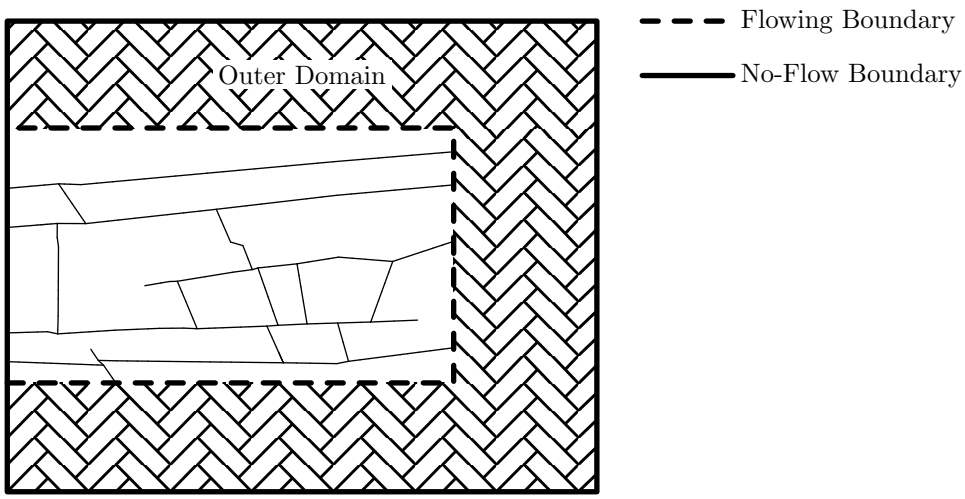

Figure 5.11: The incorporation of the outer domain to simulate the three flowing boundaies.

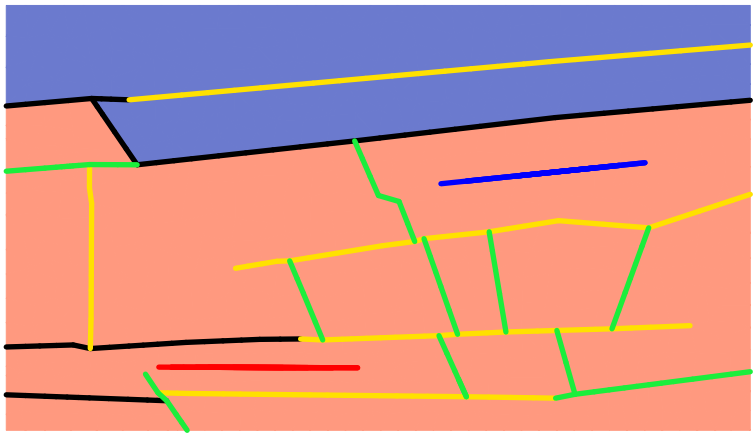

Figure 5.12: The placement of injection and production wells for Case I; the blue and red lines are the horizontal injection and production wells, respectively.

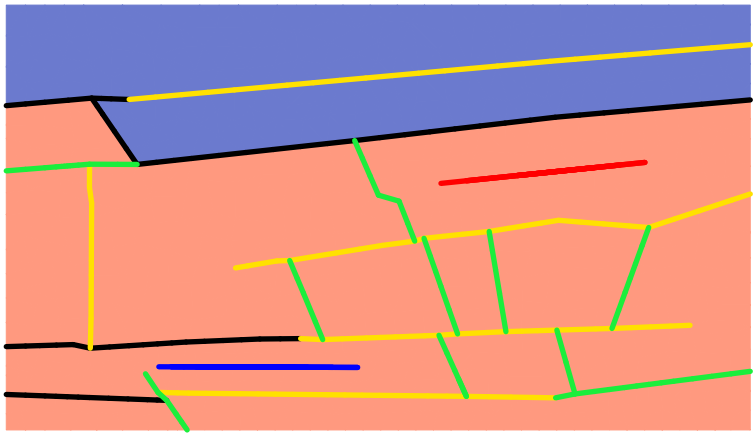

Figure 5.13: The placement of injection and production wells for Case II; the blue and red lines are the horizontal injection and production wells, respectively.

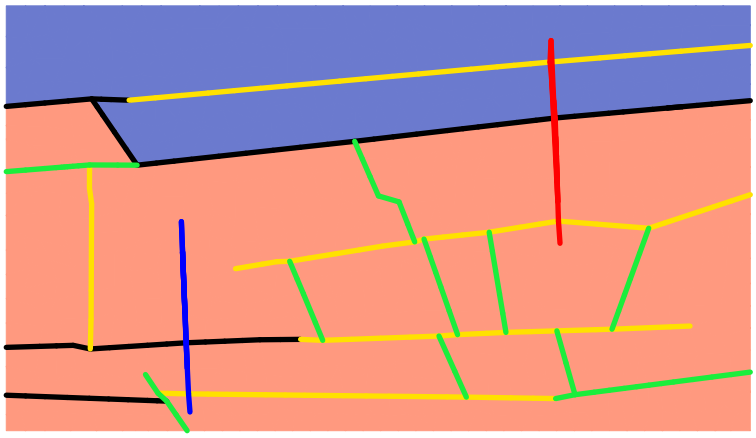

Figure 5.14: The placement of injection and production wells for Case III; the blue and red lines are the horizontal injection and production wells, respectively.

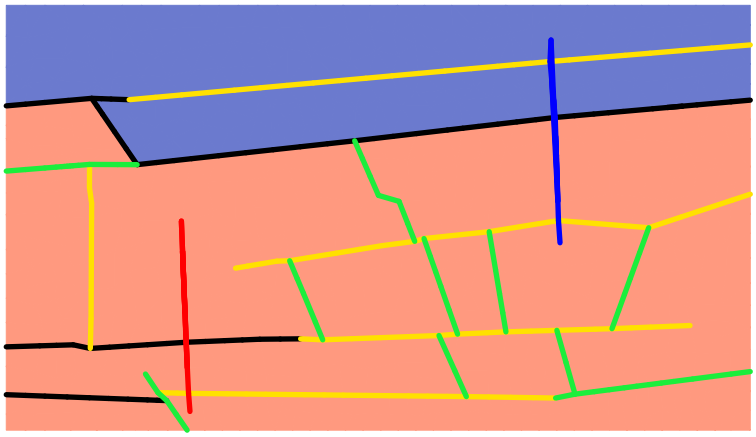

Figure 5.15: The placement of injection and production wells for Case IV; the blue and red lines are the horizontal injection and production wells, respectively.

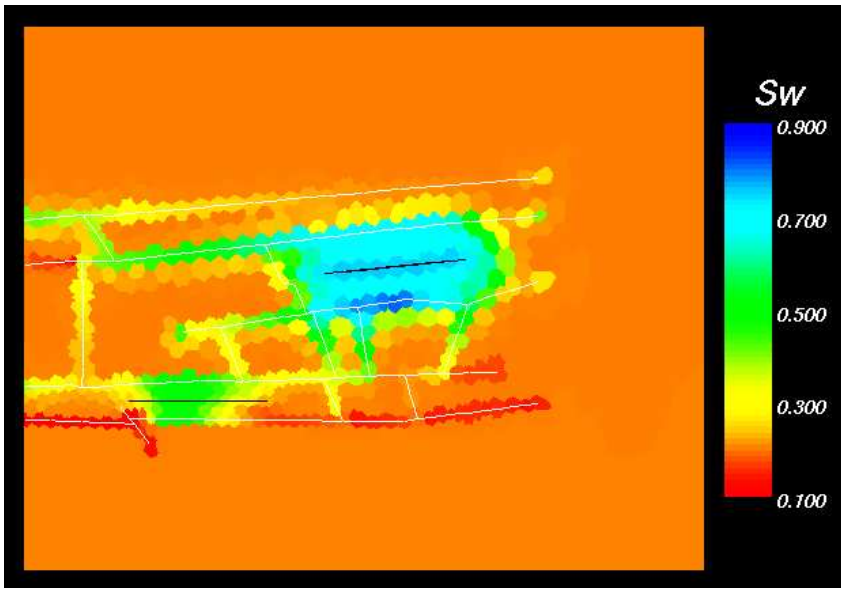

Figure 5.16: Water saturation distribution of Case I at 1000 days.

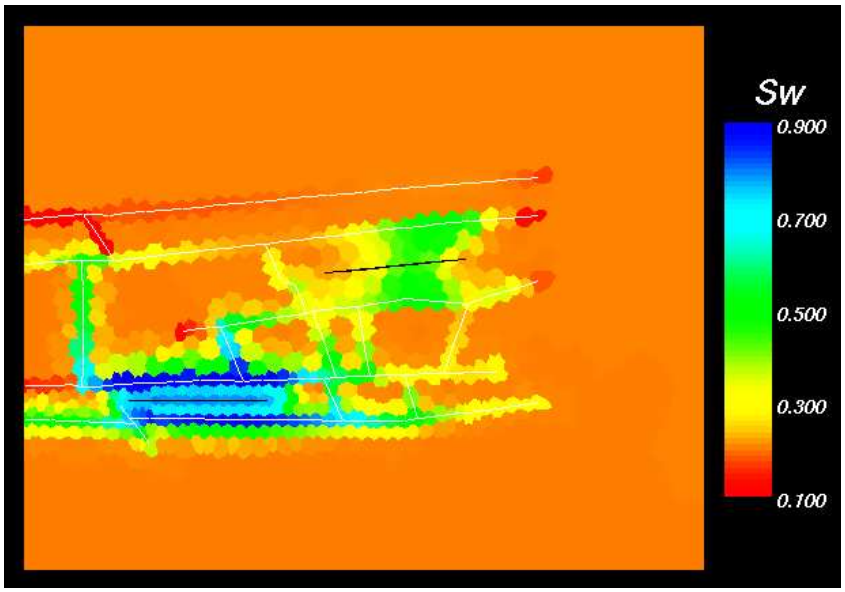

Figure 5.17: Water saturation distribution of Case II at 1000 days.

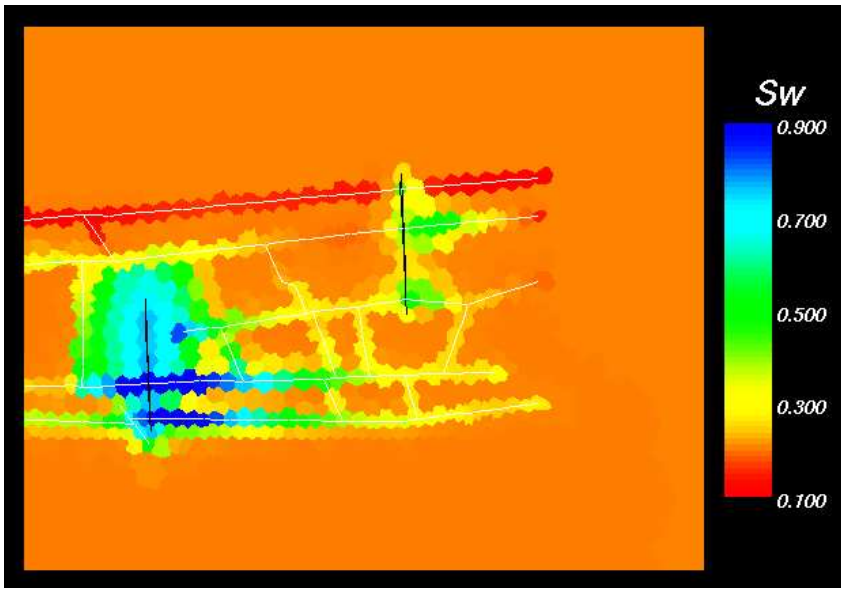

Figure 5.18: Water saturation distribution of Case III at 1000 days.

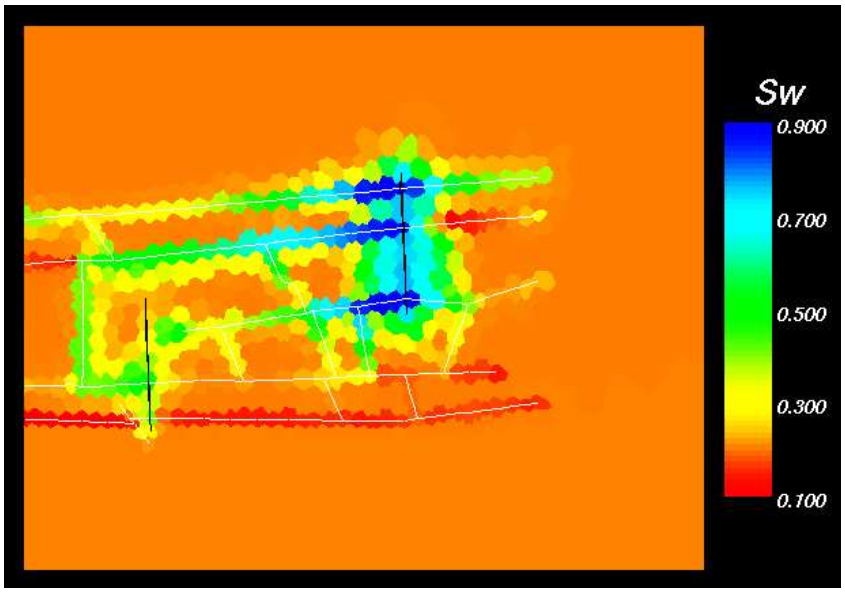

Figure 5.19: Water saturation distribution of Case IV at 1000 days.

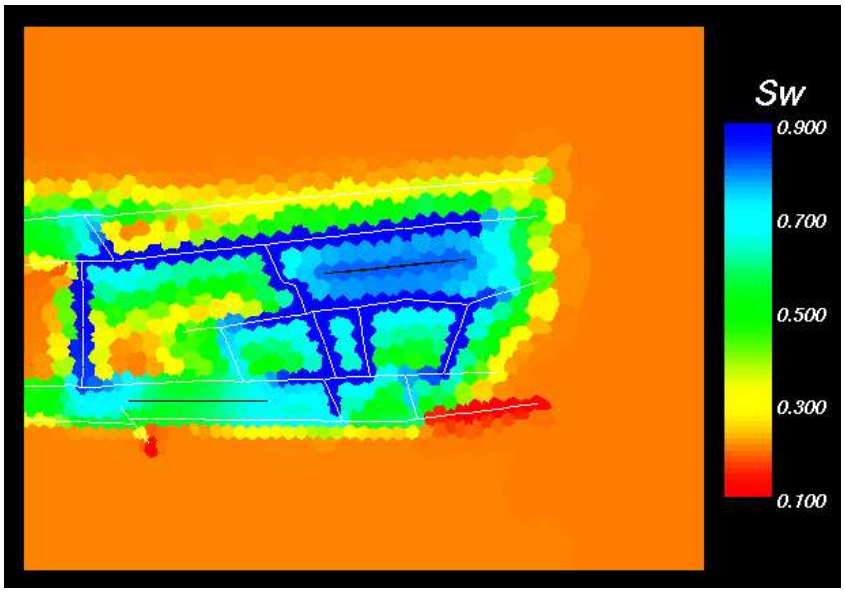

Figure 5.20: Water saturation distribution of Case I at 5000 days.

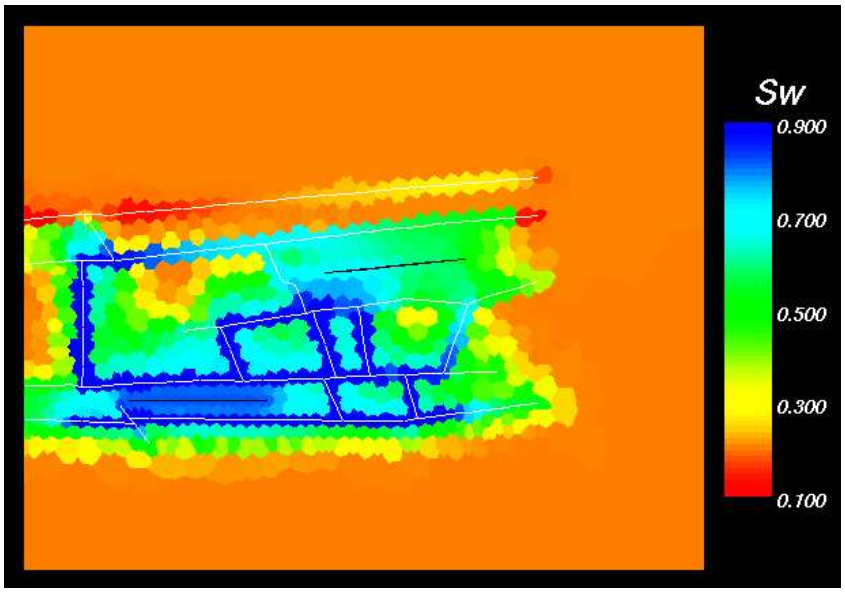

Figure 5.21: Water saturation distribution of Case II at 5000 days.

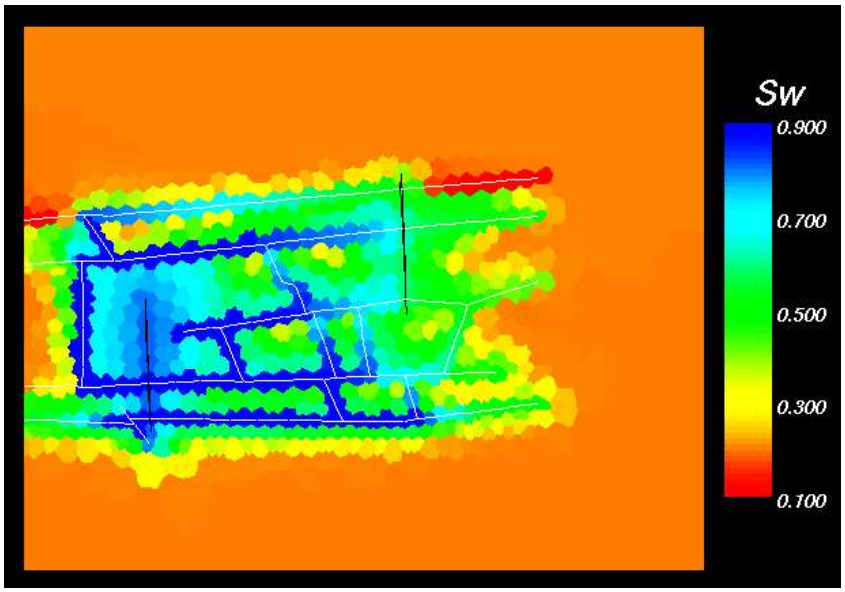

Figure 5.22: Water saturation distribution of Case III at 5000 days.

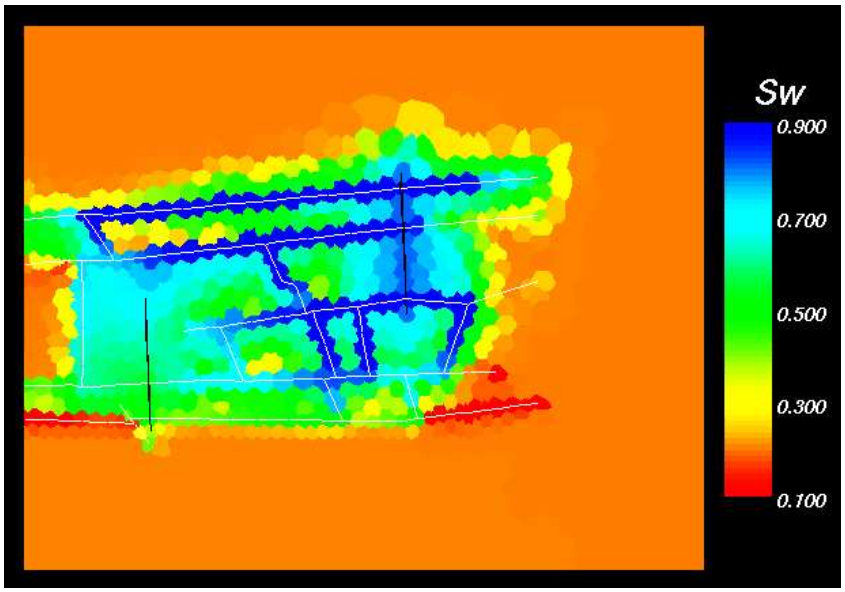

Figure 5.23: Water saturation distribution of Case IV at 5000 days.

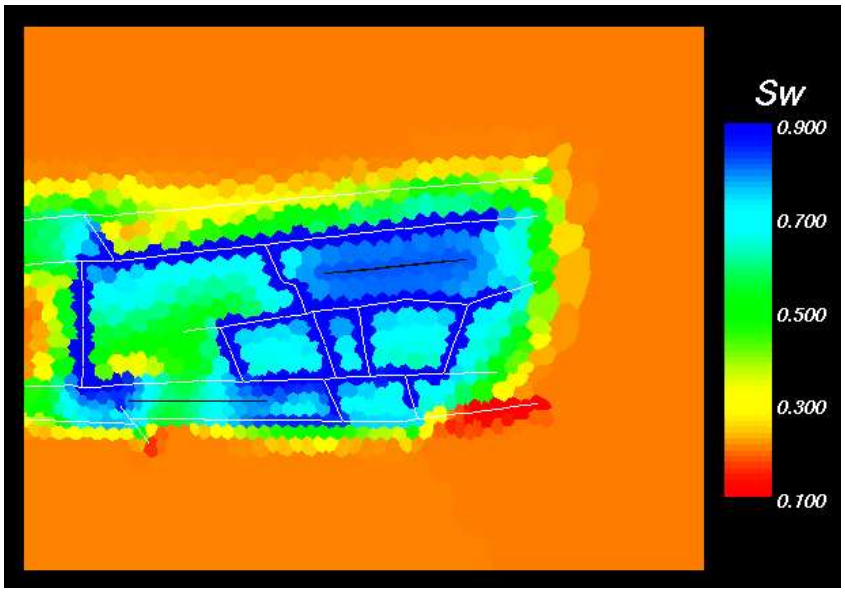

Figure 5.24: Water saturation distribution of Case I at 9000 days.

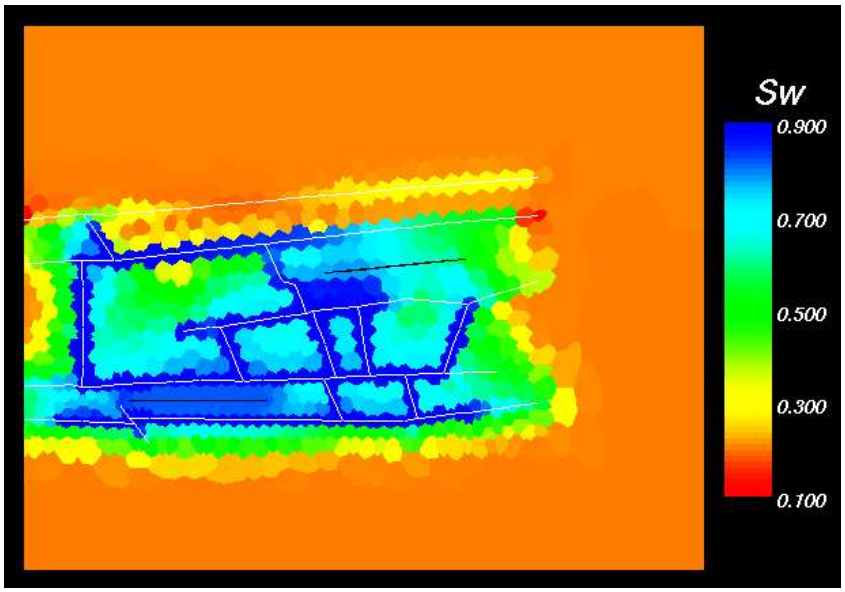

Figure 5.25: Water saturation distribution of Case II at 9000 days.

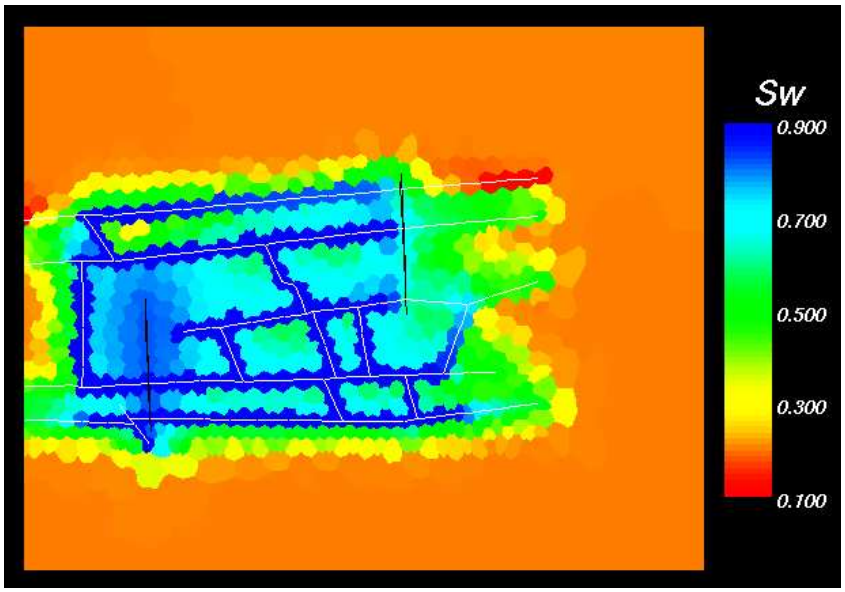

Figure 5.26: Water saturation distribution of Case III at 9000 days.

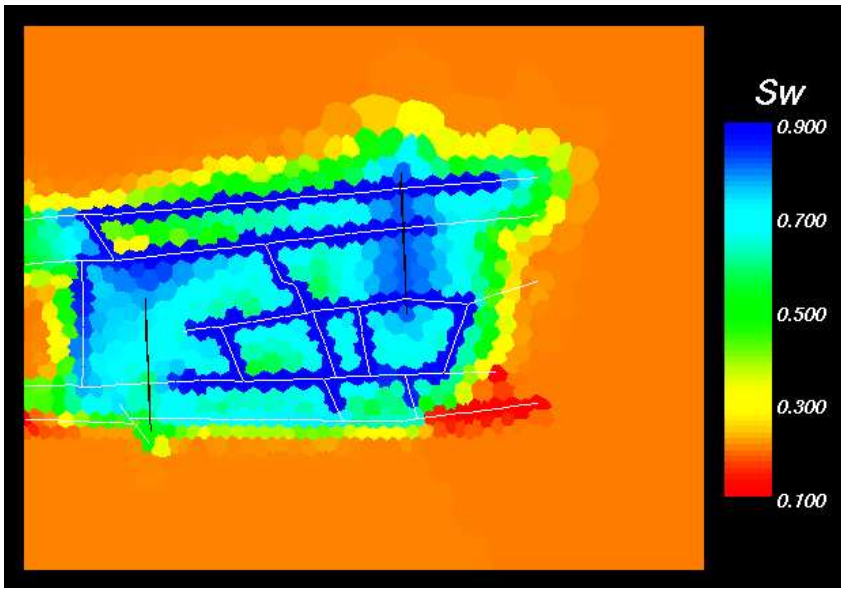

Figure 5.27: Water saturation distribution of Case IV at 9000 days.

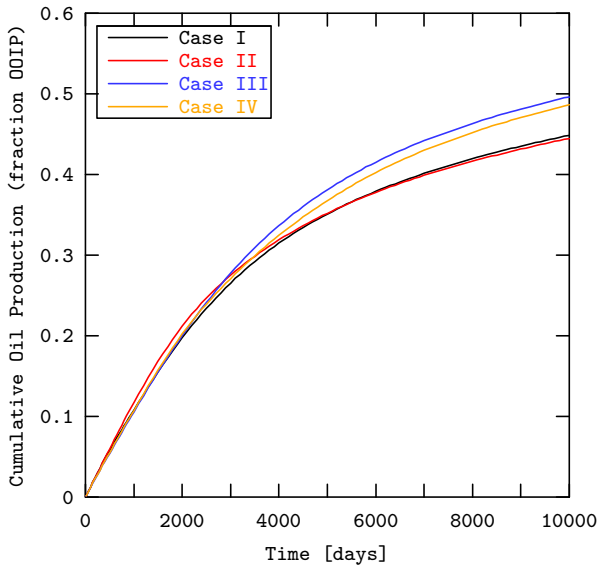

Figure 5.28: Cumulative oil production versus time for the four case studies.

Chapter 6

Examples

6.1 Example 1

In this first example, a complicated two-dimensional domain with several intersecting fractures is simulated. The domain is shown in Figure 6.1. It is difficult to simulate this domain with finite difference representation. In fractured/faulted systems like these, matrix and fracture capillary pressure relationships are critical in establishing recovery patterns. Two different waterflood studies were carried out in this system.

- In the first study, the rock matrix capillary pressure was negative (indicative of an oil-wet rock) while the capillary pressure in the fractures was zero.
- In the second, the rock matrix capillary pressure was positive (indicative of a water-wet system) while the capillary pressure in the fractures was zero.

The water saturations in the domain with negative capillary pressure are shown in Figure 6.2. It is clear that the negative capillary pressure inhibits imbibition, water does not effectively enter the rock matrix from the fractures and oil recovery is poor. The water saturations in the domain with positive capillary pressure in the rock matrix is shown in Figure 6.3. Water floods the matrix efficiently, showing large, invaded blue zones, and recovery is excellent. The simulator makes these types of fundamental studies possible.

6.2 Example 2

In this example, another complex domain, shown in Figure 6.4 is simulated. In a number fields, there are two dominant fracture directions. In this example, both these sets of fractures are shown by white lines. Use of deviating horizontal wells, sometime drilled from the same platform is fairly common, and this field development strategy is simulated. Combinations of these horizontal wells, intersecting fractures and complex boundaries are very difficult, if not impossible to represent

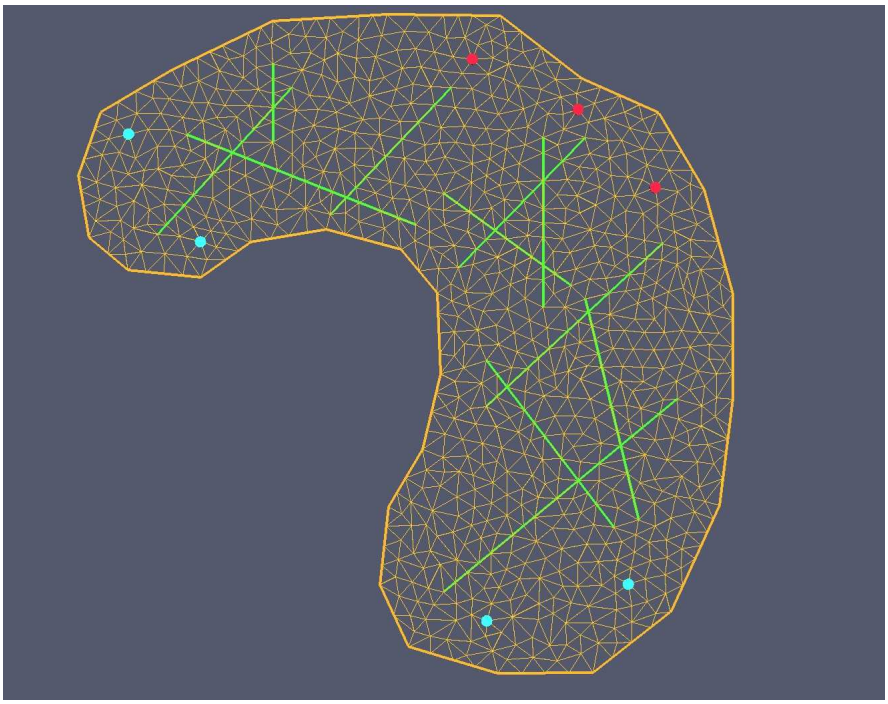

Figure 6.1: Plan view of a domain with intersecting fractures is shown. Water injectors are in blue and producers are shown in red.

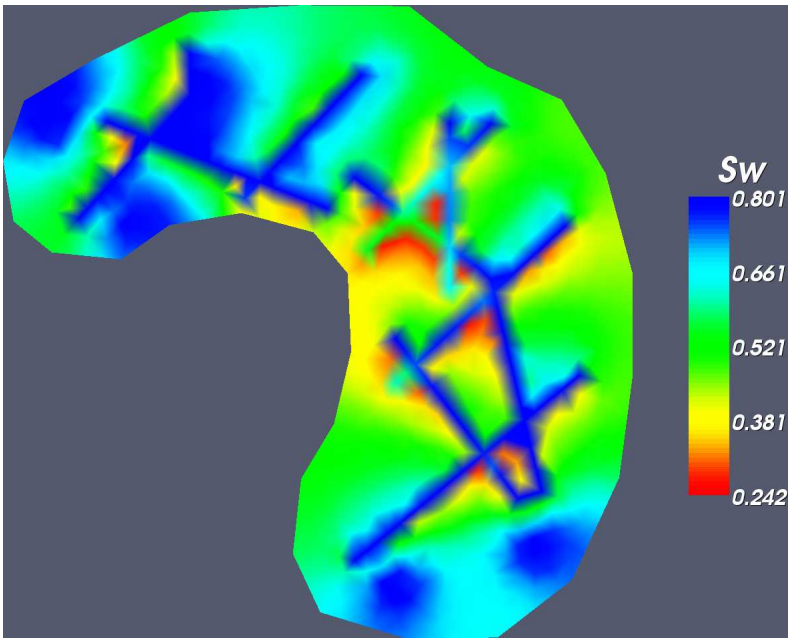

Figure 6.2: Water saturations as a result of a waterflood in a system with negative rock matrix capillary pressure and zero fracture capillary pressure

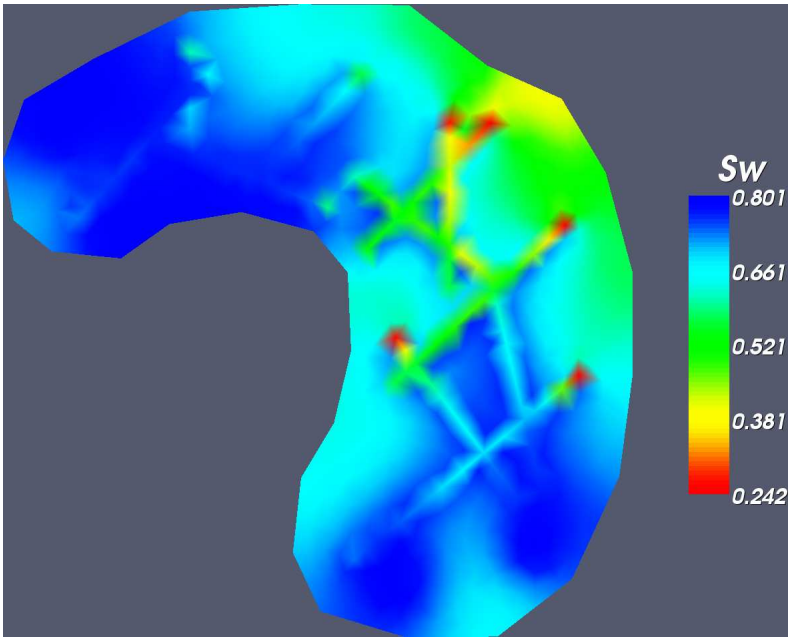

Figure 6.3: Water saturations as a result of a waterflood in a system with positive rock matrix capillary pressure and zero fracture capillary pressure

using existing simulators. Waterflood at early time is shown in Figure 6.5 and at a later time in Figure 6.6. The finite element simulators make it possible to study practical reservoir development in complex fields with fractures/faults and horizontal wells.

6.3 Example 3

Since the success of the waterflood in the Monument Butte Unit, through a U.S. DOE sponsored project with Lomax, Inc. and the University of Utah, over 2000 wells have been drilled in the Greater Monument Butte belt. The target reservoirs are in Greenriver formation. A typical unit consists of wells drilled on a 40-acre spacing. All the wells are hydraulically fracture and are produced for a period ranging from six months to two years. Subsequently, the unit is operated as a five spot waterflood. There can be significant variation in the orientations and lengths of hydraulic fractures; however, in most units the fractures display a dominant direction. The fracture half lengths are of the order of 200 feet. One section, consisting of 16 hydraulically fractured wells is shown in Figure 6.7. The fracture orientation is varied slightly (10-20 degrees) over the dominant northwest-southeast direction and fracture half lengths are varied around 200 feet. This particular simulation had 10,000 elements. The waterflood performance at early and late times is shown in figures 6.8 and 6.9. Simulating these hydraulic fractures with different orientations is very difficult using a structured mesh. The finite element simula-

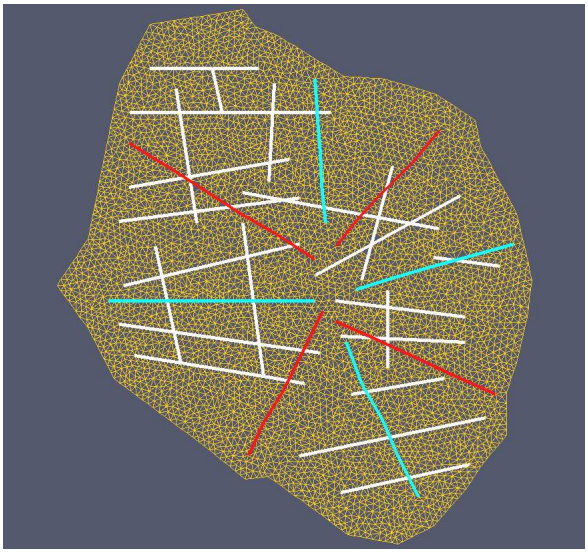

Figure 6.4: Complex domain with two sets of intersecting fractures (white lines), deviating horizontal injectors (blue lines) and horizontal producers (red lines) is shown.

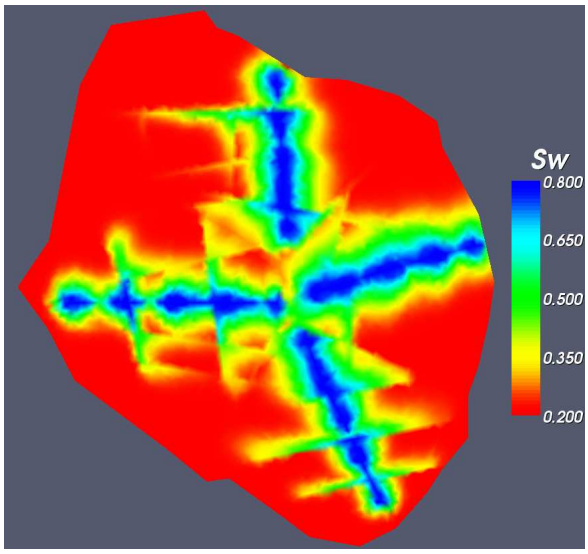

Figure 6.5: Waterflood at an early time in the domain shown in Figure 6.4 is presented.

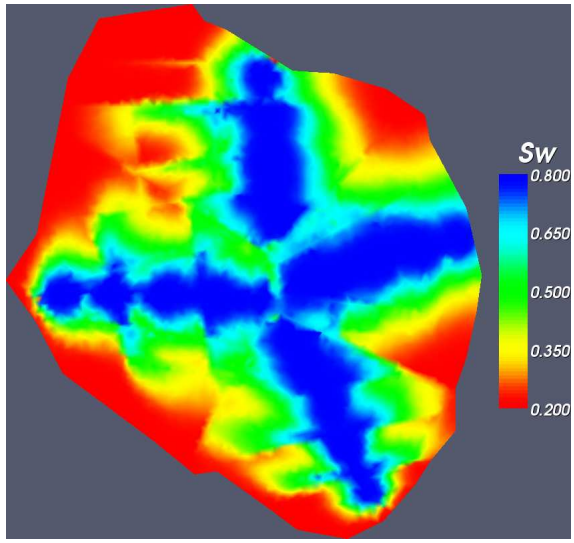

Figure 6.6: Waterflood at a late time in the domain shown in Figure 6.4 is presented.

tors can thus be used to evaluate the effectiveness of hydraulic fracturing, examine early breakthrough phenomena and plan other field management activities.

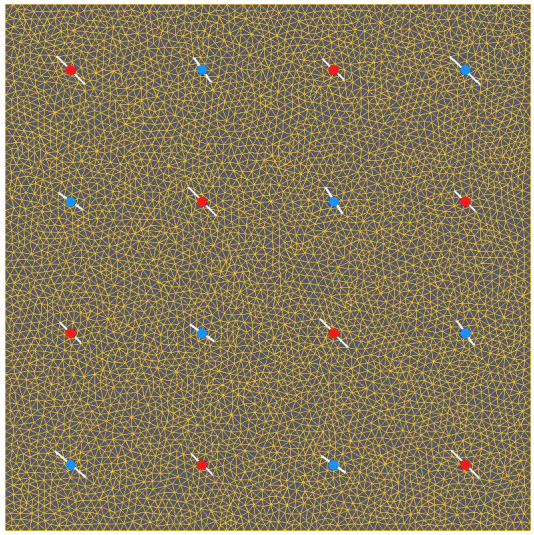

Figure 6.7: A common waterflood unit in the Greater Monument Butte field. The well spacing is 40 acres. All the wells are hydraulically fractured. Slightly different fracture orientations and fracture half lengths of 200 feet area are assumed

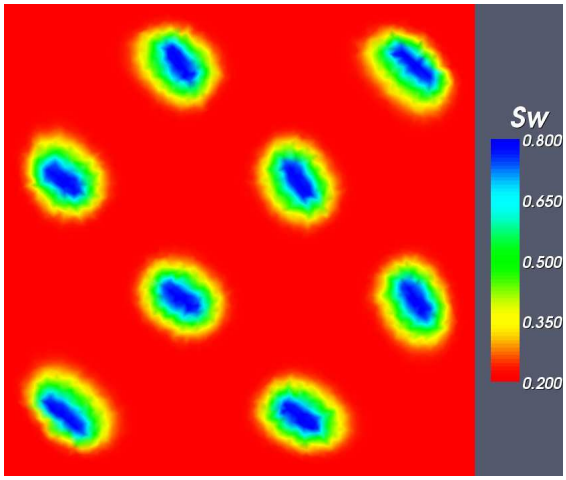

Figure 6.8: Waterflood at an early time in the domain shown in Figure 6.7 is presented.

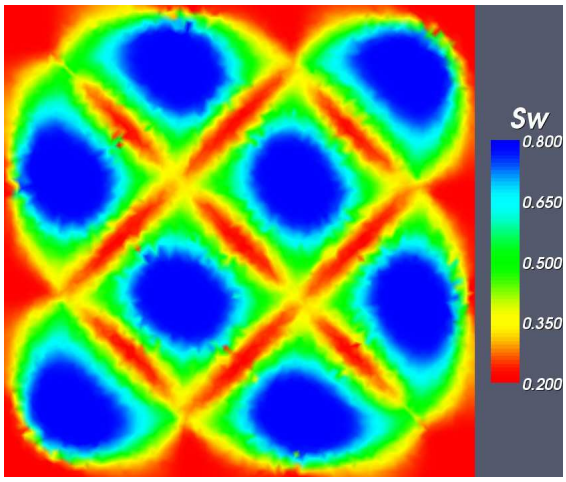

Figure 6.9: Waterflood at a a late time in the domain shown in Figure 6.7 is presented.

Chapter 7

Example of Three-Dimensional Simulation

The three-dimensional simulation was performed by constructing the three-dimensional extension of the two-dimensional domain examined in Example 1 of Chapter 5. The domain was irregular with the total dimensions in the x and in the y directions being about 600 feet. The thickness of the domain was 50 feet. Four intersecting fractures were placed in the domain. These were not all orthogonal - the dip angles varied between 70 and 90 degrees. Tetrahedral mesh was created with this system. The domain and the mesh are shown in figures 7.1 and 7.2.

A study similar to the one reported for the two-dimensional domain with positive and negative capillary pressures was conducted. The first set of pictures are with negative capillary pressures in the rock matrix. The injection and production rates were 80 barrels per day. The three-dimensional plot of water saturations in the system after 181 days of continuous injection is shown in Figure 7.3.

More useful information is obtained when cross-sections within the three-dimensional domain are observed. Four Y-Z sections at different x values are shown in Figure7.4. The impact of fractures and the gravity-driven water flow are clearly captured in this figure.

The effect of gravity on the water flood can be observed by studying the X-Y sections at different z values. Two such cross sections are shown in figures 7.5 and 7.6. The first cross section in Figure 7.5 is at a z value of 10 feet and the second cross section in Figure 7.6 is at a z value of 40 feet. As expected, the water saturations are uniformly high in the cross section at z equal to 40 feet. These figures also illustrate the fact that imbibition is poor, leading to poor rock matrix sweep.

At later times, basically the same trends are observed. Figure 7.7 shows Y-Z cross sections of water saturations after 1003 days of injection and figure 7.8 shows the plan view (X-Y cross section) at the same time.

The picture is different when positive capillary pressure values are used for the rock matrix. The three-dimensional view of water saturations after 181 days of injection is shown in figure 7.9. The Y-Z cross sections at the same x values as

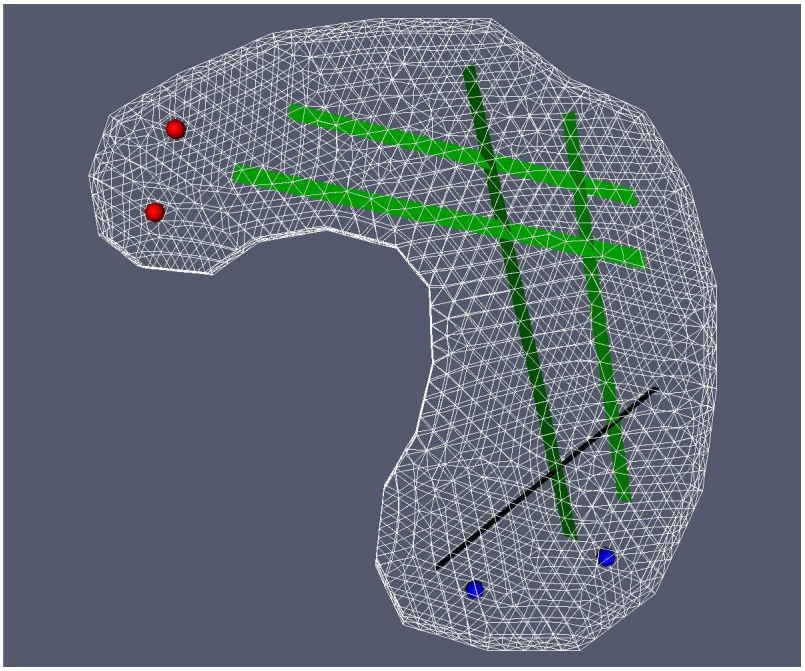

Figure 7.1: The three-dimensional domain showing the fractures and the tetrahedral mesh

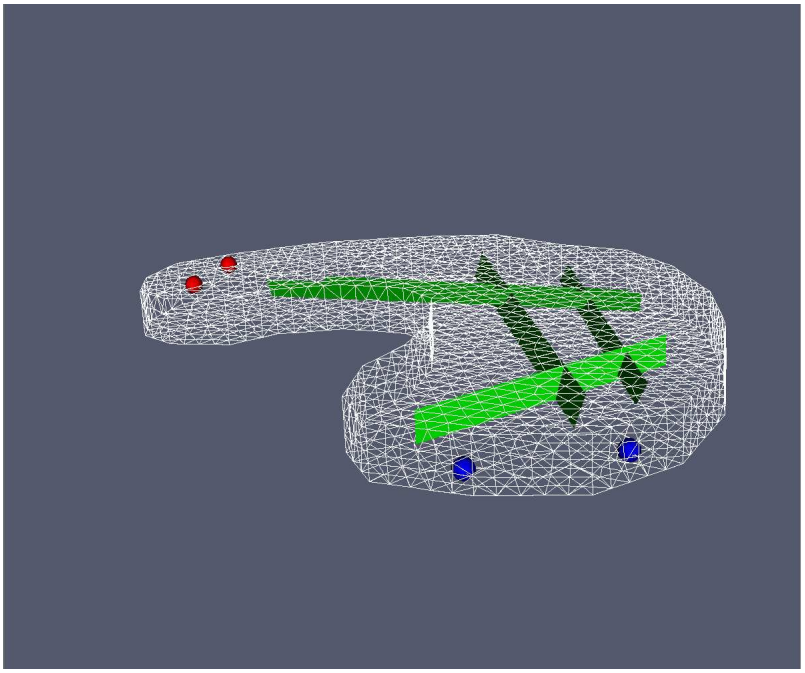

Figure 7.2: Another view of the three-dimensional domain with fractures and the mesh created

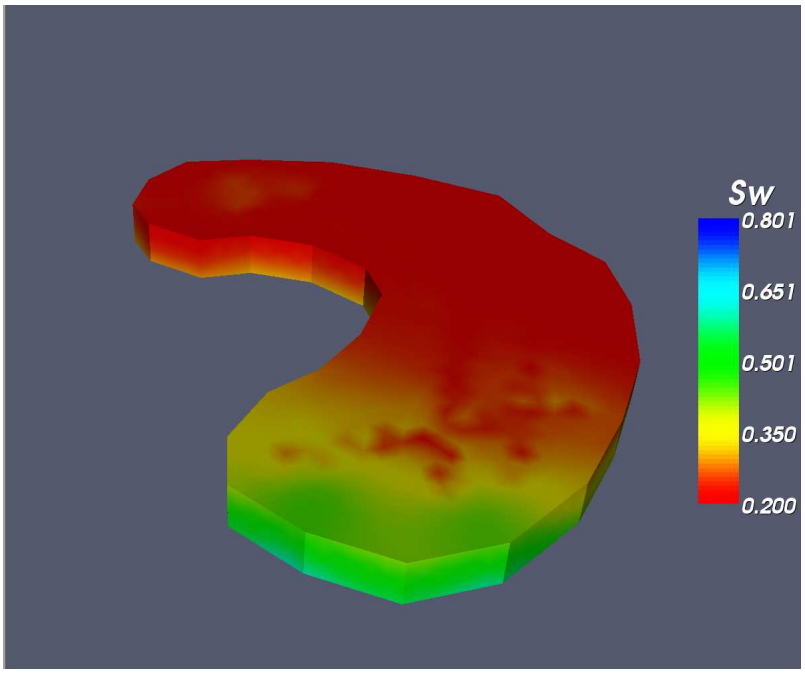

Figure 7.3: Water saturations in the three-dimensional domain with fractures after 181 days of injection

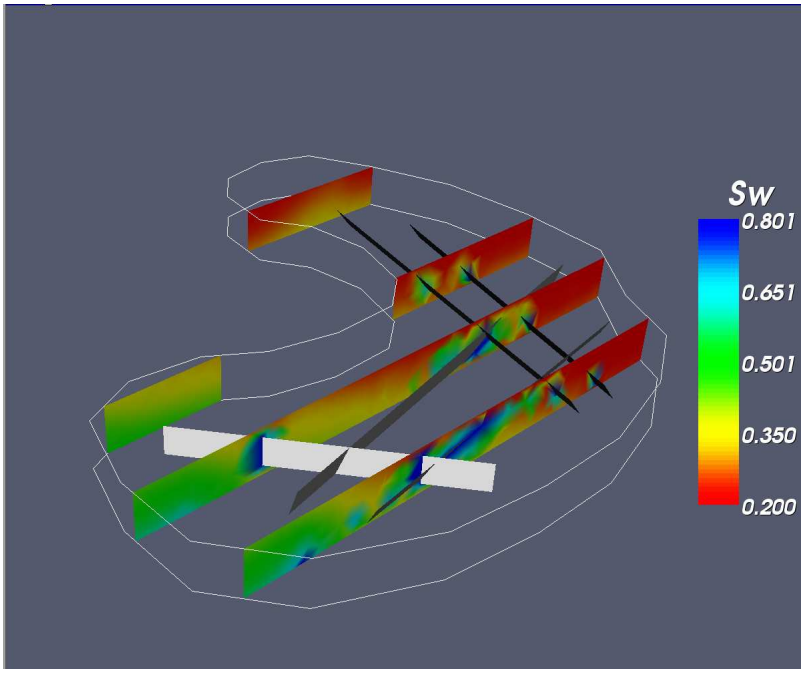

Figure 7.4: Water saturations at 181 days. Y-Z cross-sections at x values of 174, 380, 488 and 593 feet are shown.

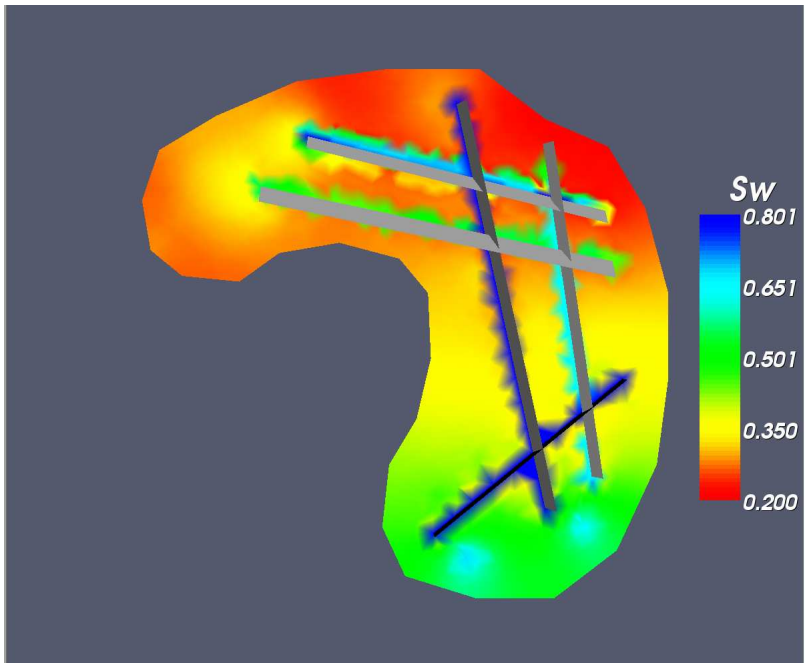

Figure 7.5: A plan view (X-Y section) at z=10 feet. Water saturations are after 181 days of injection are shown.

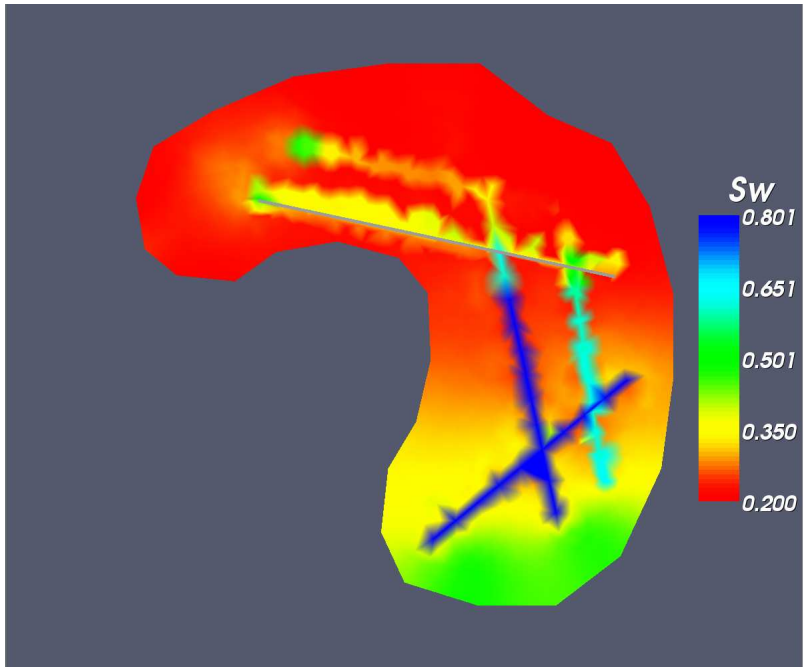

Figure 7.6: A plan view (X-Y section) at z=40 feet. Water saturations after 181 days of injection are shown.

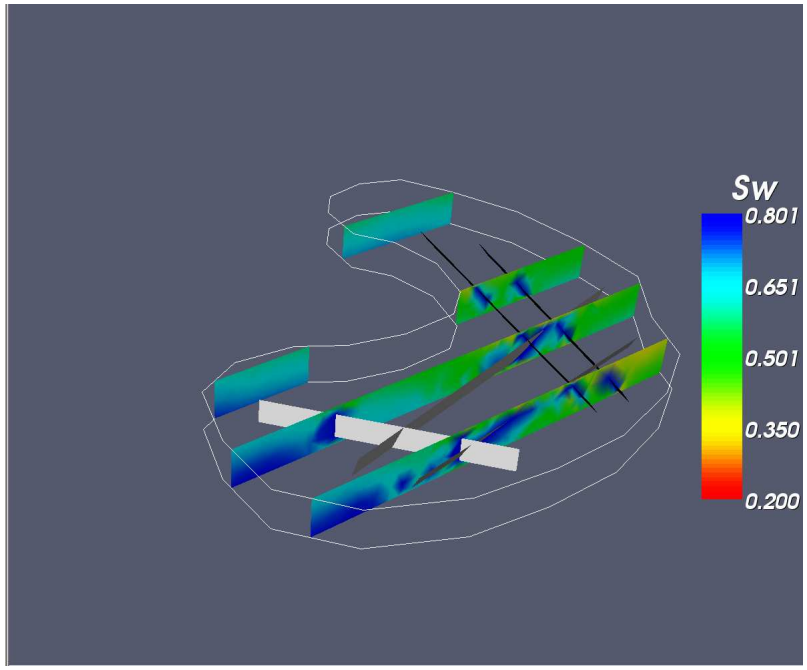

Figure 7.7: Water saturations at 1003 days. Y-Z cross-sections at x values of 174, 380, 488 and 593 feet are shown.

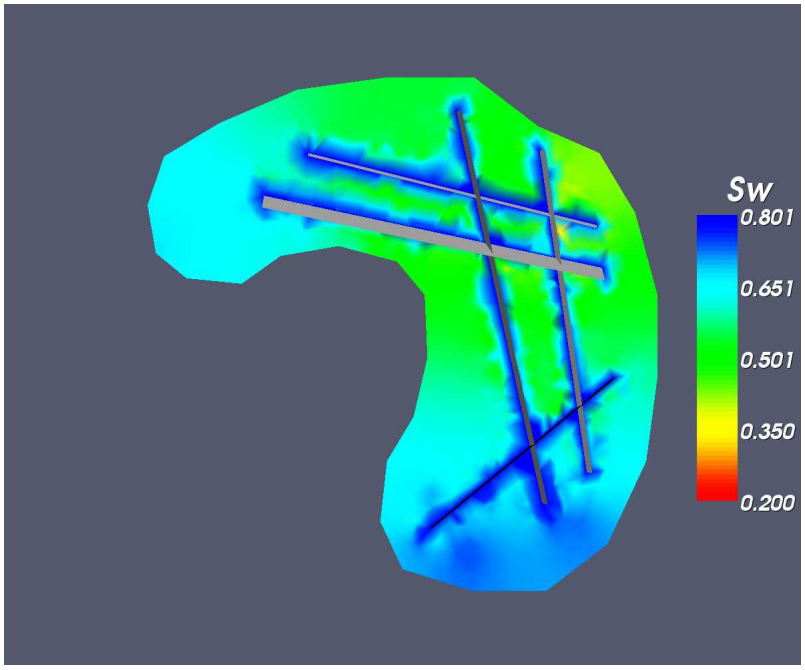

Figure 7.8: A plan view (X-Y section) at z=25 feet. Water saturations are after 1003 days of injection are shown.

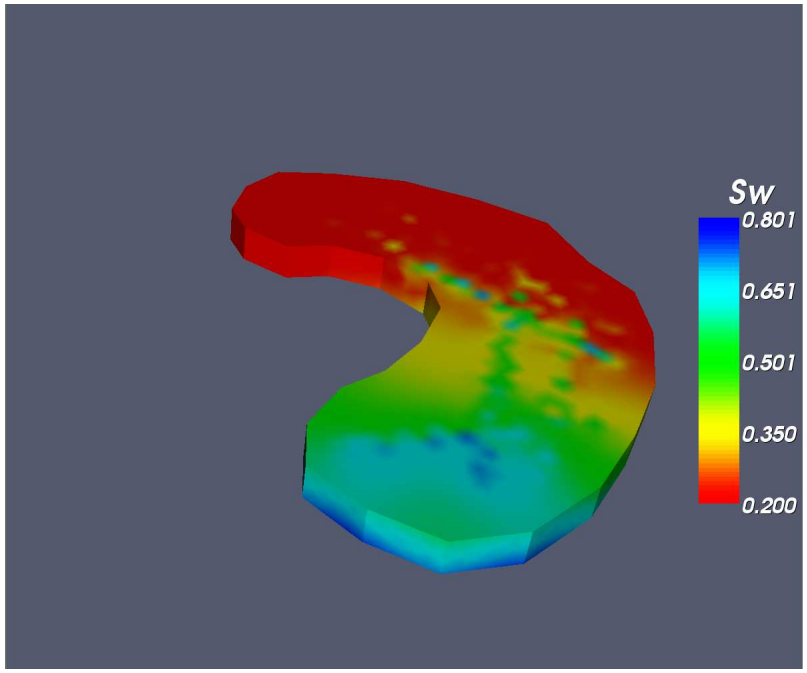

Figure 7.9: Water saturations in the three-dimensional domain with fractures after 181 days of injection; capillary pressures in the rock matrix are positive

before are shown in figure 7.10. The effectiveness of the sweep due to imbibition is best illustrated with the plan view of water saturation at z equal of 25 feet after 1003 days of injection (shown in figure 7.11.

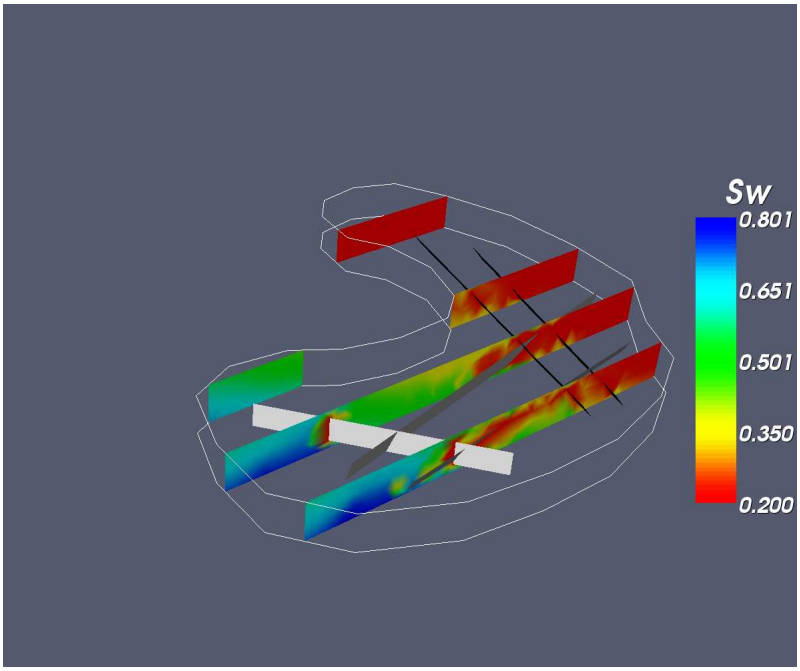

Figure 7.10: Water saturations for positive capillary pressure in the rock matrix at 181 days. Y-Z cross-sections at x values of 174, 380, 488 and 593 feet are shown.

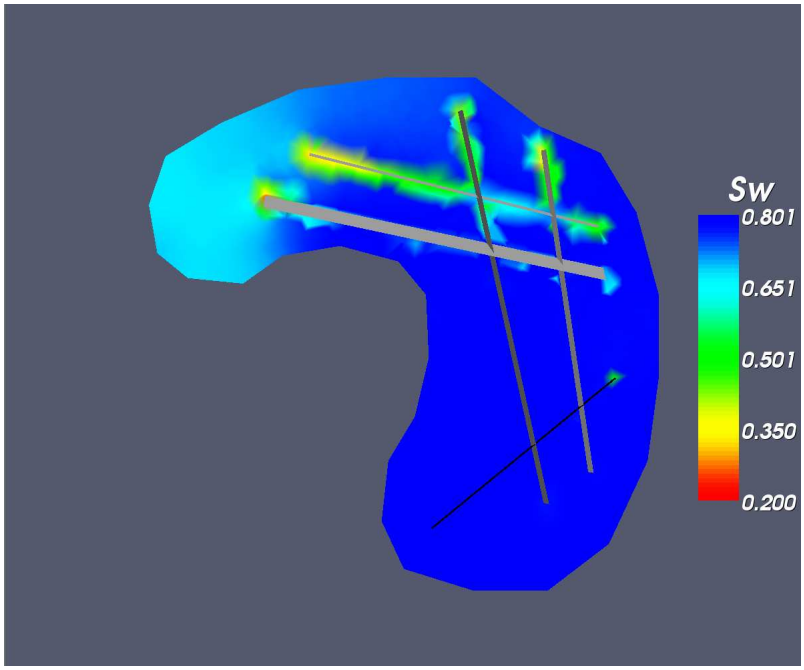

Figure 7.11: A plan view (X-Y section) at z=25 feet. Water saturations are after 1003 days of injection and the rock matrix capillary pressure is positive

Chapter 8

OPTIMIZATION TECHNIQUE AND APPLICATIONS

The first part of this chapter outlines the basics of optimization and a general overview of some of the techniques best suited to solve nonlinear optimization problems. In the second part this chapter, the algorithm used for optimization is outlined. The third part demonstrates the application of the algorithm to reservoir examples.

8.1 Description of the Optimization Problem

Optimization is the act of obtaining the best result under given circumstances using efficient quantitative methods. It is important to use established mathematical techniques rather than implement ad hoc procedures to perform optimization. A typical optimization problem is described by a system or process and a measure or criterion that is used to judge the performance of the system, subject to the imposed constraints. The system or the model as it is sometimes called is described by a set of variables known as the states. The entire process is driven by a set of inputs commonly known as control variables.

Mathematically, an optimization problem can be written as,

$$\text{Minimize (or Maximize): } J(\mathbf{u}) \tag{8.1}$$

$$\text{Subject to: } g(\mathbf{x}) = 0 \tag{8.2}$$

$$x_l \leq x_i \leq x_u \quad i = 1, 2, \ldots, n \tag{8.3}$$

$$u_l \leq u_j \leq u_u \quad j = 1, 2, \ldots, m \tag{8.4}$$

The optimization problem is described by the nature of the cost function (performance criterion), $J(\mathbf{u})$ in Equation (8.1) and the process defined by Equation (8.2). The state variables of the system are denoted by $\mathbf{x}$ and $\mathbf{u}$ represents the set of control variables. If the cost function and the process are both linear, the problem is termed as a linear problem. If any one of the function or the system

or both are nonlinear, the optimization problem is a nonlinear one. Optimization in reservoir simulation is an example of nonlinear optimization where the state variables are computed by solving nonlinear equations. The reservoir model is dynamic in nature which makes this a dynamic optimization problem. The cost function, even a simple profit function in reservoir engineering, as described later in this chapter, is nonlinear in nature.

8.2 Overview of Nonlinear Optimization Techniques and Applications

Several researchers have successfully implemented formal optimization methods to solve various reservoir engineering problems. This section describes some of the work done in this area of dynamic optimization with reasonable mathematical explanations wherever possible. Many of these methods were considered before deciding on the method used in this work.

8.2.1 Optimal Control Theory

The goal of an optimal control problem is to determine a control policy or a strategy that will maximize or minimize a performance criterion or an objective functional subject to constraints that describe the dynamics of the system. The fundamental theories of this problem are the calculus of variations and Pontryagin's Maximum Principle. The calculus of variations approach is based on the fundamental theorem of variational calculus [32]. The theorem is applied to problems with no constraints on states and controls. The theorem is coupled with Pontryagin's Maximum Principle [33] to allow constraints on state and control variables. Ramirez *et al.* [34] have presented the theory of optimal control of distributed-parameter systems where they applied the theory of calculus of variations coupled with Pontryagin's minimum principle to determine the best surfactant injection policy for tertiary recovery of oil. Fathi and Ramirez [35] applied this theory to maximize the cost of oil recovery while minimizing the cost of surfactant injected. The amount of chemical injected was used as the control variable. The control was updated using a steepest-descent gradient method. This work was extended to determine optimal operating strategies for steamflooding [36] and carbon dioxide miscible-flooding [37] enhanced oil recovery processes processes. Sudaryanto and Yorstos [38] addressed the issue of control of displacement fronts of fluids flowing in porous media. They used optimal control theory to control the injection rates at different point sources. Brouwer and Jansen [39] applied the optimal control theory to optimize the valve settings in smart wells for waterflooding of heterogeneous reservoirs. A gradient based control technique was used to solve the problem. Maximization of the Net Present Value (NPV) was identified as the objective of the problem.

8.2.2 Dynamic Programming and its Variations

The dynamic programming (DP) method is a stage-wise procedure which enables one to construct the optimal solution to a multistage decision problem. This technique was developed by Bellman [40] in mid-twentieth century. The technique decomposes a multistage decision problem as a sequence of single-stage decision problems. Thus an $n-$variable problem is broken down into and solved as a sequence of n single-variable problems. The decomposition is carried out in such a manner that the optimal solution of the original problem is obtained from the optimal solutions of the n single-variable problems, though it should be noted that the technique used for optimization of the decomposed problem could be anything ranging from simple differential calculus to a complex nonlinear programming technique and is irrelevant in the context of the optimal solution. The process of sub-optimization can be described using Bellman's principle of optimality [41]. The principle of optimality states that,

An optimal policy has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

The main limitation of dynamic programming is the *"curse of dimensionality"*. At each stage, the state variable is defined over a large domain. This means that the values of the optimal value function and control need to be stored for all values of the state variables. This storage cost is very expensive. Differential Dynamic Programming (DDP) was developed to overcome this difficulty. This method starts with a nominal control policy, $\bar{u}_j$, and compares the resulting nominal state trajectory, $\bar{x}_i$, with the neighboring trajectories and selects the neighboring trajectory that yields an optimal cost reduction. There are several good references [42, 43, 44, 45] that compare the DDP to the DP and also list the advantages of DDP over DP. Though Jacobson and Mayne [46] were the pioneers of DDP, several mathematicians came up with improvements to the algorithm. The DDP algorithm developed by Yakowitz and Rutherford [47] was modified by Chang *et al.* [48] to obtain optimal rates of pumping of groundwater to reduce the contaminants in the sub-surface. The only modification made by them was the omission of the second order derivatives of the constraint functions from the algorithm for computational ease. This is equivalent to a Linear Quadratic Regulator (LQR) problem, an established control theory. Second order derivatives of the objective function with respect to the control variables and the states are required to be computed in DDP in addition to the derivatives of the constraint function with respect to the states and controls. Although Chang [49] has used analytical expressions for these derivatives it is necessary to compute numerical derivatives for many applications and this becomes a computational burden.

8.2.3 Gradient Based Methods

The gradient of a function, $f(\mathbf{x})$ is the vector at the point $\mathbf{x}$ that gives the direction of the greatest rate of increase in $f(\mathbf{x})$. The search direction is the gradient for a maximization problem and is the negative of the gradient for a minimization problem. The minimization problem of this type is known as the steepest descent problem. The search direction, s_k, for the kth iteration of the steepest descent method can be written as,

$$s_k = -\nabla f(\mathbf{x}_k) \tag{8.5}$$

The kth iteration for a steepest descent algorithm is given by,

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + \Delta \mathbf{x}_k && (8.6) \\ &= \mathbf{x}_k + \alpha_k s_k && (8.7) \\ &= \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) && (8.8) \end{aligned}$$

where, α_k is the scalar that determines the step length in the direction s_k and is determined by line search techniques [50]. The iterations are stopped when the value of $f(\mathbf{x}_k)$ converges based on a specified tolerance. The main difficulty with the steepest descent approach is its large sensitivity to the scaling of $f(\mathbf{x}_k)$. This, in turn, leads to a slow convergence and often oscillation in the $\mathbf{x}$ space.

The conjugate gradient technique is a major improvement over the steepest descent method. It combines current information about the gradient with that of the gradients from the previous iterations to obtain the new search direction. For the kth iteration,

$$s_{k+1} = -\nabla f(\mathbf{x}_{k+1}) + s_k \frac{\nabla^T f(\mathbf{x}_{k+1}) \nabla f(\mathbf{x}_{k+1})}{\nabla^T f(\mathbf{x}_k) \nabla f(\mathbf{x}_k)} \tag{8.9}$$

The starting search direction, s_0, is computed as,

$$s_0 = -\nabla f(\mathbf{x}_0) \tag{8.10}$$

and the new update for $\mathbf{x}$ is obtained according to Equation (8.7).

The main advantage of this method is that requires only a small amount of information to be stored at each iteration. It can be noticed from Equation (8.9) that the new search information is expressed as a linear combination of the current gradient and the previous search direction. Different linear combinations have been developed by researchers and two of most commonly used ones are due to Fletcher and Reeves and Polak and Ribiere. There are many good references [50, 51, 52, 53] for the conjugate gradient methods and their formulations and applications. Yeten *et al.* [54] used the conjugate gradient method for the optimization of smart well control. The optimization method was linked with a commercial reservoir simulator. They divided the simulation period into several stages and optimized the valve settings for each period. The work presented here follows a similar approach to that followed by Yeten *et al.* but uses a different

optimization technique which takes into account the second order information of the cost function. The important difference between the algorithm developed here and the one suggested by Yeten *et al.* is that the water injection rates are optimized for the different stages of the simulation period simultaneously rather than optimizing them for each period separately. This approach identifies the presence of multiple maxima which is something that the approach of Yeten *et al.* fails to capture.

8.2.4 Newton's Method and its Improvements

Newton's method employs the use of the second-order approximation of $f(\mathbf{x})$ at $\mathbf{x}_k$. It is therefore possible to account for the curvature of $f(\mathbf{x})$ at $\mathbf{x}_k$ and identify better search directions than can be obtained using the gradient methods. In the Newton's method, the extremum of a function, $f(\mathbf{x})$, is found from the Taylor series expansion of the function to its quadratic term [50, 55]. Quadratic approximation of $f(\mathbf{x})$ is given by,

$$f(\mathbf{x}) \approx f(\mathbf{x}_k) + \nabla^T f(\mathbf{x}_k) \triangle \mathbf{x}_k + \frac{1}{2} (\triangle \mathbf{x}_k)^T \mathbf{H}(\mathbf{x}_k) \triangle \mathbf{x}_k \tag{8.11}$$

Minimum of $f(\mathbf{x})$ in the quadratic approximation is obtained by differentiating the approximation with respect to the components of $\triangle \mathbf{x}$, yielding,

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}_k) + \mathbf{H}(\mathbf{x}_k) \triangle \mathbf{x}_k = 0 \tag{8.12}$$

or,

$$\mathbf{x}_{k+1} - \mathbf{x}_k = \triangle \mathbf{x}_k = -[\mathbf{H}(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k) \tag{8.13}$$

where, $[\mathbf{H}(\mathbf{x}_k)]$ is the Hessian matrix of $f(\mathbf{x})$. Usually, a line search parameter, α_k is used for progressing quickly toward the minimum. As a result, Equation (8.13) can be written as,

$$\triangle \mathbf{x}_k = -\alpha_k [\mathbf{H}(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k) \tag{8.14}$$

The search direction for minimization is,

$$s_k = -[\mathbf{H}(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k) \tag{8.15}$$

As evident from the mathematical explanation above, the Newton's method searches in the direction of the gradient. Convergence is an issue far away from the local solution. To improve the convergence behavior of the method, a step size is chosen in the direction of the descent. This is the idea of a line-search algorithm. Dennis and Schnabel [55] have explained in detail about computing an acceptable step size for the line-search technique. In a line-search algorithm, the step direction is retained while the step length is reduced. Another method that ensures the convergence of the Newton's method is the trust region method [55]. In this method the shorter step length is first determined and then the quadratic

model of the function is used to determine the step direction. A key limitation of the Newton's method is that the positive-definiteness of the Hessian is not always assured. Also, it requires the computation of second order derivatives, which may not be practical to obtain using numerical techniques. Several methods have been suggested to ensure the positive-definiteness of the Hessian. The most common method in this category is the quasi-Newton method that replaces $H(x_k)$ by a positive-definite approximation, $\tilde{H}_k$. $\tilde{H}_k$ is usually initialized as any positive-definite symmetric matrix (usually the identity matrix or any diagonal matrix) and is updated after every line search using $\triangle x$ and the change in $\nabla f(x)$ over the two most recent points. One of the most often encountered updating formula is the BFGS (after Broyden, Fletcher, Goldfarb and Shanno) update [50, 55]. The details of this method will be discussed in Section 8.3.1.

Nishikiori *et al.* [56] applied a quasi-Newton method to find optimum gas injection rates for a group of continuous gas lift wells to maximize the total oil production rate constrained by a total gas volume available for injection. The first order derivatives were computed using a central difference approximation. The Hessian was approximated using the DFP (Davidon, Fletcher and Powell) updating formula [57]. Dutta-Roy and Kattapuram [58] presented a new approach to the simulation and optimization of the gas-lift allocation problem with a Sequential Quadratic Programming (SQP) approach for the nonlinear constrained optimization. The total oil production at the separation and processing facilities was maximized. The primary constraint considered was the availability of injection gas. Sequential Quadratic Programming (SQP) is a very popular method to solve nonlinear programming problems [57]. It solves a nonlinear programming problem using a sequence of quadratic programming (QP) approximations. A QP problem is a nonlinear problem with a quadratic cost function and linear constraints. The SQP method may fail to converge if the initial point is far from the local solution. To increase the convergence, a penalty function is used which is linear combination of the objective function and some measure of the constraint violation. This linear operator is referred to as the Lagrange multiplier. This augmented Lagrangian function is then used as the cost function for the SQP algorithm. A brief mathematical description is shown below.

The general nonlinear problem is given by,

$$\text{Minimize:} \quad f(\mathbf{x}) \tag{8.16}$$

$$\text{Subject to:} \quad g(\mathbf{x}) = b \tag{8.17}$$

The augmented Lagrangian function is,

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T (g(\mathbf{x}) - b) \tag{8.18}$$

The set of control variables is denoted by $\mathbf{x}$. $f(\mathbf{x})$ is the cost function, $g(\mathbf{x})$ the set of constraints and λ the set of Lagrange Multipliers. Several good references [50, 53, 57] explain the SQP optimization problem.

A quasi-Newton method was chosen to be used for the optimization of water injection rates in the reservoir model to maximize the profit associated with oil recovery. The reasons for this choice are:

1. Other relevant techniques discussed in Section 8.1 involve the extensive calculation of derivatives of not only the cost function but also of the states with respect to the control variables which translates into a lot of computational time.

2. The problem under consideration is an example of a "black box" type where the controls can usually be determined by forward simulations and does not require the computation of the derivatives of the cost function with respect to the state variables of the system or the derivatives of the state variables with respect to the other states and controls. It requires only the derivatives of the cost function with respect to the controls. This suggests a gradient based method as an ideal candidate for this type of an optimization problem.

3. The best known gradient techniques belong to the Newton's family. As discussed in Section 8.2.3, the quasi-Newton method eliminates the burden of computing the Hessian (second-order derivative matrix) of the cost function which would be encountered with the usage of the Newton's method.

8.3 Optimization Methodology

One of the most important contributions in this thesis, is the interfacing of an optimization toolkit to the control volume reservoir simulator MARS, developed by Yang [59]. The simulator was written in C++ and so were all the interface functions. The construction of the simulator has been described in [59]. Separate routines were written to compute the cost function and the gradient of the cost function with respect to the control variables. Several routines were added to the simulator to interface it with the optimization module which was developed using an optimization toolkit described in Section 8.3.2.

8.3.1 The quasi-Newton method

The motivation behind the quasi-Newton method was described in Section 8.2.4 and the reasons for the choice of this method to be applied in this work are given above. This section describes one of the most commonly used quasi-Newton algorithms and the one that is used in this study. To refresh the reader, the basic Newton's method is described by Equation (8.13). The quasi-Newton method replaces the Hessian matrix $\mathbf{H}(\mathbf{x}_k)$ by a positive-definite approximation $\widetilde{\mathbf{H}}_k$. The search direction $\mathbf{s}_k$ is given by

$$\mathbf{s}_k = -[\widetilde{\mathbf{H}}_k]^{-1}\nabla f(\mathbf{x}_k) \tag{8.19}$$

The Hessian $\widetilde{\mathbf{H}}_k$ is often initialized as the identity matrix and is updated after each iteration using the changes in $\mathbf{x}$ and $\nabla f(\mathbf{x})$ over the last two points. These changes are denoted by $\mathbf{d}$ and $\mathbf{y}$ where

$$\mathbf{d}_k = \mathbf{x}_{k+1} - \mathbf{x}_k, \tag{8.20}$$

$$\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k). \tag{8.21}$$

A widely used formula for updating the Hessian is the BFGS update named after Broyden, Fletcher, Goldfarb and Shanno [50] who developed it independent of each other in the same year. The update is given by

$$\widetilde{\mathbf{H}}_{k+1} = \widetilde{\mathbf{H}}_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{d}_k^T \mathbf{y}_k} - \frac{(\widetilde{\mathbf{H}}_k \mathbf{d}_k)(\widetilde{\mathbf{H}}_k \mathbf{d}_k)^T}{(\mathbf{d}_k)^T (\widetilde{\mathbf{H}}_k \mathbf{d}_k)}. \tag{8.22}$$

Dennis and Schnabel [55] proved that if $\widetilde{\mathbf{H}}_k$ is positive-definite and if $\mathbf{d}_k^T \mathbf{y}_k > 0$, $\widetilde{\mathbf{H}}_{k+1}$ is positive-definite as well. A line search parameter α_k is usually incorporated in the quasi-Newton method for quicker convergence. The convergence of a quasi-Newton method is always superlinear since it incorporates second order information. A BFGS algorithm requires more iterations than a comparable Newton implementation but each iteration of the quasi-Newton technique is faster than that in the Newton's method because second order derivatives are not required.

A disadvantage of the BFGS implementation is that for systems with thousands of variables, storing the $\widetilde{\mathbf{H}}_k$ matrices requires a lot of time and computer memory. In the past, the conjugate gradient techniques were considered more attractive due to this reason since they do not involve any matrix handling. Later, researchers improved the BFGS method to make it less memory intensive. Nocedal [60] developed a variation of the BFGS technique that used a variable amount of storage but still ensured the positive-definiteness of the matrices. Benson and Moré [61] further extended the procedure to obtain the minimum for a bound constrained problem where the variables are bound by an interval, that is

$$\text{Min } f(\mathbf{x}) : \; l \leq \mathbf{x} \leq u \tag{8.23}$$

where l and u are the lower and upper bounds for the variables $\mathbf{x}$. In this method, the Hessian is updated using projected gradients. This algorithm is referred to as the bound limited variable variable metric method (BLMVM) and has been implemented in the toolkit described in Section 8.3.2.

8.3.2 Toolkit for Advanced Optimization

The Toolkit for Advanced Optimization (TAO) [62], developed at the MCS division of the Argonne National Laboratory (ANL), is aimed at the design and implementation of component-based optimization software for the solution of large-scale optimization applications. The TAO design philosophy uses object-oriented techniques to create a flexible optimization toolkit. It reuses external tools where

appropriate. For instance, the design of TAO enables connection to linear algebra support provided in toolkits such as PETSc. The four fundamental objects used by TAO solvers to define and solve optimization problems are: vectors, matrices, index sets and linear solvers. The concepts of vectors and matrices are standard and are borrowed from PETSc, while the term index set refers to a set of integers that is used to identify specific elements of a vector or a matrix. An optimization algorithm is a sequence of well defined operations on these objects. The C++ include file for TAO is **tao.h** and should be used via the statement

#include "tao.h"

Some of the commands often required when using TAO are listed below.

- **TaoInitialize()** initializes TAO and also PETSc, if it not already initialized, and all programs begin with this command.
- **TaoFinalize()** is usually the final statement written by the user and handles the options to be called at the conclusion of the program. It also closes PETSc if it was initialized using **TaoInitialize()**.
- **TaoCreate()** is used to specify the optimization method to be used by the solver. Both unconstrained and constrained minimization can be performed using TAO solvers. Unconstrained minimization methods offered by TAO include conjugate-gradient methods, Newton's method with line search and trust region and the Limited Memory Variable Metric (LMVM) method which is a quasi-Newton method and does not require the computation of the Hessian. The constrained minimization methods include the Newton trust region method, the gradient-projection conjugate-gradient method, the interior-point Newton algorithm and the bound LMVM method. The bound LMVM method was used for the studies performed here.
- **TaoAppSetObjectiveRoutine()** and **TaoAppSetGradientRoutine()** are used to call the routines that compute the objective function and the function gradient respectively. Sometimes for more robust computation, the function and the gradient are called using the same command **TaoSetFunctionGradient()**.
- **TaoSolve()** is invoked to solve the problem after all the necessary problem parameters have been initialized and after all the required TAO and PETSc commands have been specified.

In addition, TAO uses many of the commands developed in PETSc for handling vectors and matrices.

8.3.3 Formulation of the Cost Function

In optimization, defining the objective of any process is of utmost importance. The definition of the objective plays a very important role in setting up the optimization problem. It is up to the operator to decide the objective of the problem. Several different cost functions have been suggested and used in the literature [63, 54, 64]. One of the most common cost functions is in the form net income deliverable over a certain period of time where the profit associated with oil production is maximized. The costs that negate the cost of oil recovery are the costs of water injection and production. A typical profit function is written as,

$$J = C_{OP} - C_{WI} - C_{WP} \tag{8.24}$$

It is alternately written as,

$$J = \sum_{i=1}^{N} (q_{wi}^{i} c_{wi} + q_{wp}^{i} c_{wp} - q_{op}^{i} c_{op}) \Delta t_i \tag{8.25}$$

Another cost function that was used for this study was the Net Present Value (NPV). NPV can be expressed as the sum of the discounted cash flows during the project period. A typical NPV function is as described below,

$$NPV = \sum_{i=1}^{N} \frac{((q_{wi}^{i} c_{wi} + q_{wp}^{i} c_{wp} - q_{op}^{i} c_{op}) \Delta t_i - FC^i)(1 - r^i)}{(1 + d)^{\frac{i \Delta t_i}{365}}} \tag{8.26}$$

Both cost functions were studied in this work and results for comparative studies have been presented in the latter half of this chapter. In both cases, a complete forward simulation was required to compute the cost function. The differences in the nature of both the functions is discussed in the results as well.

8.3.4 Gradient Computation

The logical step after computing the cost function is to compute the derivative of the function. It is not straightforward to compute analytical derivatives for the types of objectives described in the previous section. Therefore a good numerical derivative scheme needs to be employed to compute the gradient of the cost function. The definition of the derivative $f'(x)$ of a function $f(x)$ using a forward difference approximation is

$$f'(x) \approx \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \tag{8.27}$$

Though the derivative computation seems to be easy at first sight, it is hardly so. Choosing the right value of h is very crucial in computing the derivative accurately. The two main sources of error in Equation (8.27) are the truncation

and roundoff errors. The truncation error arises because of omitting the second and higher order terms in the Taylor series expansion shown below.

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + \cdots \tag{8.28}$$

The truncation error, as a result, is of the order of hf''.

The roundoff error arises primarily due to choosing h with a high *effective* value which refers to the difference between $x+h$ and x as represented in the machine. This value is of the order of $\epsilon_m x$ where ϵ_m is the machine's floating-point precision. The fractional error in the order of h can be expressed as an order of $\frac{\epsilon_m x}{h}$. A large fractional error translates into a large error in the derivative.

A significantly more accurate way to evaluate the derivative is using a central difference approximation to estimate it as follows,

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{8.29}$$

In this case, the roundoff error is about the same as before but the truncation error is now of the order of $h^2 f'''$. However, this too depends on how wisely the value of h is chosen.

An improved method to compute the derivative was suggested by Ridders [65]. In this method, the derivatives computed using finite-difference calculations are extrapolated, using Neville's algorithm for fitting a polynomial, with progressively smaller values of h so that eventually $h \to 0$. A subroutine, `dfridr`, in C has been reported in [66] and will be used for gradient computation in the optimization algorithm. As a side note, a forward difference scheme is used to compute the gradient at the lower control bound and a backward difference scheme is used at the upper control bound.

It is quite obvious that the gradient computation requires the function to be evaluated several times. During the entire optimization algorithm, the cost function is evaluated hundreds of times.

8.3.5 Optimization Algorithm

Water injection rate was chosen as the control variable for the optimization problem. The simulations were performed with a total fluid rate constraint. The total volume of water injected equalled the total volume of fluids produced. The simulation period was divided into multiple stages and the algorithm was used to optimize the water injection rates in all those periods. The number of stages determined the number of control variables. The gradient of the cost function with respect to the control variable was computed using the numerical scheme described in Section 8.3.4. The set of control variables obtained after each optimization iteration is referred to as the control policy. The initial set of values assigned to the water injection rates is known as the nominal control policy and

the final set of water rates is known as the optimal control policy. The TAO solver taoblmvm was used as the optimization method. A step-by-step description of the optimization algorithm is given below.

1. A starting control policy was assigned to start the optimization algorithm.
2. The simulation was run forward in time more than once to compute the cost function and the gradients.
3. The tao_blmvm routine employing a quasi-Newton method was used to perform optimization and compute the control policy for the next iteration.
4. Steps 2 and 3 are repeated till convergence is achieved in the solution to the control policy. An absolute tolerance of 10^{-4} was used as the convergence check for the cost function.

8.4 Case Studies

The results of applying the optimization algorithm to simple reservoir models are presented and discussed in this section. Two reservoir models were considered for the studies: one without fractures and one with. A third case of a heterogeneous reservoir model is studied to illustrate the differences between the method proposed by Yeten et al. [54] and the method developed here.

8.4.1 Case I

The domain used in this case study, shown in Figure 8.1, has dimensions of 40 ft $\times$ 40 ft $\times$ 1 ft and is gridded using blocks of size 10ft $\times$ 10 ft $\times$ 1 ft. The domain has a quarter-of-a-five-spot pattern with a single injector and a producer at the opposite corners. The porosity of the reservoir is 0.1 and the residual S_w is 0.2. The initial reservoir pressure is 3000 psi and the matrix has a positive capillary pressure (P_c) curve. The two cost functions described in Section 8.3.3 were used in the studies. In all the examples, the costs of injecting/producing fluids were assumed to be the same. The following values were used:

$$\begin{aligned} \text{Cost of oil produced } (c_{op}) &= \$50/bbl \\ \text{Cost of water produced } (c_{wp}) &= \$1/bbl \\ \text{Cost of water injected } (c_{wi}) &= \$1/bbl \end{aligned}$$

In the NPV formulation, the capital cost was assumed to be zero. The annual tax rate r was assumed to be 4.6 % and the annual discount rate was assigned a value of 16.7 %. These values were taken from [64] purely for the purpose of these examples. In reality, these values are bound to vary. *Study Ia:* In this case, the optimal water injection rate for the domain is computed using the net income or

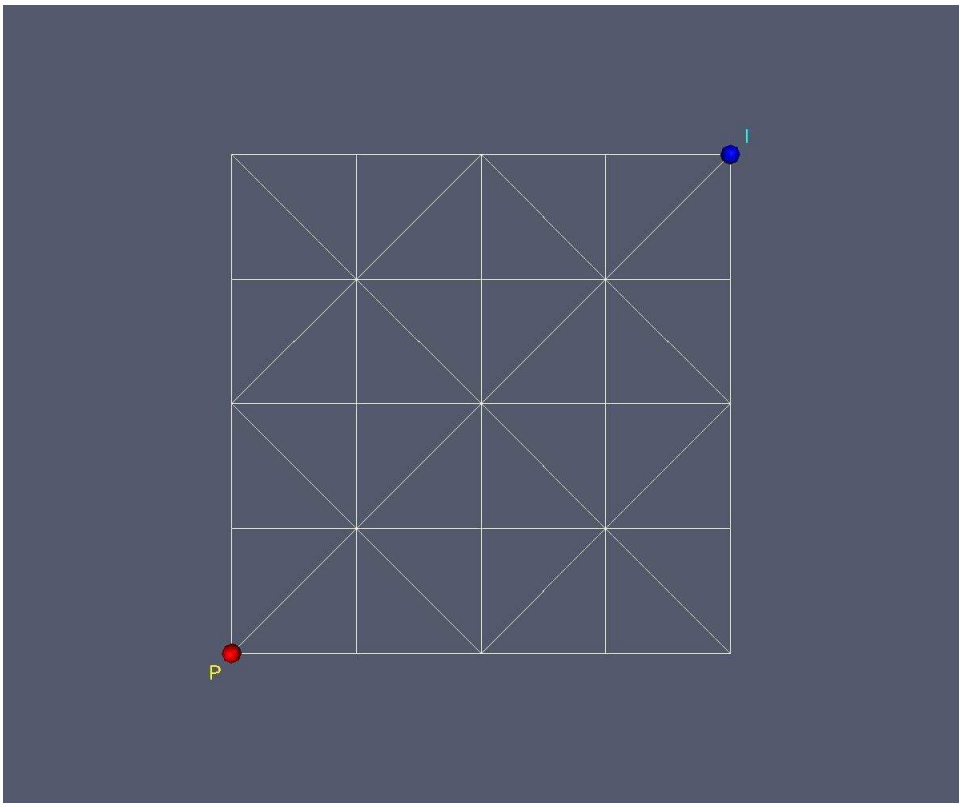

Figure 8.1: Finite-element mesh of the domain used in Case I.

profit as the cost function over a period of 2000 days. The single-stage optimum was determined to be 0.0501 bbl/day and the corresponding profit was determined to be \$515.76. This was found to be a unique optimum by providing different initial guesses to the optimization algorithm. Figure 8.8 shows the cumulative oil produced as a result of the optimal injection rate and compares it against the oil produced due to two base or uncontrolled cases. The base case 1 represents a case where the water injection rate is higher than the optimal injection rate and the base case 2 corresponds to a scenario where the injection rate is less than the optimal rate. Figure 8.9 shows the water cut obtained as a result of the optimal and the base cases. The water cut is the ratio of the volume of water produced to the volume of water injected. In base case 1, though more oil is produced than the optimal case, the corresponding water cut is much higher and hence is not very profitable. In base case 2, the oil produced is too little compared to the optimal case and hence is not profitable in the given period of time.

Study Ib: The entire period of simulation (2000 days) was divided into two periods or stages with each stage having a duration of 1000 days. The optimization algorithm was implemented to determine the optimal injection rates for the two stages. Different initial guesses were provided to the algorithm and a unique

optimum was not obtained. Three different optima are presented in Table 8.1. It can be noticed that there is no unique optimum although the profits are reasonably close to each other. The main reason for this is due to the fact that the total amount of water injected over the entire period is roughly the same in all the three cases. This is further corroborated by Figures 8.10 and 8.11, where it can be seen that the cumulative oil produced and the final water cuts are very similar. The presence of multiple optima for a two-stage optimization problem can be seen in Figure 8.2. A five-stage optimization was performed to observe the effect of

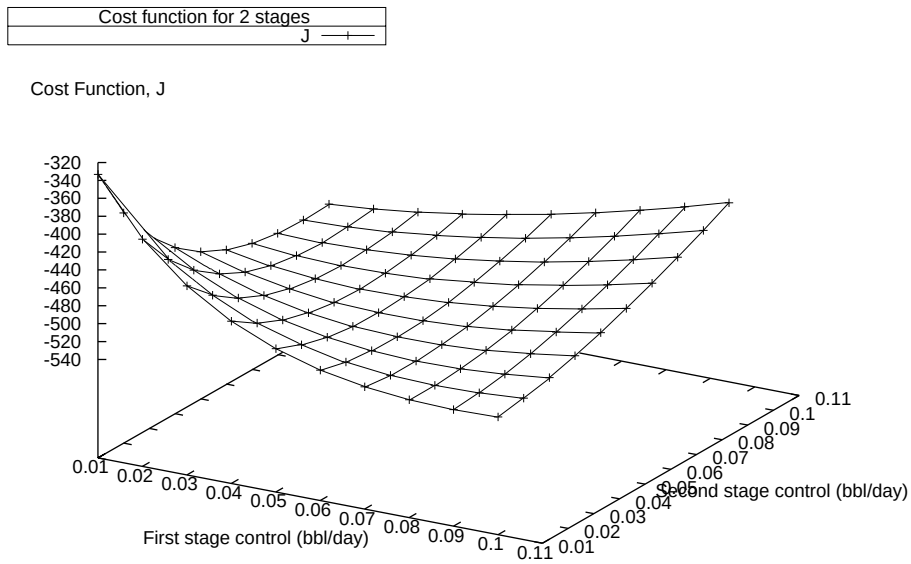

Figure 8.2: *Study Ib:* Cost function for a two-stage optimization problem.

Table 8.1: *Study Ib:* Optimal injection policies for a two-stage optimization for a profit-based objective.

Starting Policy (bbl/day)	Optimal Policy (bbl/day)	Profit ($)
$\left\{\begin{matrix}0.01\\0.01\end{matrix}\right\}$	$\left\{\begin{matrix}0.0552\\0.0482\end{matrix}\right\}$	516.25
$\left\{\begin{matrix}0.1\\0.01\end{matrix}\right\}$	$\left\{\begin{matrix}0.0982\\0.0\end{matrix}\right\}$	520.95
$\left\{\begin{matrix}0.02\\0.05\end{matrix}\right\}$	$\left\{\begin{matrix}0.0343\\0.0645\end{matrix}\right\}$	515.195

optima on the number of stages. The cumulative oil production for three different

optimal policies obtained using different nominal policies are shown in Figure 8.12 and the corresponding water cuts can be read from Figure 8.13. Table 8.2 presents the different optima obtained using different initial conditions. It can be easily shown that some of the optimal results obtained are purely mathematical and can be easily avoided by the operator by taking a look at them. For instance, the Optimal Policy 1 from Figure 8.12 is not a practical way to recover oil from the field since very little water is injected in the first half of the operational period. In this case, almost no oil is recovered for a long time and this is not desirable from an economical viewpoint. Similarly, in the Optimal Policy 3, the water injection rates in all the five stages are almost identical to each other and are very similar to the single stage optimum obtained in *Study Ia* which makes the five-stage process a pointless exercise. The operator may safely choose the Optimal Policy 2, where bulk of the oil is produced in the first few hundred days and there is no such thing as a penalty for injecting a lot of water at the later stages. This sits well with the physics of the process and is very profitable. As can be seen from Table 8.2, the profit obtained in this case is also significantly higher than the corresponding value in the single-stage case. However, this is not a conclusive evidence to assert that varying the injection rates over the period of operation yields a better profit than the single-stage scenario.

Table 8.2: Five-stage optimal Policies for domain without fracture

Starting Policy (bbl/day)	Optimal Policy (bbl/day)	Profit ($)
$\left\{\begin{matrix}1.0\\1.0\\1.0\\1.0\\1.0\end{matrix}\right\}$	$\left\{\begin{matrix}0.0\\0.0257\\0.1680\\0.0313\\0.0054\end{matrix}\right\}$	523.27
$\left\{\begin{matrix}0.01\\0.01\\0.01\\0.01\\0.01\end{matrix}\right\}$	$\left\{\begin{matrix}0.2364\\0.0067\\0.0\\0.0\\0.0\end{matrix}\right\}$	524.33
$\left\{\begin{matrix}0.001\\0.001\\0.001\\0.001\\0.001\end{matrix}\right\}$	$\left\{\begin{matrix}0.0501\\0.0484\\0.0462\\0.0447\\0.0385\end{matrix}\right\}$	515.80

The optimization algorithm can be used to determine optimal policies for multi-stage problems in a similar manner. In some cases, there tends to be a small increase in the profits obtained with an increase in the number of stages. This could, possibly, be attributed to the type of solutions obtained in a multi-stage

problem. For instance, a solution which indicates a sharp rise in oil production due to a high volume of water injection during an early stage followed by a sharp decrease in water injection, possibly even a well shut-in, which leads to little or no oil production because most of the oil has been swept out of the reservoir, will yield a high profit due to low water injection rates at later stages and due to the fact that most of the oil has been drained out of the reservoir.

Study Ic: A comparative study of the profit function and the NPV formulation is presented in this section. The optimal injection rate obtained by using profit as the objective is compared with that obtained by using NPV as the performance criterion. The profit-based optimum is described in *Study Ia*. The optimal water injection rate in the NPV-based case was determined to be 1.02 bbl/day and the corresponding profit was observed to be \$400.10. This objective value is significantly lower than its counterpart in *Study Ia* and the optimal rate is much higher than the corresponding rate in *Study Ia*. This is due to the natures of the two different cost functions under question. The profit function sums the profit of oil over the entire period of time whereas NPV sums the profit but discounts the value of this profit at later times. In other words, the NPV dictates that the oil be produced as quickly as possible under the imposed operating constraints. This can be witnessed in Figure 8.14 where the oil is produced very rapidly and there is very little oil left to produce at later times. As a result, the water produced at later times is quite high to account for mass balance of the fluids. A high water cut at later times can be seen in Figure 8.15. The difference in oil production and water cut patterns between the two objectives can be clearly seen in these two figures. The operator has to choose between these clearly different objectives based on the facility constraints that he faces.

Study Id: A two-stage NPV problem was studied here and compared with a single stage problem with the same objective. The results of the cumulative production and water cut comparisons are given in Figures 8.16 and 8.17 respectively. It can be seen that the curves are almost identical for the two-stage and the single stage cases. This is due to the nature of the NPV cost function. There is no unique optimum for the later stages in a NPV-based multi-stage solution. The function, as discussed earlier, directs the algorithm to yield a high enough water injection rate in the first stage to sweep out most of the oil early on. The amount of water injected in the second or later stages is not very significant in determining the value of NPV. The two different optimal policies for a two-stage problem based on NPV as the objective can be viewed in Table 8.3.

8.4.2 Case II

The domain used in this study, shown in Figure 8.3, is a 60 ft $\times$ 60 ft $\times$ 1 ft $2-$dimensional domain with fractures. The domain has a single injector and a single producer as can be seen from the figure. The porosity of the reservoir is 0.2 and the residual S_w is 0.05. The initial reservoir pressure is 1000 psi and

Table 8.3: *Study Id:* Optimal injection policies for a two-stage optimization for a NPV-based objective.

Starting Policy (bbl/day)	Optimal Policy (bbl/day)	Profit ($)
$\left\{\begin{matrix}0.1\\0.1\end{matrix}\right\}$	$\left\{\begin{matrix}1.0368\\0.1\end{matrix}\right\}$	399.12
$\left\{\begin{matrix}10.0\\10.0\end{matrix}\right\}$	$\left\{\begin{matrix}0.9962\\9.43\end{matrix}\right\}$	399.13

the matrix and the fracture have the same positive capillary pressure curve. The relative permeabilities of the oil and water phases are shown in Figure 8.4 and the capillary pressures in the matrix and the fractures obey the curve shown in Figure 8.5. The formation volume factors and viscosities of the fluids can be found in Tables 8.4 and 8.5 respectively. The profit function was used in the studies. The fluid costs used in this case are the same as in **Case I**.

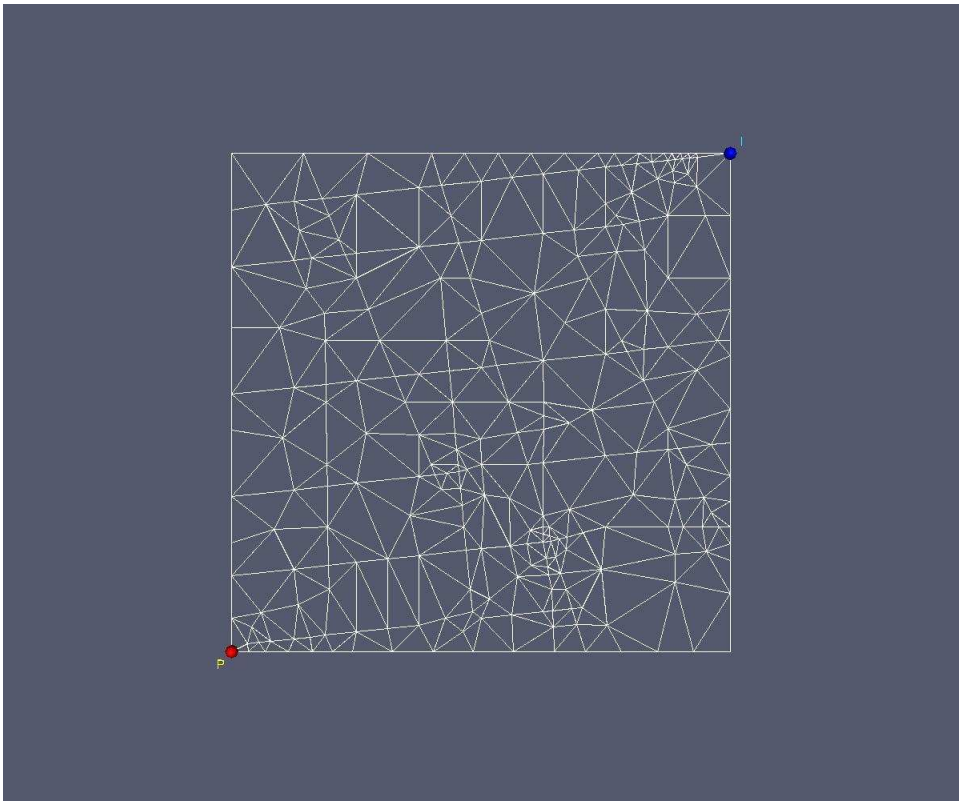

Figure 8.3: Finite-element mesh of the fractured domain used in Case II.

Study IIa: The purpose of the studies in this section is to apply the optimization algorithm for domains with complicated geologic features such as faults and

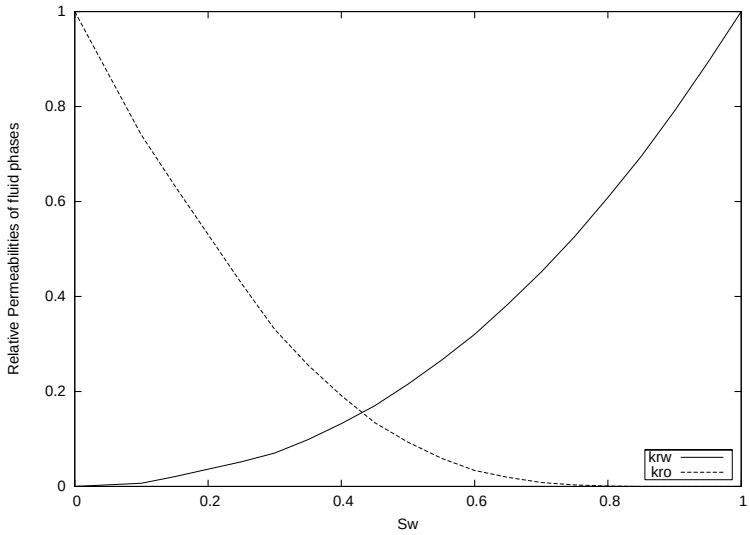

Figure 8.4: Relative Permeabilities of the fluids for domain in Case II.

fractures. The optimal injection rate was computed for a single stage problem as in *Study Ia*. The optimal case was compared to a base or uncontrolled case and the results for cumulative oil production and water cut comparisons are shown in Figures 8.18 and 8.19. The simulation was performed for 1000 days and the optimal injection rate and profit were determined to be 0.5 bbl/day and $1703.87 respectively.

Study IIb: It is important to know the period of operation of the reservoir before determining the optimum. This study uses the fractured domain to distinguish between the optimal policies for two different periods of time. The optimal rate for a period of 1000 days is described in *Study IIa*. In this study, that optimum is compared to the optimal rate for a period of 5000 days. This comparison is provided in the form of Table 8.6. The plots of cumulative oil production and water cut due to the optimal injection rate obtained for a period of 5000 days are shown in Figures 8.20 and 8.21 respectively. The most striking result is that the optimal injection rate for the shorter period is greater than that for the longer duration of operation. This is due to the nature of the profit function. A NPV-based objective will suggest different optimal policies without any significant difference between them.

8.4.3 Case III

Results comparing the optimization method developed by Yeten *at al.* [54] to arrive at stage-wise or multistage solution to a reservoir optimization problem and the approach followed in this study have been presented in this section. In the approach of Yeten *et al.*, the simulation period was divided into several stages to change the control at those times. For each period, the optimum was computed

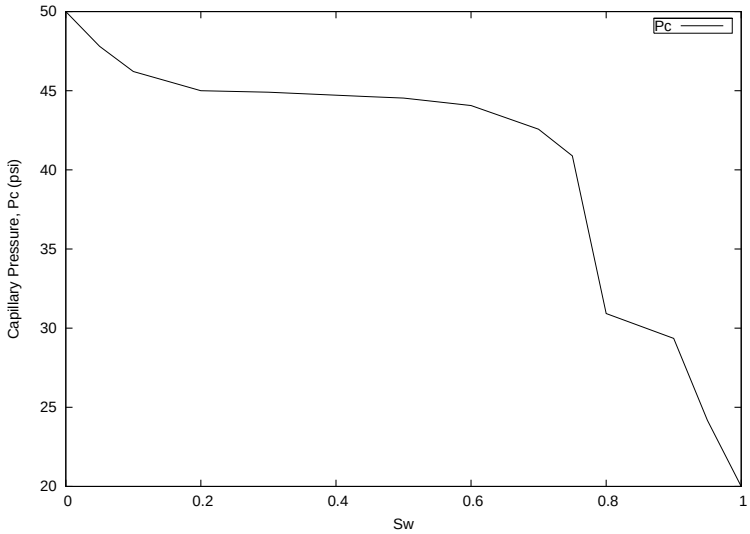

Figure 8.5: Capillary Pressures in the matrix and fractures for domain in Case II.

by maximizing the oil recovery for the remaining simulation period with the simulation being started from the end of the previous period for each optimization run. For example, if the controls were to be altered twice during the simulation run, the simulation run period was divided into 2 stages. First, the simulation was performed from the initial time to the end to compute a single stage optimum. This was assigned as the optimum for the first stage. The simulation was then restarted from the time at the end of the first stage and performed to the end of the simulation period, that is the second stage in this example. The optimum obtained in this case was fixed to be the optimal policy for the second stage. The optimization method developed by Yeten *et al.* is demonstrated in Figure 8.6.

The study in this section compares this approach to the optimization method adopted in this study. The 2−dimensional heterogeneous domain shown in Figure 8.7, with the two layers having different porosities of 0.1 and 0.2 and permeabilities of 50 and 200 millidarcies, was used for this comparative study. Each layer is 40 ft $\times$ 19.5 ft and the layers are separated by an impermeable shale barrier which is 1 ft thick. Gravitational effects are taken into account. The fluid properties are the same as the ones for the domain used in **Example I**. The simulation period was 2000 days in both cases and the two stages were assigned to be $0-1000$ and $1000-2000$ days. Table 8.7 shows the optimal policies for a problem with 2 stages solved using the procedure of Yeten *et al.* Comparison of these values with those in Table 8.8 makes it evident that the results of the two stage problem using their approach are the same as the single stage results obtained using the approach developed in this work. In other words, the total amount of water injected in the wells for the optimal cases is the same in both approaches. This was verified for different initial conditions. It is therefore, safe to state that the multistage solution obtained using the procedure of Yeten *et al.* is equivalent to the single

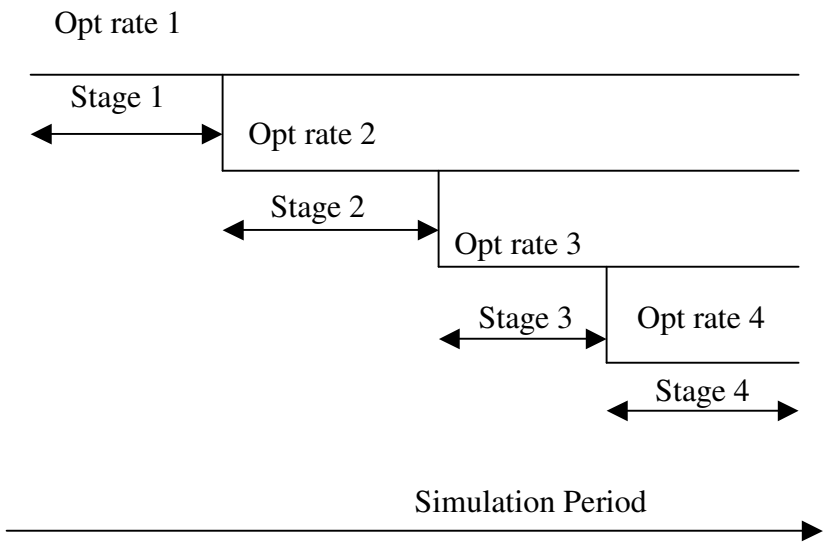

Figure 8.6: Diagrammatic description of the optimization method developed by Yeten *et al.*

stage approach of this study. Further studies showed the absence of a unique optimum for the multistage case. The results for a two-stage problem using the same domain are presented in Table 8.9. I_1-opt and I_2-opt refer to the optimal injection rates of injectors I_1 and I_2. It can be noticed that the amount of water injected in this case is approximately the same as the single stage case which shows that, in the multi-stage case, the quantity of water injected is distributed between the stages. The lack of a unique optimal solution merely shows that there is no unique way to distribute the amount of water injected.

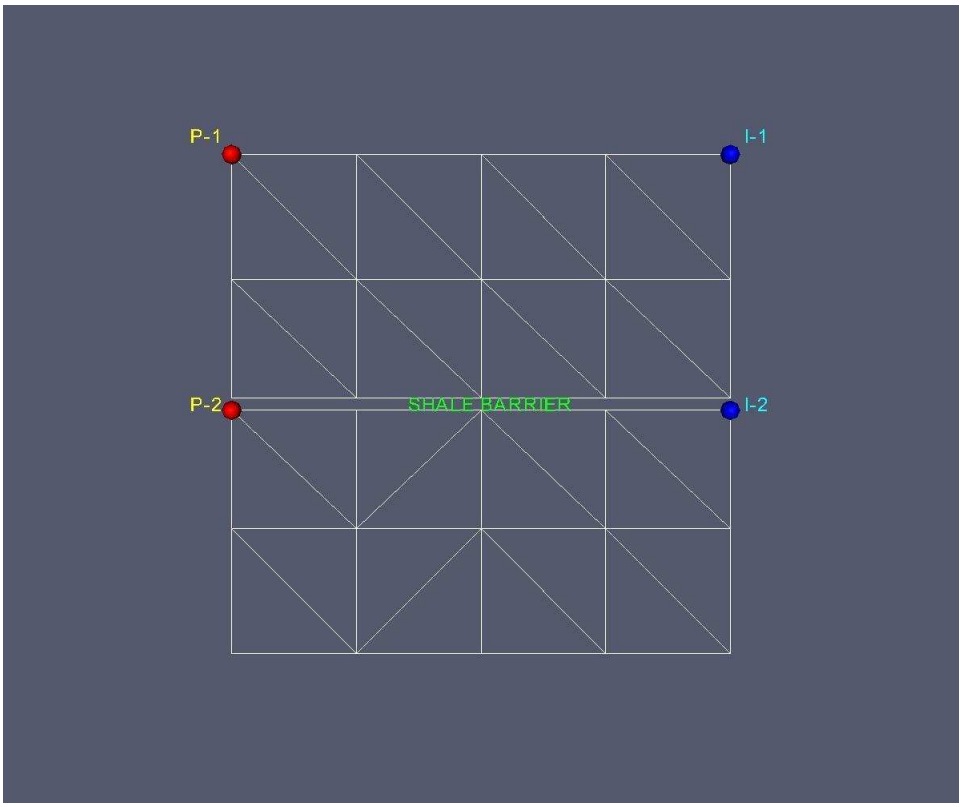

Figure 8.7: Finite-element mesh of the domain used in Case III.

Table 8.4: Formation Volume Factors of the fluids for the domain in Case II.

P_o (psi)	B_w	B_o
14.7	1	1
10000	1	0.7962

Table 8.5: Fluid viscosities for the domain in Case II.

P_o (psi)	μ_w (cp)	μ_o (cp)
14.7	0.5	10
10000	0.5	10

Table 8.6: *Study IIb:* Comparison of optimal policies for a single-stage optimization for different periods of operation.

Period of Operation (days)	Optimal Policy (bbl/day)	Profit ($)
1000	0.5	1703.87
5000	0.08	1668.55

Table 8.7: Case III: Optimal Policies for the two-stage optimization using the approach of Yeten *et al.*

I_1-opt (bbl/day)	I_2-opt (bbl/day)
$\left\{ \begin{array}{c} 0.0258 \\ 0.0259 \end{array} \right\}$	$\left\{ \begin{array}{c} 0.0504 \\ 0.05 \end{array} \right\}$

Table 8.8: Case III: Optimal Policies for the single stage optimization using the method developed in this work

I_1-opt (bbl/day)	I_2-opt (bbl/day)
0.0254	0.0504

Table 8.9: Case III: Optimal Policies for the two-stage optimization due to different initial conditions using the method developed in this work

I_1-opt (bbl/day)	I_2-opt (bbl/day)
$\left\{\begin{matrix}0.0194\\0.0324\end{matrix}\right\}$	$\left\{\begin{matrix}0.0515\\0.0492\end{matrix}\right\}$
$\left\{\begin{matrix}0.0486\\0.0006\end{matrix}\right\}$	$\left\{\begin{matrix}0.0210\\0.0795\end{matrix}\right\}$
$\left\{\begin{matrix}0.0551\\0.0\end{matrix}\right\}$	$\left\{\begin{matrix}0.1\\0.0\end{matrix}\right\}$

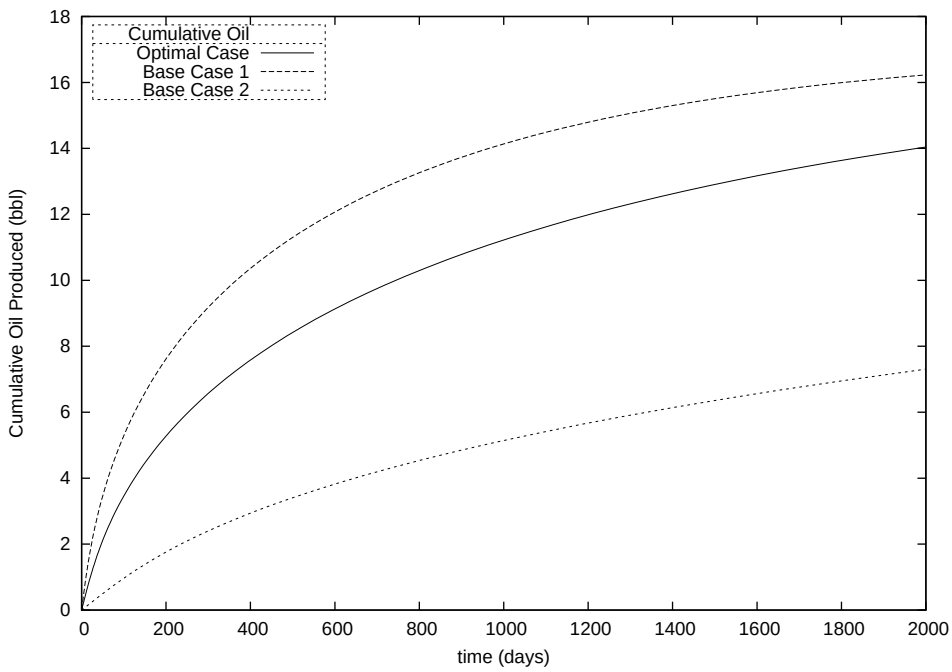

Figure 8.8: *Study Ia:* Cumulative oil production comparison.

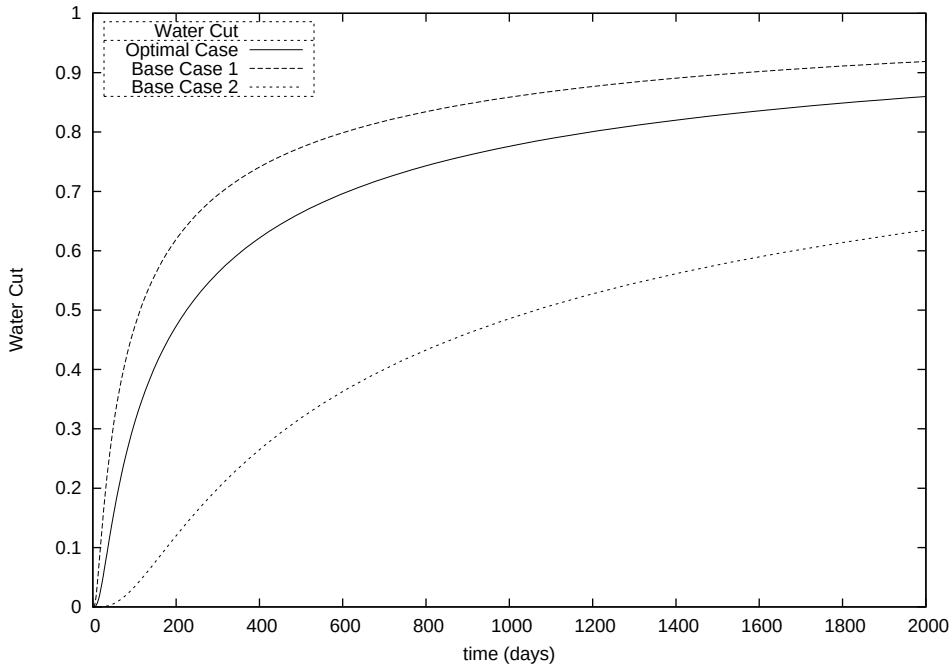

Figure 8.9: *Study Ia:* Water cut comparison.

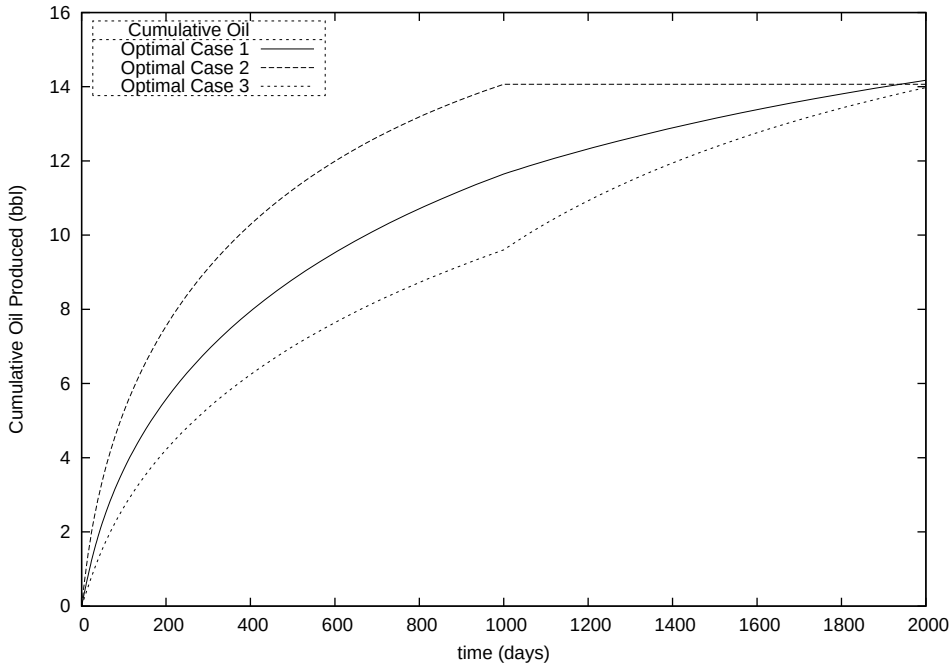

Figure 8.10: *Study Ib:* Cumulative oil production comparison for the two-stage problem.

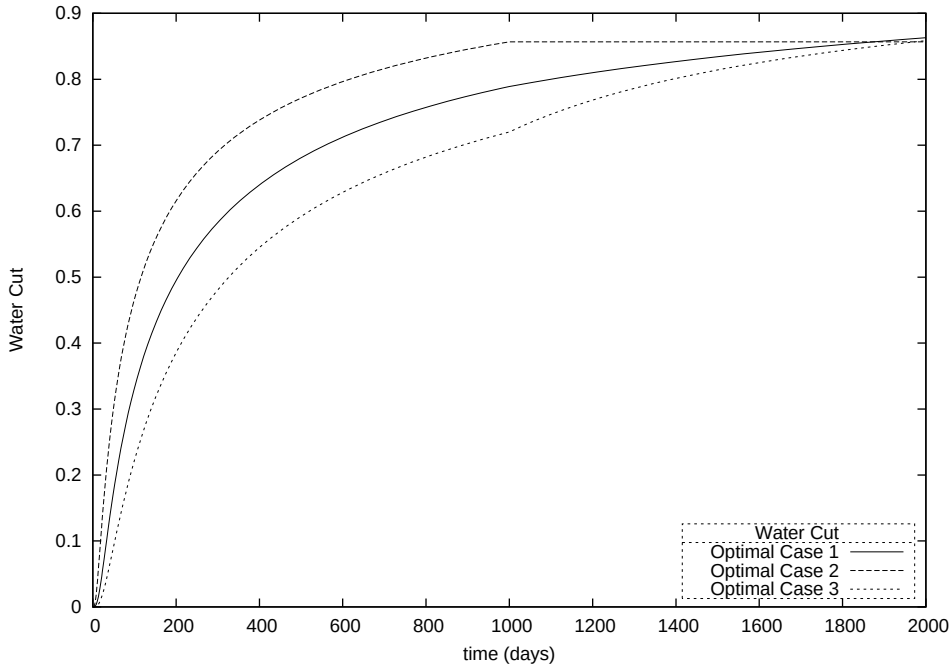

Figure 8.11: *Study Ib:* Water cut comparison for the two-stage problem.

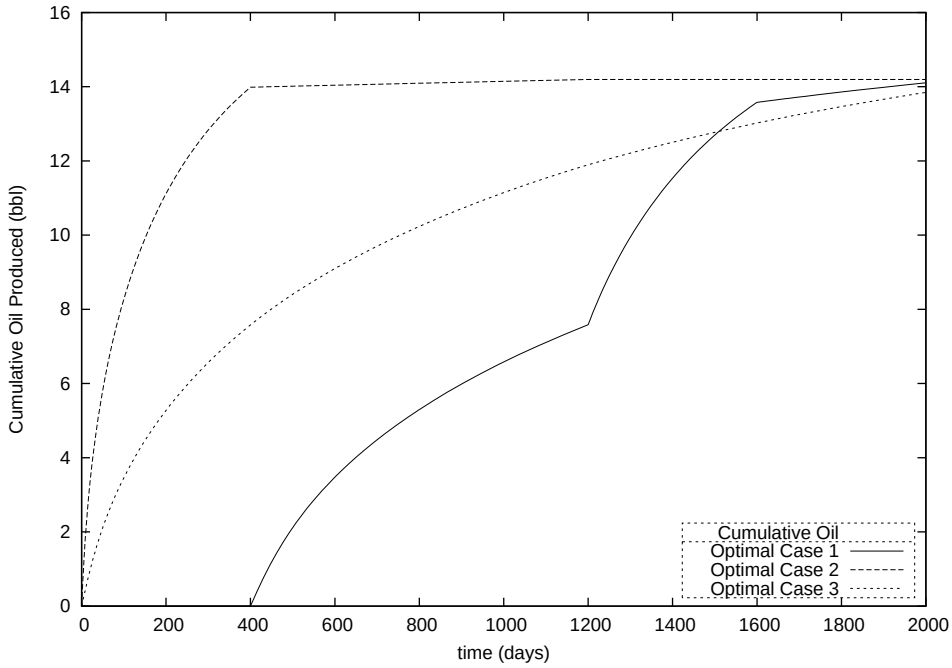

Figure 8.12: *Study Ib:* Cumulative oil production comparison for the five-stage problem.

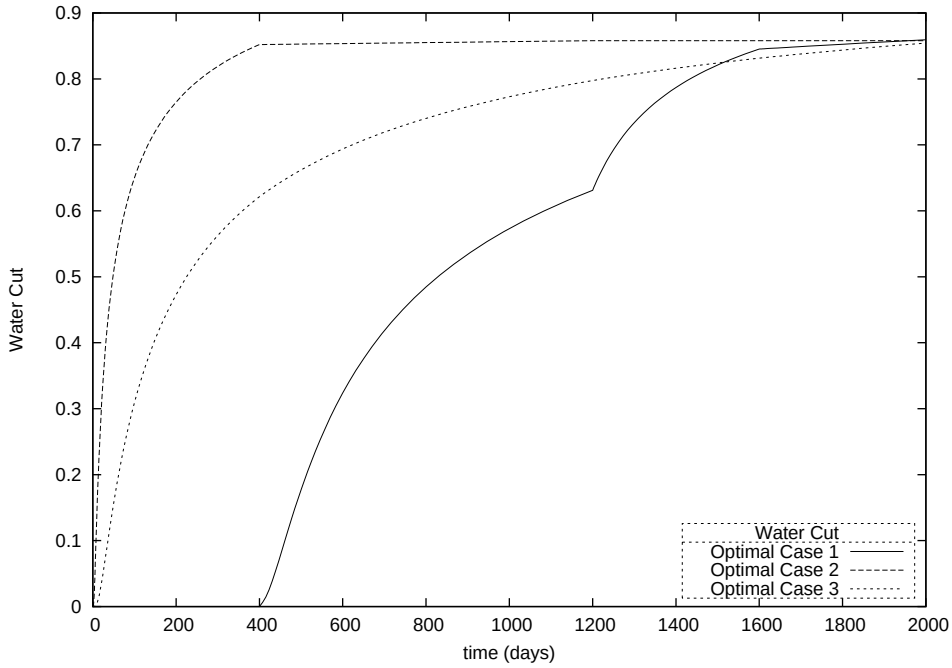

Figure 8.13: *Study Ib:* Water cut comparison for the five-stage problem.

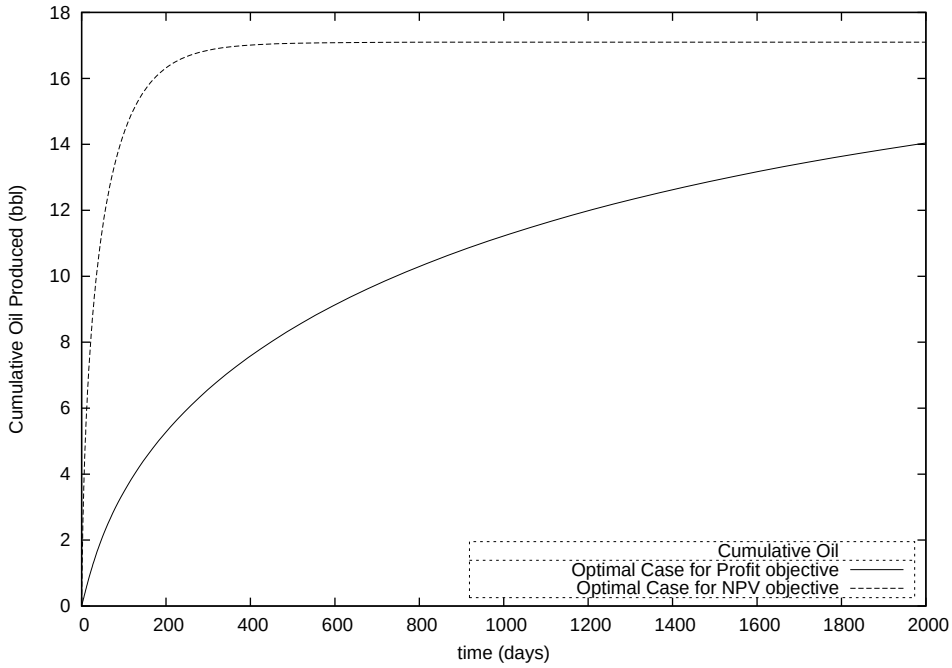

Figure 8.14: *Study Ic:* Cumulative oil production comparison.

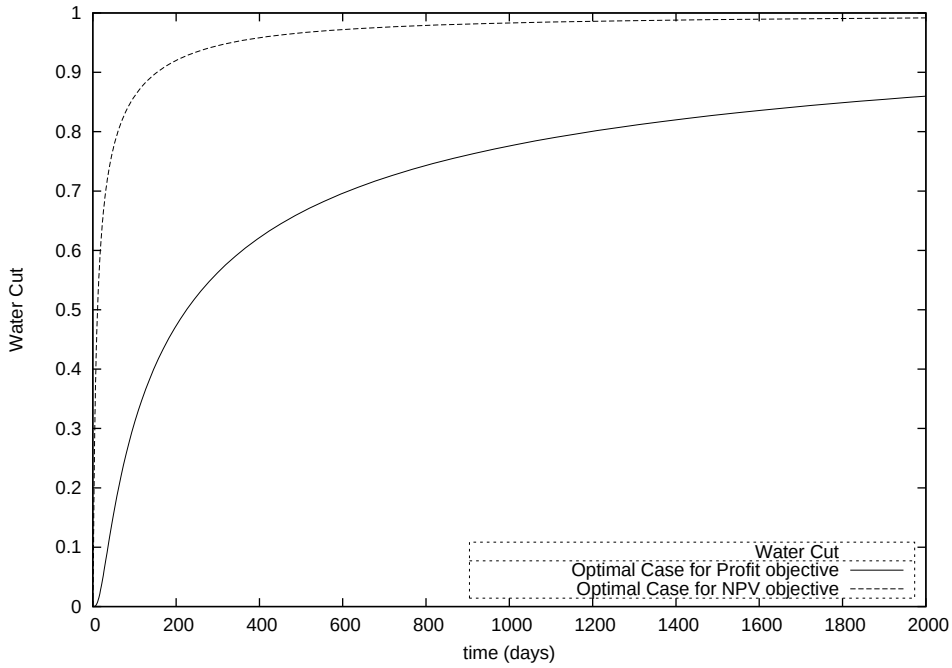

Figure 8.15: *Study Ic:* Water cut comparison.

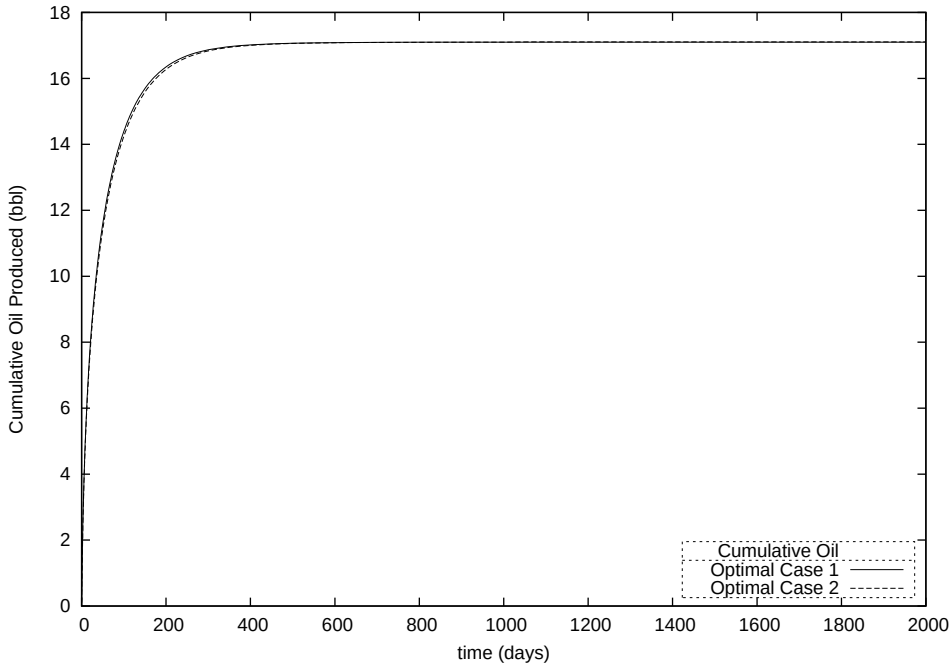

Figure 8.16: *Study Id:* Cumulative oil production comparison.

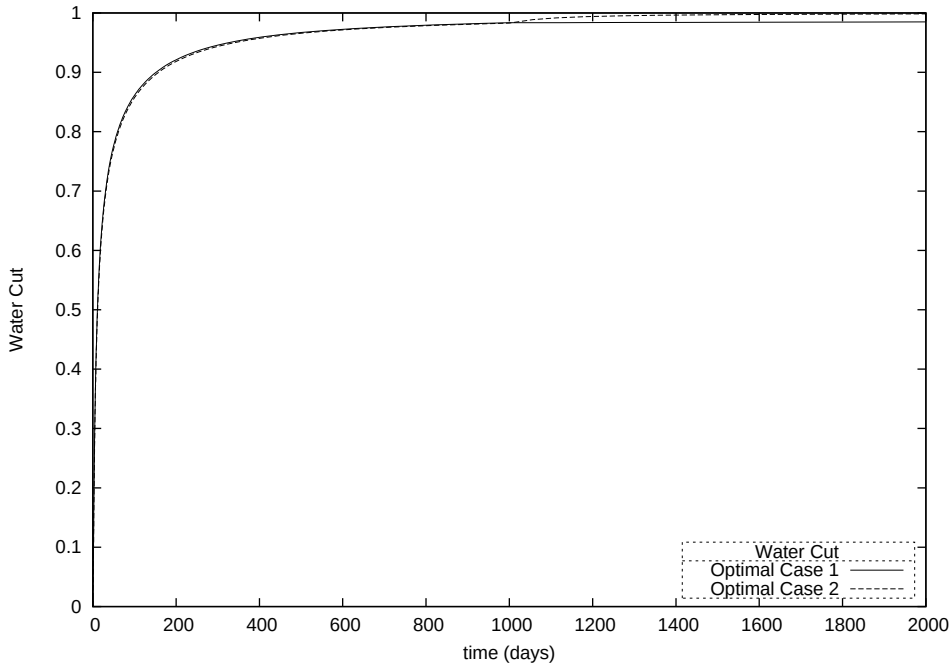

Figure 8.17: *Study Id:* Water cut comparison.

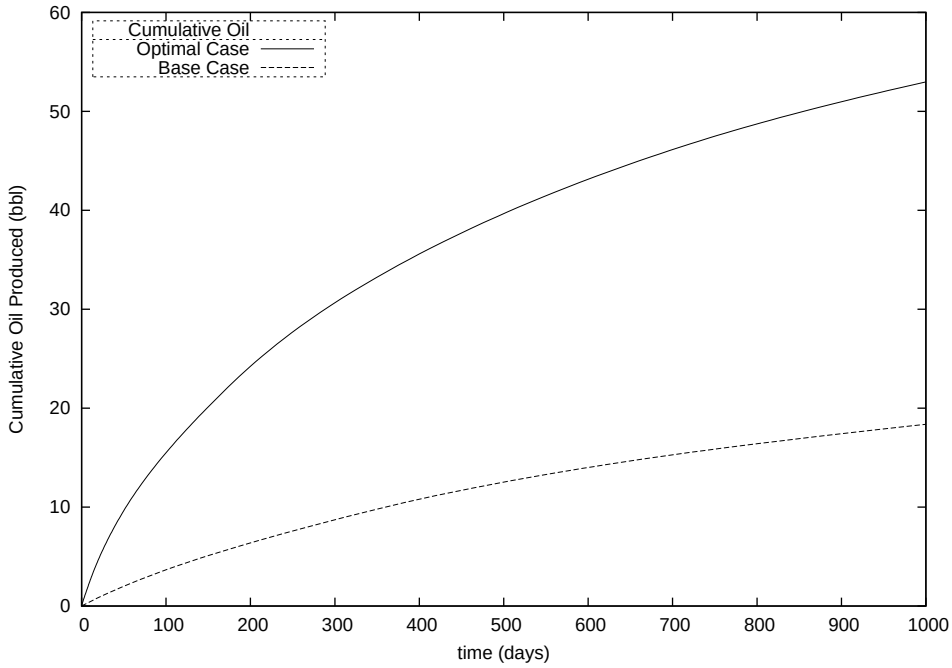

Figure 8.18: *Study IIa:* Cumulative oil production comparison.

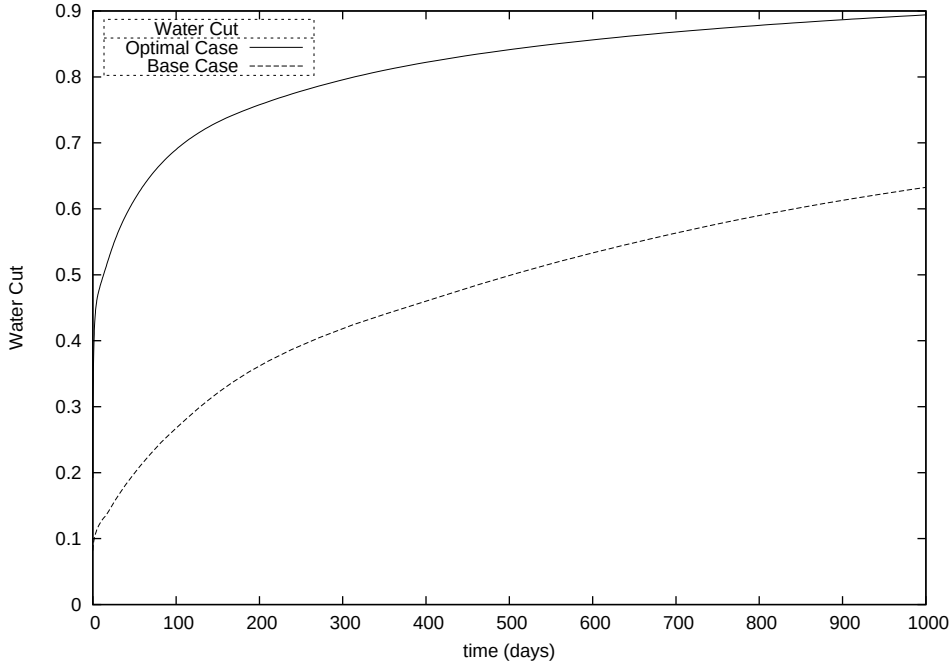

Figure 8.19: *Study IIa:* Water cut comparison.

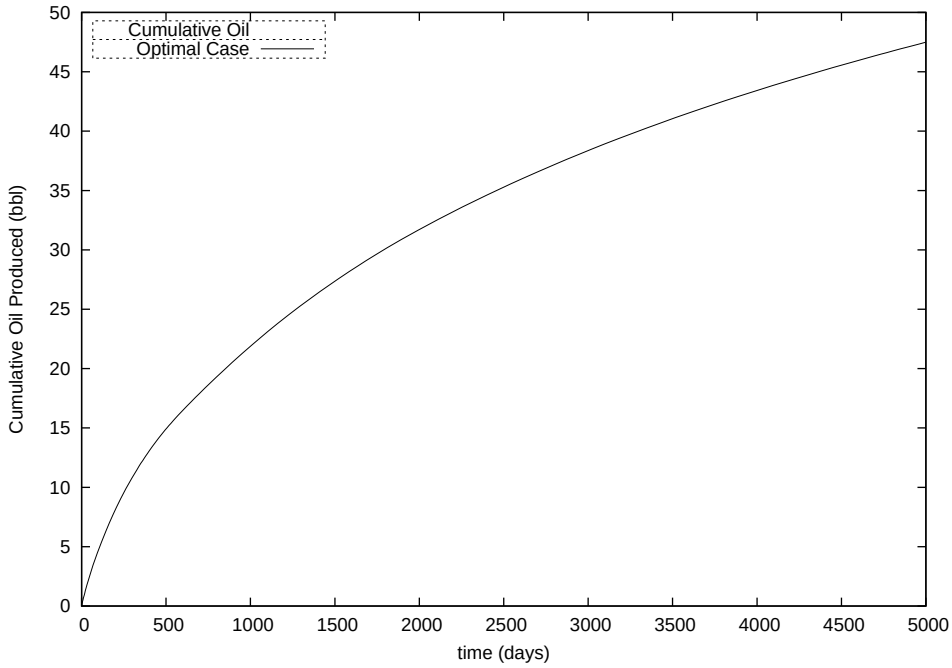

Figure 8.20: *Study IIb:* Cumulative oil production for 5000 days.

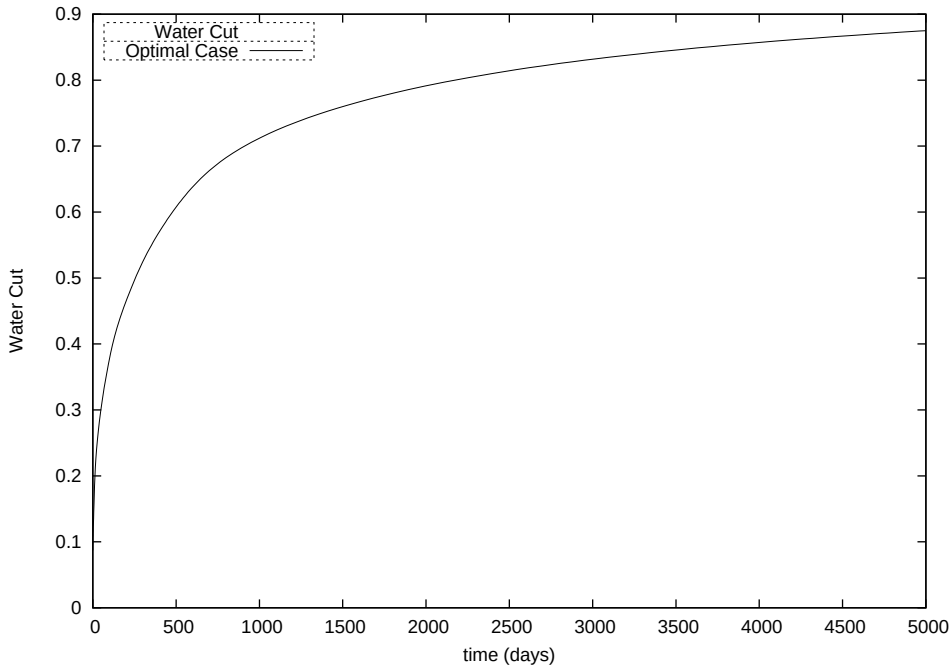

Figure 8.21: *Study IIb:* Water cut for 5000 days.

Chapter 9

Parallel Computation

Parallel computing studies were carried out using an 18 processor Linux cluster. The cluster has eight dual-processor nodes and one master node. Each of the processors is equipped with a a 64-bit AMD Opteron chip. A two-dimensional system consisting of 250,000 nodes was tested. A waterflood study was performed. The computational time on mutiple processors was compared with the base case of computational time on a single processor. Message Passing Interface (MPI) was used to distribute the domains among different processors. The scaleup performance is shown in figure 9.1. A speedup of between 11 and 12 was obtained for 16 processors. The optimization of parallel performance for a variety of domains is being examined.

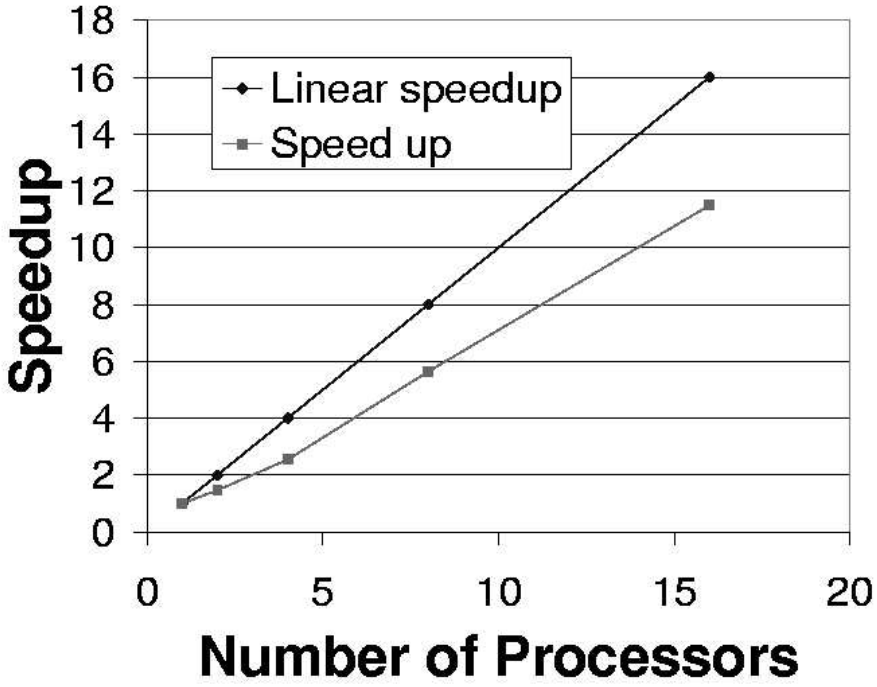

Figure 9.1: Scaleup performance on a parallel linux cluster. A two-dimensional, 250,000 node problem was tested.

Chapter 10

The Reservoir Simulator Interface

10.1 Introduction

10.1.1 Overview

The UFES Reservoir Simulator Interface

The UFES Reservoir Simulator Interface consists of two separate service packages: the UFES Reservoir Simulator Interface Server package, and the UFES Reservoir Simulator Interface Client package.

The interface server package is located on the computer which is supposed to handle all of the detailed computations within reservoir simulation. The mesh generator and the finite element simulator should also be on the same computer as the server package. The interface client package should be installed on the end-user's computers.

Both programs are written in JAVA and operate on any computer with JAVA Runtime Environments (JRE) available. Since the current simulator is compiled in LINUX/UNIX environments, the Interface server package should be executed under LINUX/UNIX operating systems. The interface client package can be installed on any computer on which JRE is installed. IBM RS/6000, Silicon Graphics, SUN, Pentium-based PCs, AMD-based PCs etc with more than 256 Mb of RAM would all be appropriate to run these applications.

About this chapter

This section contains technical descriptions of the principal features in both packages. Seven sections are included in this chapter. The first section is the introduction which includes the overview, objects and design. In this section, the project objectives have been reviewed and the basic concept has been introduced. Section 2 deal with developments of the current interface. The work flow, features and keywords are introduced in this section. Section 3 describes the installation part to show how the user can install this interface. Server side installation, client side

interface installation and some environment parameters are discussed. Section 4 is the focus of the client side reservoir domain data input. For example, the creation of a two-dimensional reservoir domain, fractures, and wells by drawing on a canvas is described. Furthermore, user can modify the domain, fracture and well coordinates in the automatically generated XML file. Section 5 goes through the data input XML file in detail. This is a very important section which contains all of the information about the reservoir simulation input file. Section 6 helps the end user operating this interface to complete a petroleum reservoir simulation by providing step-by-step instructions. Section 7 introduces a few ancillary programs bundled with the client side interface. It includes the triangle mesh viewer, interface input/output console, Results images viewer and the XML source file viewer.

10.1.2 Objects

Reservoir simulation is an important tool in modern-day reservoir management. Calculation of accurate pressure and saturation distributions requires solution of coupled partial differential equations. When the system of interest is a highly heterogeneous, anisotropic porous medium, accurate representation and simulation requires the use of high performance computing. High performance computing know-how and infrastructure are not easily available to independent producers, who might benefit from the technology. One of the objectives of this project was to make this technology accessible to independent operators through remore access via the internet. The idea was to have the simulator reside in a remote site (server) and provide the user with a client interface to interact with the server and perform the calculations. Since the client-side computer requirements are minimal, this approach would provide wider access to sophisticated computing.

Some of the considerations that went into planning this interface were:

- Robustness of the server-side simulator
- Ease of use on the client side
- Platform independence on the client side
- Speed of computing
- Quick and easy transfer of results to the client

10.1.3 Design

With the needs identified above, Java was chosen as the language of implementation. Platform independence and compatibility of client and server protocols were the overriding factors in this decision. Use of Java meant that, as long as a Java Running Environment (JRE) was installed on the client side, simulations could

be performed remotely. The interface was designed to be modular - there were two layers in the implementation process; the server side interface and the client side interface. The Java Socket was chosen as the data transfer channel. A full set of communication protocol was defined for these two layers.

The server side interface directory should includes the following items:

- The server side interface Java classes for client side commands waiting, commands accepting, commands sorting, detail job order submitting on local server, job results detecting on local server and transferring back to the client etc.
- The finite element meshing software packages for the meshing of reservoir domain with its fractures, wells information, etc.
- The reservoir simulator.

The client side interface has the following functions:

1. There is one Java class to host the main frame which is written by Java Swing. The title bar, file menu, tool bar and drawing canvas should be listed in this frame.
2. One Java class is designed for reservoir drawing by Java 2D Graphics technique: closed domain boundary is in polygon mode; fractures is in 2D line mode; injection wells is in circle mode; production well is in square mode. End-user can use mouse functions like click and drag to draw rough reservoir domain. For every action on the domain drawing, the relative coordinates information will be recorded into the XML format data input file automatically. Also, this class can handle a saved file instead of drawing from the very beginning.
3. One Java class was planed for communicating with server to finish the reservoir mesh. This class was designed to pick up the current domain information from the drawing canvas (basically the coordinates of all vertices such as boundaries, fractures and wells) and send it to the server through Java Sockets with its meshing order by specific protocol, then accepting the meshed results from the server and transferring the meshed info into client frame. At the same time, it will combine the mesh info with the reservoir description file to form final simulation data input XML file automatically.
4. One Java class was used as the communicating with the server to complete the final simulation. It was built to send the final simulation data input XML file into the server with its simulating order, then wait and accept the simulation results in real time. Furthermore, after accepting every image file from the server, this class would call another class to display the image result.

5. One Java class is required to display the results, either mesh results or final simulation results.

6. One Java class was needed to generate another window as Java input/output monitor. This is called the Java Input/Output Console.Using this class, end-user can easily see what process the simulation status and error messages. Also, this function was very helpful in debugging the code.

7. Multitrhreading implementation was necessary to run multiple jobs on the server without having to wait for jobs to finish.

10.2 Interface Development

10.2.1 Work Flows

The basic simple work flow of the reservoir simulation interface developed is as follows.

For a new reservoir domain:

1. Open the client side interface executive file (UFESSimulator.jar). The default domain information of the interface is to create a new reservoir domain.

2. Draw the closed domain boundaries first.

3. Draw the fractures, injection wells, production wells within the reservoir domain boundaries.

4. Save all of the coordinates information mentioned above into "*_ori.xml" file. '*' is the file name. "ori" means original file which is not meshed yet. XML is the file format.

5. Connect the client side interface with the server and send the domain information mentioned above to the server for meshing

6. Receive the meshed information from the server and automatically generate a final data input XML file as "*_fin.xml" format. "fin" represents "final" which is distinguished from "ori".

7. Use a file editor (OpenOffice or WordPad) find and open "*_ fin.xml" file and make related modifications such as the wells informations, simulation time etc. The details will be talked in later sections.

8. Connect the client interface with server, send the modified "*_fin.xml" file to the server and ask the server side simulator to run the simulation.

9. Interface server detects the simulation results in real time and send the results back to clients.

10. Receive the simulation results from the server and display them on the client side local machines (either by slide show or animation).

11. With a previously saved simulation domain, the only differences of simulation work flow are items 2 and 3 from the new reservoir domain simulations.

12. On the second step: use "open" a saved "*_ori.xml" file action instead of "Draw the closed domain boundaries first".

The user can make modifications to geometrical features. Well information, etc. is modified by opening the xml input file and modifying it directly.

10.2.2 Features

All of the interface code is in Java. The codes is compiled and zipped into an executable file by Java. The interface runs on JRE 5.0 and above under any operating systems. This makes the interface accessible to users across all computing platforms.

The data files in the interface and in the simulator are in XML format. XML is a standardized markup language for documents containing structured information. Structured information contains both content (words, pictures, etc.) and some indication of what role that content plays. Almost all documents have some structure. Reservoir simulation data files, include complicated data information about boundaries, fractures, wells, simulation period, etc. and are highly structured documents. The XML specification defines a standard way to add markup to documents. The client side interface centralized almost all of the reservoir simulation tools which include tools from drawing the domain to viewing simulation results. By using the socket technique, a robust data communicating protocol has been established between the server side interface and the client side interface. Through specific port numbers, the server can identify the working orders from the clients and run jobs in specified high performance computers. Also, the server side interface has the capability of detecting and transferring the newest simulation results. The source code is modularized, which makes it easy to maintain. Other simulators can potentially be plugged into the interface.

10.3 Installation

10.3.1 Server

The server package includes at least four parts:

1. Java source codes and executable Java server side interface file: domServer.jar.

2. Domain mesh software package.

3. Finite Element reservoirs simulator and its image file generator.

4. Linux/Unix environment file.

System administrator needs to copy all of the components mentioned above into one directory in the host machine which is under Linux/Unix environments. Make sure the Sun JRE 5.0 and higher version have been installed on that machine and the path has already been set.

Continue with the following steps:

1. Type "mpdboot -f mpdhost" in the command line to start mpd.

2. Type "java -jar domServer.jar &" in the Linux/Unix command line to start server side interface layer. The server side interface layer should be kept open to listen to the client request.

10.3.2 Client

The client side interface is mainly composed of two categories:

1. Java client side source code and executable Java client side interface file.

2. A template file for new domain generation.

End users need to copy the files mentioned above into one directory of the local machine. Any operating system can be used. However, the Sun JRE 5.0 and higher versions are required at the local machine. The examples of how to run client side reservoir simulation interface is described beow.

For Windows operating system, either find and double click the executable jar file, UFESsimulator.jar or in the command window, locate directory and type "java -jar UFESsimulator.jar &". For Linux/Unix operating system, in the command window's command line, type: "java -jar UFESsimulator.jar &".

10.3.3 Environments Requirement

For Windows operating system end users, after the JRE installation, the following steps can help the end user to set up the access path of JRE.

1. Click on the Start —— Control Panel —— System ——- Advanced —— Environment Variables —— System variables:

2. Variable name: CLASSPATH

   Variable value: "C:/Program Files/Java/jdk1.5.0_01/lib/tools.jar"

3. Variable name: JAVA_HOME

   Variable value: "C:/Program Files/Java/jdk1.5.0_01"

4. Variable name: path

   Variable value: "C:/Program Files/Java/jdk1.5.0_01/bin"

10.4 Client Side Reservoir Domain Data Input

10.4.1 Start from a new domain

The examples shown in this chapter will be for the Windows XP operating systems. When end user opens the client side interface, the picture should be looks like: 10.1

The upper left window is the main reservoir simulation client side interface window. In this window, end user can start and finish the reservoir simulation. The default simulation status is starting from a brand new domain.

The lower left window is the system input/output console window. In this window, end user can monitor the input and output information for the entire simulation process which includes the domain drawing information, data input information, client/server connecting information, file and command data transferring information, and so on. This console panel is designed both for the end users and the interface programmers. By monitoring the whole simulation process, end users can monitor the status of the simulation while the programmers can easily verify if information added is correct or not.

For the new reservoir domain simulation, drawing the reservoir domain which includes the enclosed domain boundaries, main fracture coordinates, injection well locations and production wells positions, comes first. Enclosed domain boundaries should be drawn first. An example of how to draw a simple five-point enclosed boundary is shown in Figure 10.2. The first point is marked as "0" and the fifth point is "4". The activity of drawing the domain is completed by double clicking.

Once the domain boundaries are drawn, the user can draw the main fractures, injection wells and production wells. There is no specific order in which these need to be drawn. Example is shown in Figure 10.3. In this reservoir simulation interface, domain boundary is modeled by polygon, fracture is modeled by line, injection well is represented by round circle and production well is shown by a square. In the picture shown (10.3), two fractures and one injection and one production well are drawn. In the lower/left input-output consoles, the absolute coordinates of the points drawn appear. This is illustrated in Figure 10.4.

As an simple example, lets save the above reservoir domain information into the "*_ori.xml" format. From the "File" menu of the main reservoir simulator interface, click "Save to *_ori.xml" option, then input the full file name on the "File Name" option of the popped up file chooser window. Here, the name "simpleExample1_ori.xml" as the input reservoir domain file name was provided as the "File Name" part of the popped up window. This is shown in Figure 10.5.

10.4.2 Start from an existing domain

If the user wants to start from an existing, saved file, this file can be opened by starting with the "File" menu. An example of how to open the previously saved "simpleExample" is shown in Figures 10.6 and 10.7. At this stage, new features

may be added, if desired. This modified file must be saved using tools described previously and as shown in Figure 10.8. Based on the old file, a few fractures, injection wells and production wells have been added to the reservoir domain and saved as "simpleExample2_ori.xml". This exercise basically shows the domain modification capability of the program.

10.4.3 Reservoir domain coordinates modifications

The coorodiantes themselves can be modified by opening the file and modifying them directly using the edior. This concept is shown in Figure 10.9. As shown in the figure, for the domain file "simpleExample2_ori.xml", there are 30 vertex numbers and 12 line relations as follows:

```
#A face with 31 points in 2D
31 2 0 1
#Vertex
0 386.0 902.0 1
1 692.0 1052.0 1
2 1098.0 866.0 1
3 1150.0 424.0 1
4 428.0 294.0 1
5 598.0 864.0 2
6 874.0 606.0 2
7 818.0 760.0 2
8 574.0 710.0 2
9 466.0 838.0 3
10 852.0 700.0 4
11 514.0 380.0 4
12 468.0 488.0 2
13 624.0 398.0 2
14 580.0 492.0 3
15 960.8166666666666 477.18333333333334 3
16 1005.0333333333333 511.925 3
17 1011.3499999999999 467.70833333333337 3
18 840.8 924.0875 4
19 779.2125 903.5583333333333 4
20 821.85 879.8708333333333 4
21 750.7875 597.2 2
22 798.1625 431.38750000000005 2
23 806.0583333333333 538.7708333333334 2
24 741.3125 469.2875 2
25 788.6875 551.4041666666667 2
26 719.2041666666667 537.1916666666667 2
27 1001.875 693.5291666666667 2
28 1098.2041666666667 622.4666666666667 2
```

29 1095.0458333333333 723.5333333333333 2
30 986.0833333333333 584.5666666666667 2
# of segments, each with a boundary marker
13 1
0 0 1 1
1 1 2 1
2 2 3 1
3 3 4 1
4 4 0 1
5 5 6 2
6 7 8 2
7 12 13 2
8 21 22 2
9 23 24 2
10 25 26 2
11 27 28 2
12 29 30 2

If the user wants to change the locations of the domain boundaries, fractures, injection wells or production wells, the only part that needs to be changed is in the #vertex section which contains the coordinates.

Example: The following modifications are proposed in the reservoir domain file "simpleExample2_ori.xml"

1. Add a new node between node 3 and node 4, the new node located at (818.0, 254.0).

2. Move production well at node 19 from current location to the new location at (1093.1, 679.9).

3. Move injection well at node 15 from current location to (621.0, 783.0)

4. Delete fracture 29 — 30

5. Save the modified file as the name of "simpleExample3_ori.xml"

Remembering that there are four different node (vertex) types (shown in the 4th column in the vertex section) being defined in the domain file:

1. Type 1 — domain boundaries

2. Type 2 — fractures

3. Type 3 — injection wells

4. Type 4 — production wells

Step-by-step description of how to achieve the proposed changes.

1. Initially, there are 31 (from 0 to 30) nodes in total. If one new node is added, the total vertex number should be 32 instead of 31 and the new vertex number should be 31, so, a line "31 818.0 254.0 1" should be added to the "#vertex" section. Then, in the section of "# of segments, each with a boundary marker", three changes are required.

   (a) Change "13 1" to "14 1"

   (b) Change line "3 3 4 1" into "3 3 31 1".

   (c) Add a new line of "13 31 4 1" at the end

2. In the "#vertex" section, modify "19 779.2125 903.5583333333333 4" to "19 1093.1 679.9 4".

3. In the "#vertex" section, modify "15 960.8166666666666 477.18333333333334 3" into "15 621.0 783.0 3".

4. In the "#vertex" section, delete the line of "29 1095.0458333333333 723.5333333333333 2" and "30 986.0833333333333 584.5666666666667 2".

   Then the total nodes should be 30 and change the vertex number 31 into 29 in all other places

   In the section of "# of segments, each with a boundary marker", make three changes:

   (a) Change "14 1" to "13 1"

   (b) Delete line "12 29 30 2"

   (c) Change line "13 29 4 1" into "12 29 4 1"

5. Save the modified file as the name of "simpleExample3_ori.xml", then the file is saved.

All the required modifications have been accomplished - the new table should like the one shown below.

```
#A face with 31 points in 2D
30 2 0 1
#Vertex
0 386.0 902.0 1
1 692.0 1052.0 1
2 1098.0 866.0 1
3 1150.0 424.0 1
4 428.0 294.0 1
5 598.0 864.0 2
6 874.0 606.0 2
7 818.0 760.0 2
```

8 574.0 710.0 2
9 466.0 838.0 3
10 852.0 700.0 4
11 514.0 380.0 4
12 468.0 488.0 2
13 624.0 398.0 2
14 580.0 492.0 3
15 621.0 783.0 3
16 1005.0333333333333 511.925 3
17 1011.3499999999999 467.70833333333337 3
18 840.8 924.0875 4
19 1093.1 679.9 4
20 821.85 879.8708333333333 4
21 750.7875 597.2 2
22 798.1625 431.38750000000005 2
23 806.0583333333333 538.7708333333334 2
24 741.3125 469.2875 2
25 788.6875 551.4041666666667 2
26 719.2041666666667 537.1916666666667 2
27 1001.875 693.5291666666667 2
28 1098.2041666666667 622.4666666666667 2
29 818.0 254.0 1
# of segments, each with a boundary marker
13 1
0 0 1 1
1 1 2 1
2 2 3 1
3 3 29 1
4 4 0 1
5 5 6 2
6 7 8 2
7 12 13 2
8 21 22 2
9 23 24 2
10 25 26 2
11 27 28 2
12 29 4 1

With these modifications, the domain should look like the one shown in Figure 10.10.

10.5 XML Input File

As the data input file in the reservoir simulation interface, XML format file includes several tags to markup the different functions. As mentioned in front

sections, there are two XML format files in total: "*_ori.xml" is used for domain generation and as the input file to be meshed by server; "*_fin.xml" will be automatically generated after the "*_ori.xml" is successfully meshed, and "*_fin.xml" is the final simulation data input file. The only different sections between these two files is the last part of the file: "*_ori.xml" is the rough domain information and "*_fin.xml" is the fine meshed file.

10.5.1 "*_ori.xml" file (rough domain file)

Since XML file is a markup language, there are a number of tags in this file. The basic structure is shown and explained as follows (the explanation is after the "—" sign).

```
<?xml version="1.0"?>  --- Required line, no modification necessary


<simulation name="test"> --- The simulation section start.

<domain name = "zero">  --- The domain section start.

<permeability>  --- The permeability section start.
   --- The content of permeability, i.e.: 0    50   0 100
</permeability> --- The permeability section end.

<porosity> --- The porosity section start.
   --- The content of porosity, i.e.: 0 0.1
</porosity>  --- The end of porosity section.

<Sw>  --- The solubility of water section start.
   --- The content of water solubility. i.e.: 0 0.22
</Sw>  --- The solubility of water section end.

<Po>  --- The oil pressure  section start.
   --- The content of oil pressure. i.e.: 0 1000
</Po>  --- The oil pressure section end.

<fluid-type> --- The fluid-type section start.
   --- The content of fluid-type. i.e.: 0 PVT-10
</fluid-type> --- The end of fluid-type section.

<rock-type> --- The start of rock-type section.
   --- The content of rock-type.
</rock-type> --- The end of rock-type section.
```

```
<fluid-table name = "PVT-10"> --- The start of fluid-table section.
  <oil-density unit="API">  --- Start of oil-density section.
   --- Contents of oil-density. i.e.: 30
  </oil-density> --- End of oil-density section.

  <water-density unit="density"> --- Start of water-density section.
   --- Contents of water-density. i.e.: 63
  </water-density>

  <oil type = "function"> --- start of oil section.
   --- Contents of oil. i.e.: 14.7   1        1.e-5  1.0 0.0
  </oil> --- end of oil section.

  <water type = "function"> --- start of water section.
   --- Contents of water. i.e.:  14.7   1        1.e-7  1.0  0.0
  </water> --- end of water section

</fluid-table> --- end of fluid-table section.

<rock-table name = "kr-10"> --- start of rock-table section.

  <oil-water> --- start of oil-water section.
    --- Contents of oil-water. i.e.:
    0.22 0.00 1.0000 7.0
    0.30 0.07 0.4000 4.0
    0.40 0.15 0.1250 3.0
    0.50 0.24 0.0649 2.5
    0.60 0.33 0.0048 2.0
    0.80 0.65 0.0000 1.0
  </oil-water> --- end of oil-water section

  <rock> --- Start of rock section.
   --- Contents of rock. i.e.: 14.7 0
  </rock> --- End of rock section.

</rock-table> --- End of rock-table section.

<well name = "I1" fluid = "PVT-10"> --- Start of well section. ``I1'' means it is

  <tube> --- Start of tube section.
    --- Contents of well. i.e.: 0 49 49
    <!-- the vertex number of current injector well -->
```

```
  </tube> --- End of tube section.

  <diameter unit="in">---Start of well diameter section.
   --- Contents of well diameter. i.e.: 0 3
  </diameter> --- End of well diameter section

</well> --- End of well section

<event time = "0"> --- Initialization of the simulation.

  <well name="I1"> --- start of first injection well initialization.
    --- contents of the first injection well status.
    --- i.e.: open injection bhp 1100 0
  </well> --- End of the first injection well.

  <well name="P1"> --- start of first production well initialization.
    --- contents of first production well status.
    --- i.e.: open production bhp 800 0
  </well>

  <boundary> --- start of domain boundaries initialization section.
      --- contents of the boundaries initialization.
      --- i.e.: 0 NoFlow
  </boundary> --- end of domain boundaries initialization section.

</event> --- End of the initialization of the simulation.

<event time = "10"> --- start of the specified time data output.
  <vtk file="10.vtk"> --- require to output format.
    SW
  </vtk>
</event> --- end of the output section on specified date.


<poly> --- Start of rough domain shape information section.
  --- contents of the domain rough information. There are four sub sections in thi
</poly> --- End of the rough domain shape information section.

</domain> --- End of the domain section from the very beginning.

</simulation> --- End of the simulation input file (*\_ori.xml).
```

The rough domain shape section in the above paragraphs are made of four sub-sections. Here, as an example, the ¡poly¿ section in the file of "simpleExample1_ori.xml" has been extracted and shown.

```
<poly>

simpleExample1_ori.xml  --- Name of the rough domain.
#A face with 15 points in 2D --- There are 15 nodes in the whole domain.

15 2 0 1 --- <numbers of vertexes> <dimension, must be 2 here>
--- <number of attributes> <boundary marker (0 or 1),0 means no, 1 means yes>
#Vertex  --- <vertex number> <x> <y> <vertex marker>
0    386.0    902.0   1 ---(386.0, 902.0)is a boundary vertex
1    692.0    1052.0   1
2    1098.0    866.0   1
3    1150.0    424.0   1
4    428.0    294.0   1
5    598.0    864.0   2 --- (598.0,864.0) is a fracture vertex
6    874.0    606.0   2
7    818.0    760.0   2
8    574.0    710.0   2
9    466.0    838.0   3 --- (466.0,838.0) is a injection well
10    852.0    700.0   4
11    514.0    380.0   4 --- (514.0,380.0) is a production well
12    468.0    488.0   2
13    624.0    398.0   2
14    580.0    492.0   3 --- vertex 14 is a injection well to

# of segments, each with a boundary marker
8 1 --- <number of line relations> <boundary marker (0 or 1)>
    --- <line relation number> <start point> <end point> <relation marker>
0    0    1    1 --- first domain boundary segment is from vertex ``0'' to ``1''
1    1    2    1
2    2    3    1
3    3    4    1
4    4    0    1
5    5    6    2 --- vertex  is a fracture segment
6    7    8    2
7    12    13    2

# hole section --- <number of holes>

0 --- no holes (if it is not 0, the following line should be
  --- <hole number> <x> <y>
```

```
# region section --- optional line

</poly>
```

The relative snapshot of the file "simpleExample1_ori.xml" is shown as Figure 10.11.

Generally, the ¡poly¿ section will be generated by the interface drawing canvas automatically. Care must be exercised in modifying this section.

10.5.2 CVFE Simulation XML input file ("*_fin.xml" file)

The only differences between the "*_ori.xml" file (rough domain file) and "*_fin.xml" file (final simulation data input file) are the last section of the XML file. In the "*_ori.xml" file, the last section is the ¡poly¿ section, the details have already been discussed in the first part of this section. The file "simpleExample1_fin.xml" generated after sending the domain through the mesher is shown as an example (Figure: 10.12)

Differences between "simpleExample1_ori.xml" and "simpleExample1_fin.xml" are discussed below.

```
<vertex>  --- start of vertex section

0 386.0 902.0 0
1 692.0 1052.0 0
...
707 720.9604192083833 599.305693897363 0
708 625.6670927916116 993.1597965224233 0

</vertex> --- end of vertex section

<element> --- start of triangle element section

0 596 597 203
1 543 545 19
...
1323 496 708 122  --- the 1323th triangle element is made by vertex numbers 496, 7
1324 103 708 691

</element> --- end of triangle element section

<boundary> --- this is just for the fracture segments
```

```
0 474 97
1 48 390
...
27 546 201 --- the 27th fracture segment is from vertex 546 to 201
28 550 42

</boundary>
```

The user can modify property and well information, if desired.

10.6 Operations

10.6.1 Write property informations into the XML file

The reservoir property section is located in the front of the XML file. It includes the following information.

```
1.permeability --- <permeability>
2.porosity --- <porosity
3.water solubility --- <Sw>
4.fluid-type --- <fluid-type>
5.rock-type --- <rock-type>
6.fluid-table --- <fluid-table>
(1) <oil-density>
(2) <water-density>
(3) <oil function>
(4) <water function>
7.rock-table --- <rock-table>
(1) <oil-water>
(2) <rock>
8.well information <<well name = "*" fluid = "*">
(1) <tube> (the vertex number of well)
(2) <diameter>
9.reservoir initialization <event time = "0">
(1) <well name="*"> (open/close, type(injection/production), control methods (bhp
(2) <boundary> (domain boundary properties)
10.output scenarios <event time = )
```

A default property section is automatically attached and the user can modify these properties. However, for well informations, reservoir initialization, and out-

put scenarios sections, additional modifications are required. An example of how to do this is shown below (Figure 10.13:

The picture shows that vertex numbers 9 and 14 are injection wells, vertex numbers 10 and 11 represent the production wells. Alternatiely, in the file of "simpleExample1_ori.xml", the well information can be accessed.

```
<well name = "I1" fluid = "PVT-10">
  <tube>
    0 9 9
  <diameter unit="in">
    0 3
  </diameter>
</well>

<well name = "I2" fluid = "PVT-10">
  <tube>
    0 14 14
  <diameter unit="in">
    0 3
  </diameter>
</well>

<well name = "P1" fluid = "PVT-10">
  <tube>
    0 10 10
  <diameter unit="in">
    0 3
  </diameter>
</well>

<well name = "P2" fluid = "PVT-10">
  <tube>
    0 11 11
  <diameter unit="in">
    0 3
  </diameter>
</well>
```

In the reservoir initialization section, the input should includes all of the wells and domain boundary information.

```
<event time = "0">

  <well name="I1">
    open injection bhp 1100 0
  </well>

  <well name="I2">
    open injection bhp 1100 0
  </well>

  <well name="P1">
    open production bhp 800 0
  </well>

  <well name="P2">
    open production bhp 800 0
  </well>

  <boundary>
    0 NoFlow
  </boundary>

</event>
```

The times for visualization of saturation data can also be controlled.

```
<event time = "10">
  <vtk file="10.vtk">
    SW
  </vtk>
</event>

<event time = "20">
  <vtk file="20.vtk">
    SW
  </vtk>
</event>

<event time = "30">
  <vtk file="30.vtk">
    SW
  </vtk>
```

```
</event>

<event time = "40">
  <vtk file="40.vtk">
    SW
  </vtk>
</event>

<event time = "50">
  <vtk file="50.vtk">
    SW
  </vtk>
</event>
...
...
<event time = "580">
  <vtk file="580.vtk">
    SW
  </vtk>
</event>
```

Instead of the interface drawing canvas, from the ¡poly¿ section of the "*_ori.xml" file, user can also get the vertex numbers of the injection wells and the production wells. This is done by looking for index types 3 and 4.

```
<poly>
#simpleExample1\_ori.xml

#A face with 15 points in 2D
15 2 0 1
#Vertex
0    386.0    902.0   1
1    692.0    1052.0   1
2    1098.0    866.0   1
3    1150.0    424.0   1
4    428.0    294.0   1
5    598.0    864.0   2
6    874.0    606.0   2
7    818.0    760.0   2
8    574.0    710.0   2
9    466.0    838.0   3
10    852.0    700.0   4
```

```
11    514.0    380.0   4
12    468.0    488.0   2
13    624.0    398.0   2
14    580.0    492.0   3
```

10.6.2 Mesh the rough domain with reservoir features

Once the reservoir properties are adjusted, the domain needs to be meshed again and submitted to the server. Here, "simpleExample1_ori.xml" will be used as an example (10.14).

From the menu bar find "Mesh" and click on its submenu button "Connect to the Mesh Server and Mesh the current domain".

After this action, the interface will send the reservoir domain coordinates to the remote server to discretize the domain. The mesh result will be transferred back to the client side interface simultaneously. The meshed domain is drawn in a separate window, as shown in 10.15. The lower left window in 10.15 is the data input/output console window. All of the actions for above process have been monitored and recorded by this function.

10.6.3 Simulate the fine domain

Once the mesh is created, a new file "simpleExample1_fin.xml" was generated automatically. As the final simulation input data file, all of the mesh elements information and the reservoir properties are included in this file. The user may modify this file based on the instructions previously provided.

The simulation is begun by using the Calculate button on the menu bar. Refer to 10.16.The snapshot 10.16 is made in the middle of the simulation (110 days of simulation). The simulation results are transferred to the end user's local machine right after they have been generated. The process is monitored in the lower left window. Animation of the simualtion is shown in the upper right window.

After 580 days of simulation (it takes the server a couple of minutes), the final results of the simulation are displayed on a desktop snapshot 10.17. The blue color represents the water, while the redd color represents the oil phase. Water channels through the fractures as it arrives at the production wells, as shown.

10.7 Ancillary Programs

In this reservoir simulation interface package, some ancillary programs are provided. All of them are written by using independent Java class. During the processes such as domain generating, meshing, simulating, some of these functions are called by the interface automatically, some aren't. They can always open these programs anytime.

10.7.1 Triangle Mesh Viewer

This program is designed for the purpose of triangle mesh elements viewing. The name of "Triangle Mesh Shower V1.0" is given to this program. This window will be opened right after the domain action. This is mostly used for the cases where the domain is repeatedly modified and meshed. If end user closed the last mesh map accidentally, he/she can retrieve it by clicking the sub menu ofthe Mesh menu bar. The meshed results should be looks like the Figure 10.18: The image can be saved as a JPEG file by clicking on the left panel.

10.7.2 Results Images Viewer

User can either view the simulation results dynamically by using the animation window or by using this "Result Images Viewer". This is a picture view program in order to watch the result image separately. This function is located at the sub menu button of the Simulation results under the Calculation menu bar. The snapshot 10.19 shows three result pictures at days 10, 200 and 500 in the example simulation. Since they are opened in separate windows, user can compare them easily.

10.7.3 XML Source File Viewer

This program was written to view the XML input file. In the middle of the simulation, if the end user wanted to view the input XML data file, the file editors provided by the system or this program could be used. This is a read only file editor, and cannot be used to modify or edit the input XML file. For this reason, this is only for quick viewing of the input file and serves as a backup to other system editors. The program can be easily located as shown in Figure 10.20:

10.7.4 Interface Input/Output Console

The purpose of this conole is to serve as a monitor for this client-server mode reservoir simulation interface. By using this program, the user and the programmer can check the status of the simulation. All of the input/output actions between the user and the interface, the client interface and the server, are displayed by this control console. Since the whole interface package is based and executed on the Java Running Machine, if there is something wrong in the compiler, the error information is also displayed in this window. Furthermore, some pre-qualification conditions were built into the original Java code, so that, if, the user's input file had mistakes or didn't follow the right processes, system will provide the coded error information to the end user through this console. Example is shown in Figure 10.21.

There are two buttons located on the bottom of the input/output console; "Clear" button can remove all of the informations from current monitor window,

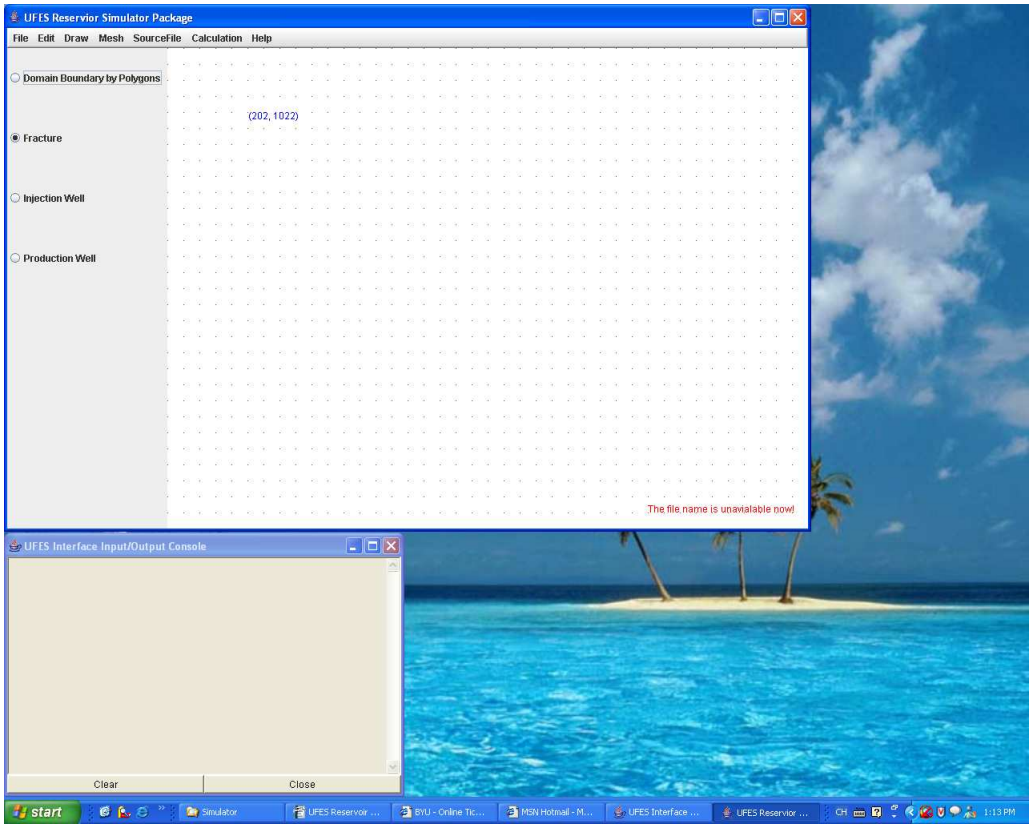

Figure 10.1: Open the Client Side Interface.

while the "Close" button works the same as upper right corner cross sign, for the console window to shut down.

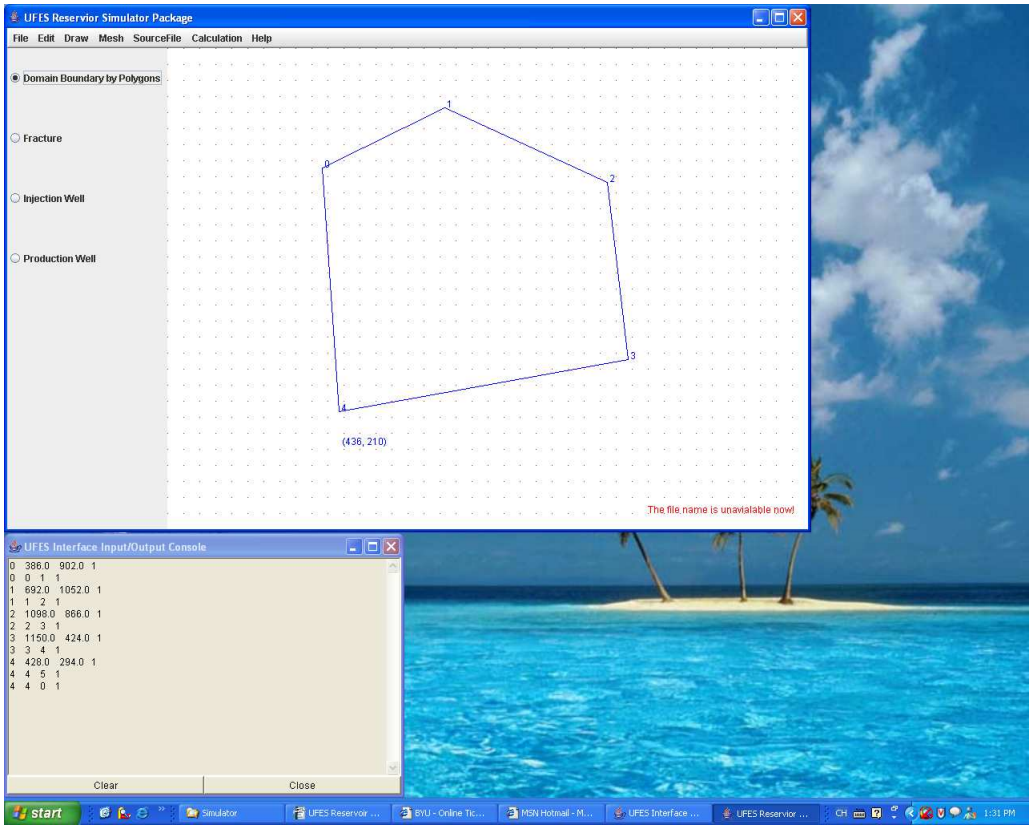

Figure 10.2: Draw a simple domain first.

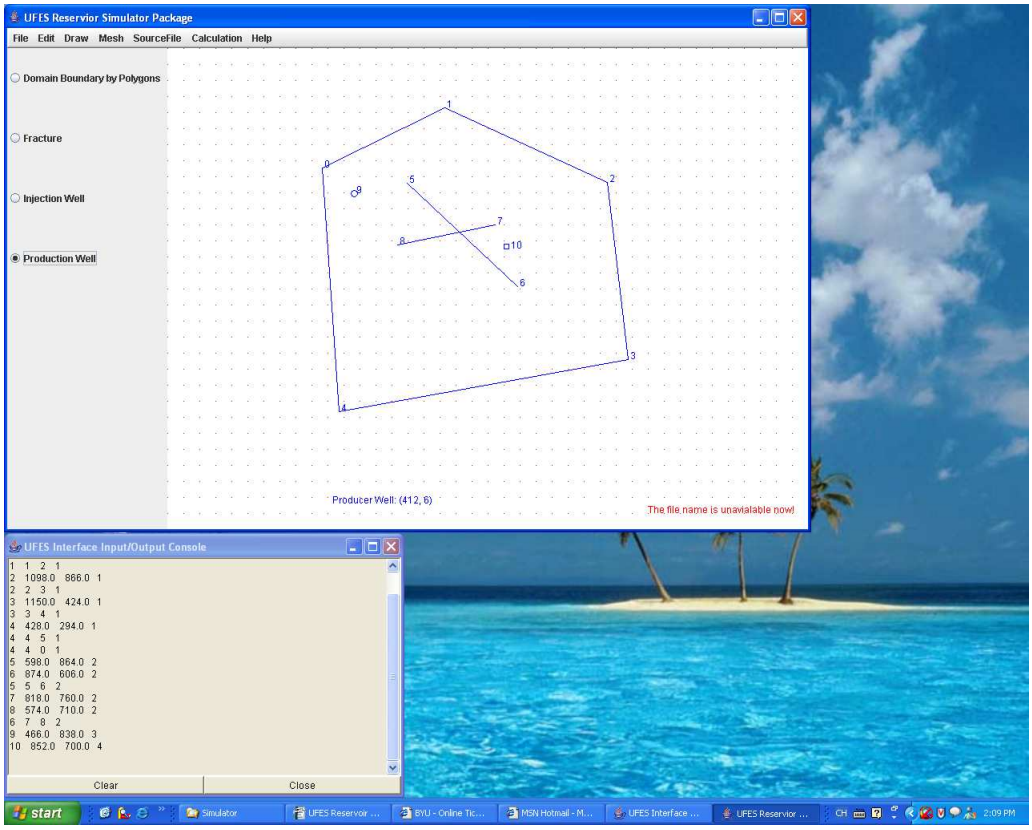

Figure 10.3: Draw fractures, injection wells and production wells.

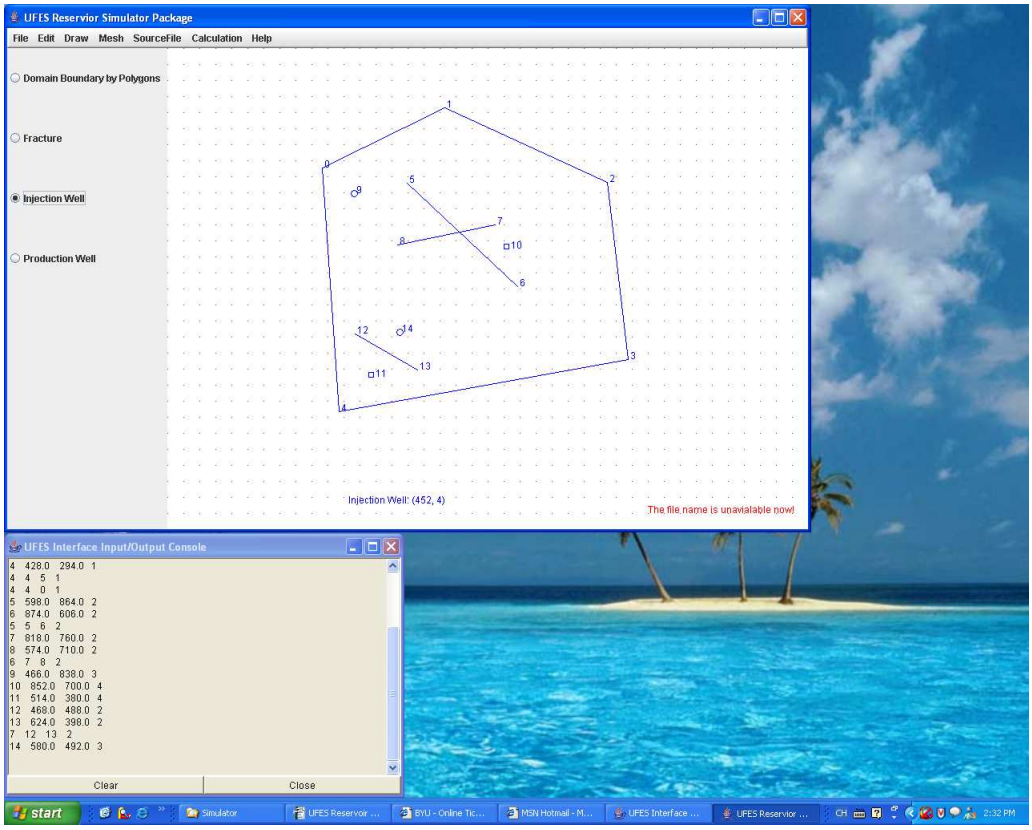

Figure 10.4: Draw additional fractures, injection wells and production wells.

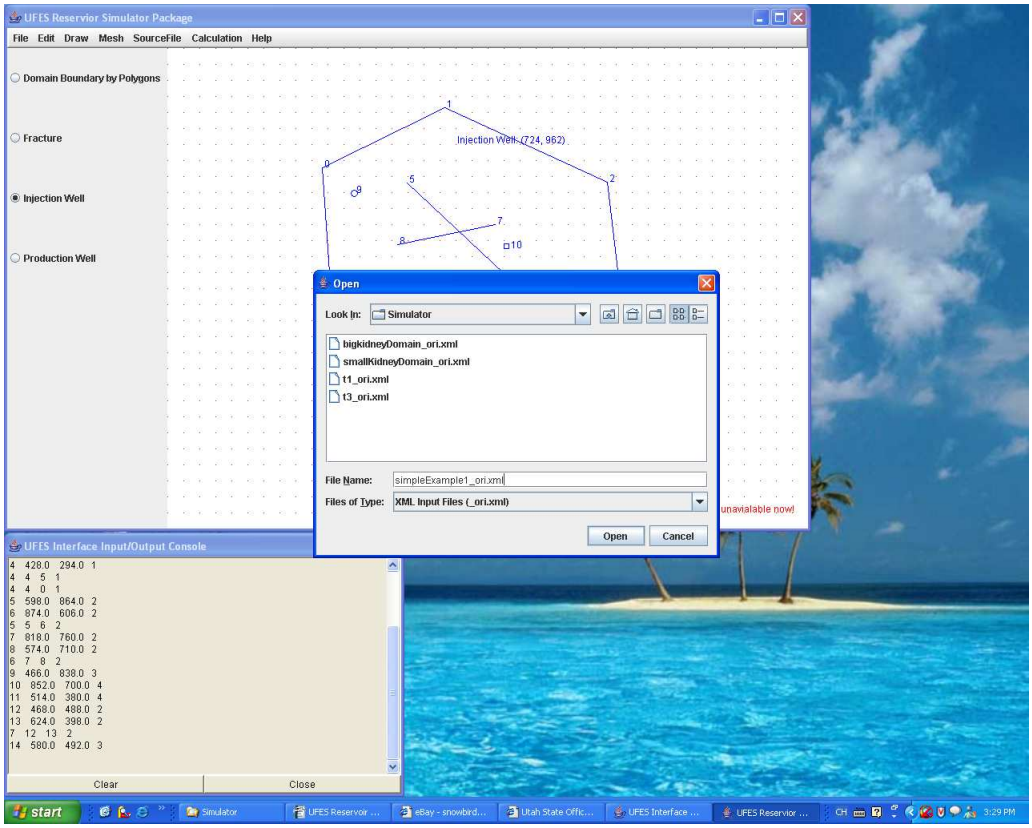

Figure 10.5: Save the reservoir domain information.

10.7.5 Some Defintions

Server: The high performance computing resource which is used for accepting work orders from end user (client) to complete computing jobs. In this work, server package is includes the executable Java server side interface class, triangle mesh software package and the finite element simulators. The whole server package is included in one directory located on the specified Linux/Unix machines.

Client: The end user. In this work, client package is composed by a series of Java classes source code and its executable file which is located on end user's local machines.

Java: A web based objective oriented computer language developed by Sun Inc. This computer language is running under Java Running Environments (JRE) instead of directly under the operating systems. In this reservoir simulator interface work, all of the computer code was written by Java language.

XML: A markup language for documents containing structured informations. In this reservoir simulator work, all of the data is default saved as XML format.

Java Socket: A network Input/Output communicating technique in Java. From the standpoint of clients: programs that open a socket to a server that's listening for connections. However, client sockets themselves aren't enough; clients

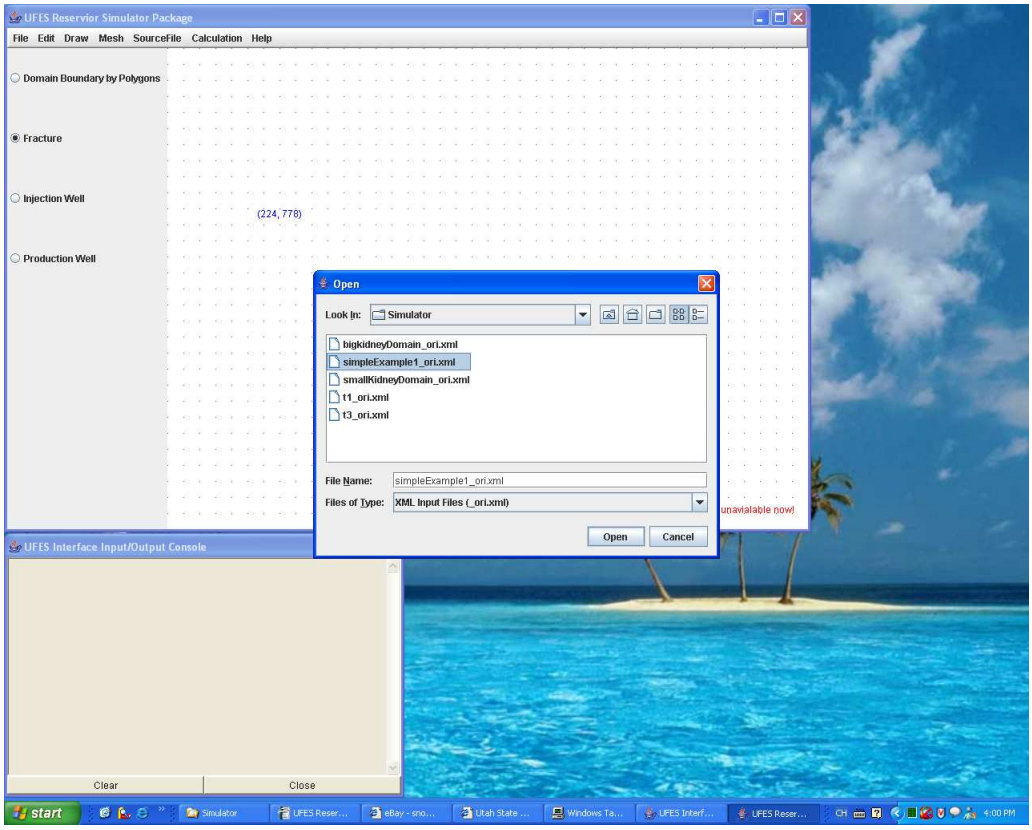

Figure 10.6: Open a save the reservoir domain.

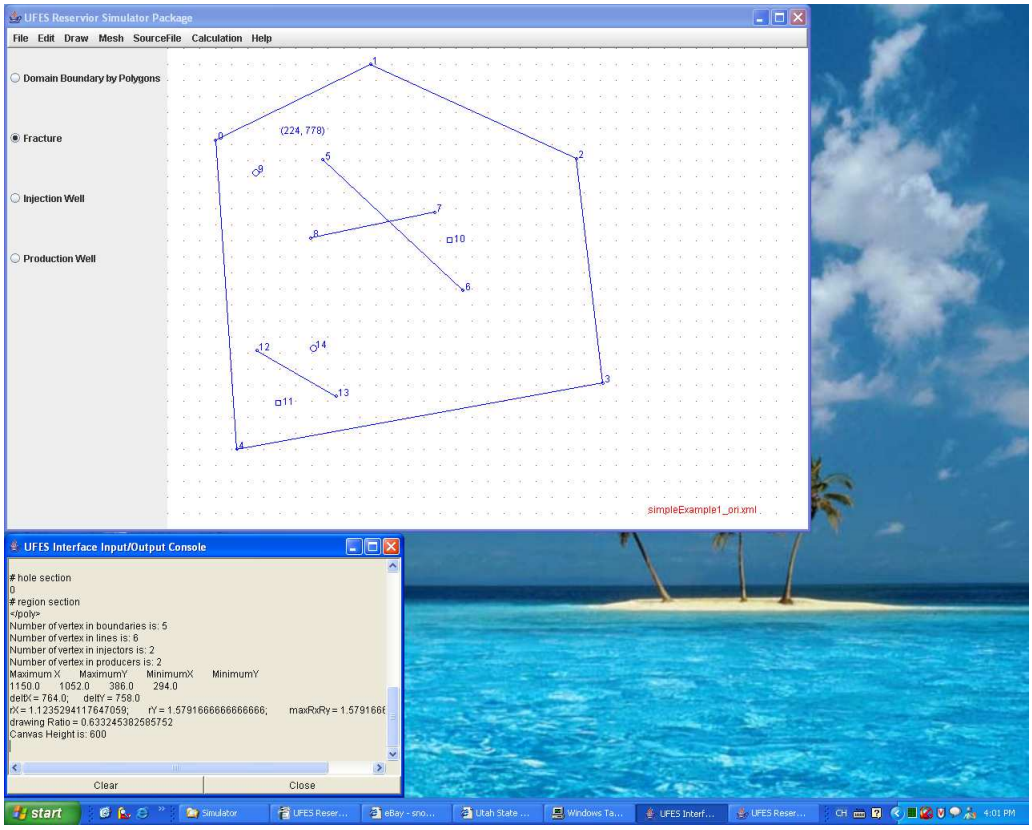

Figure 10.7: The opened reservoir domain.

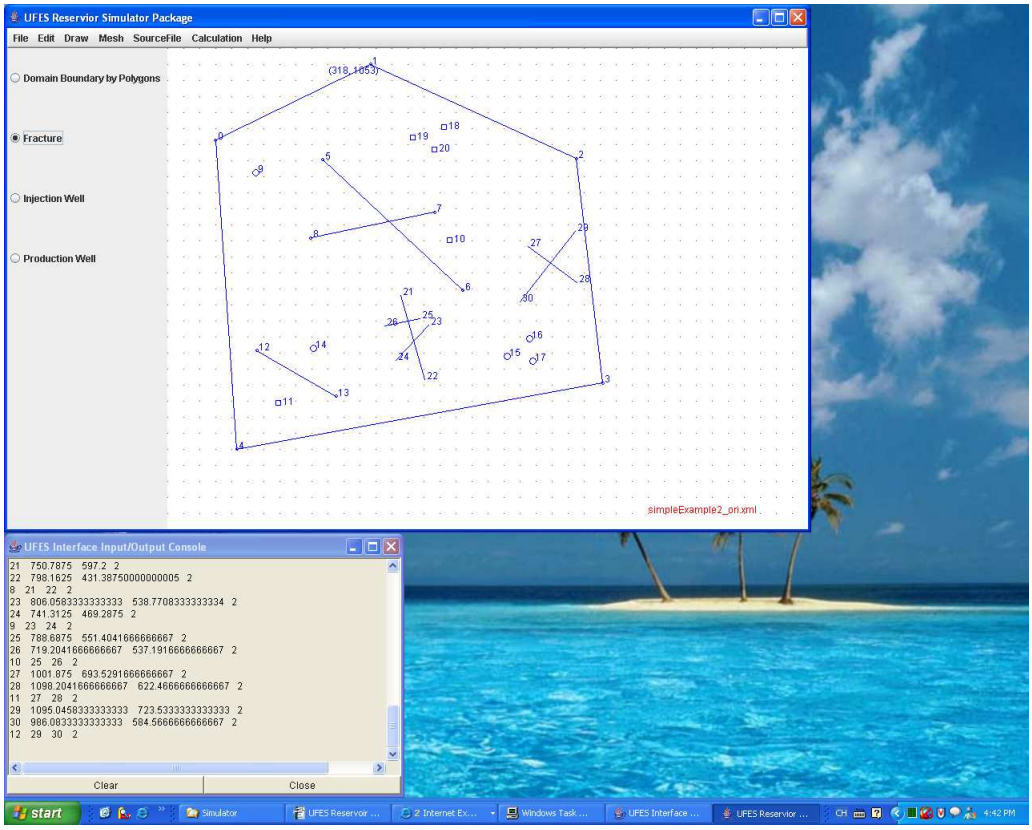

Figure 10.8: The modified reservoir domain.

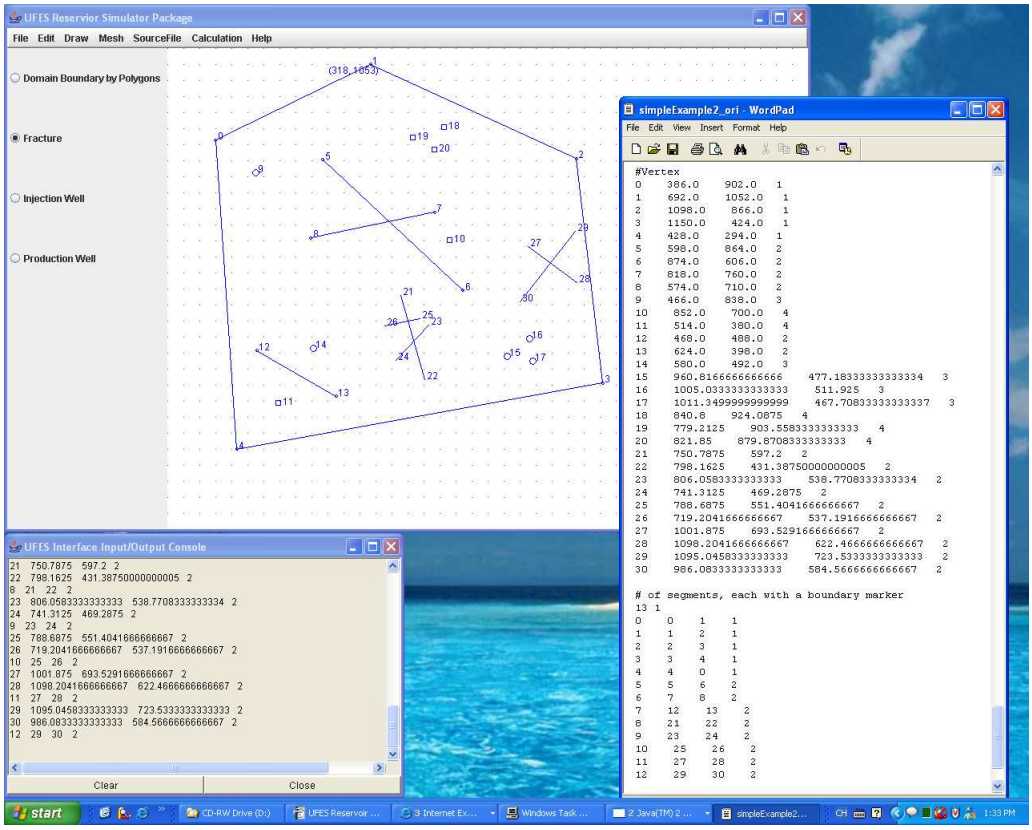

Figure 10.9: The *_ori.xml file.

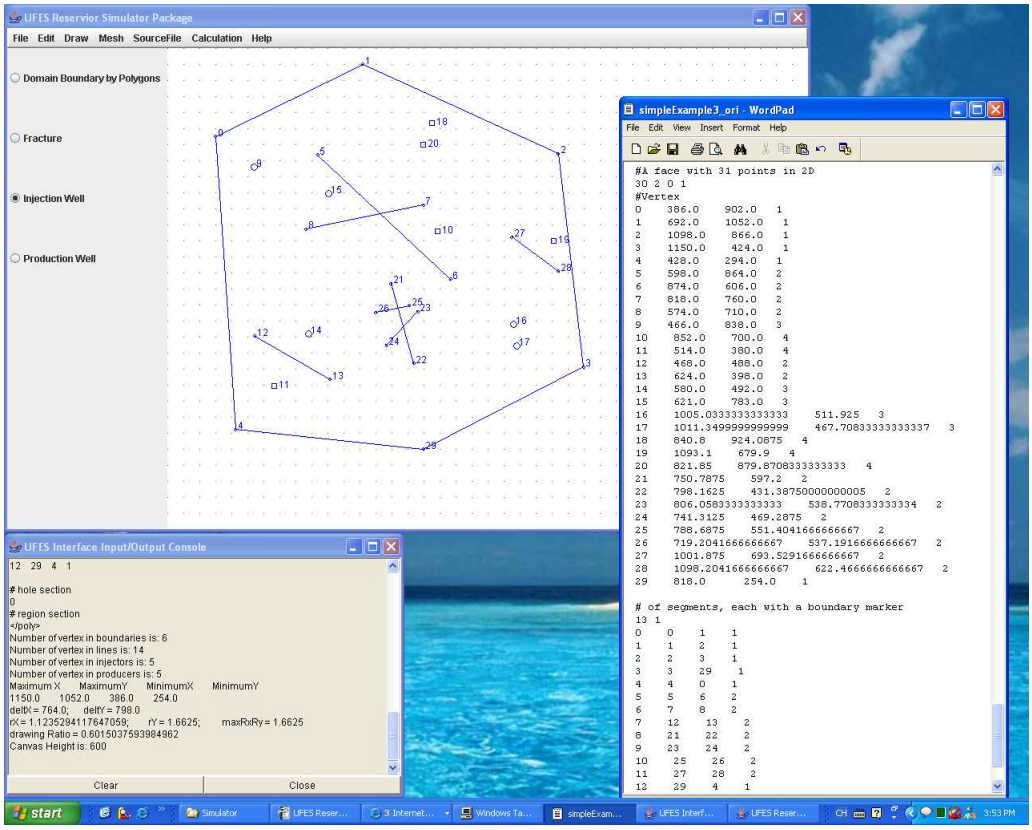

Figure 10.10: The modified *_ori.xml file.

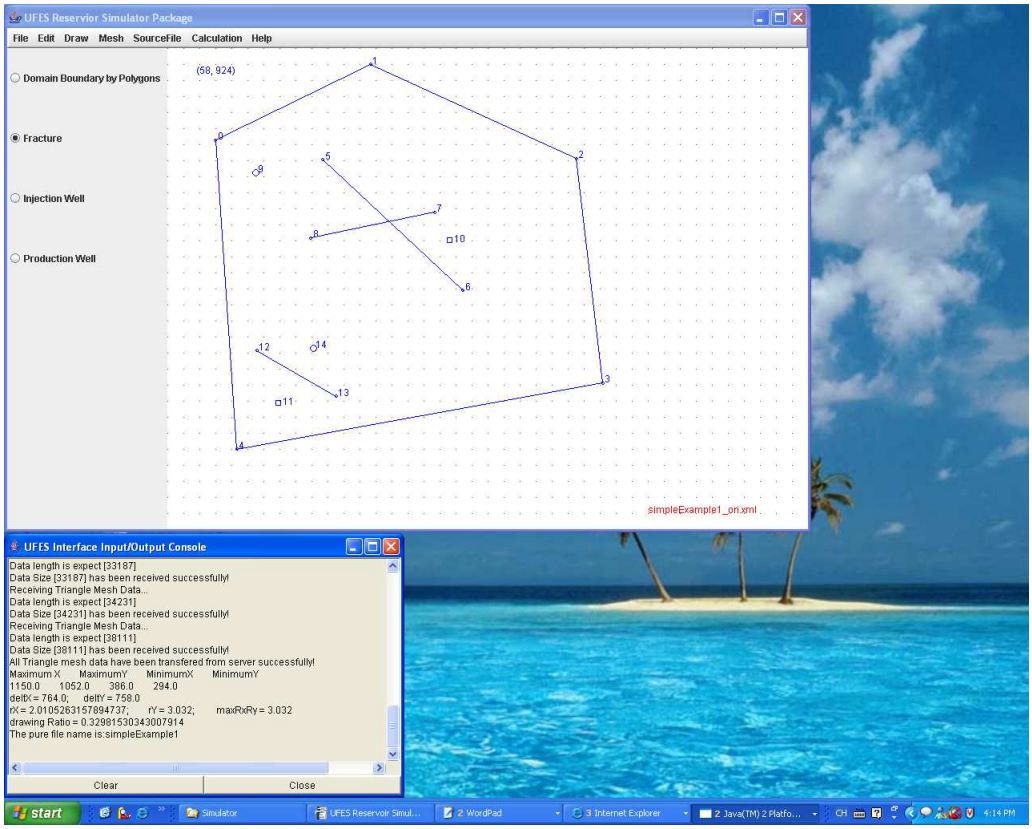

Figure 10.11: The snapshot of “simpleExample1_ori.xml”.

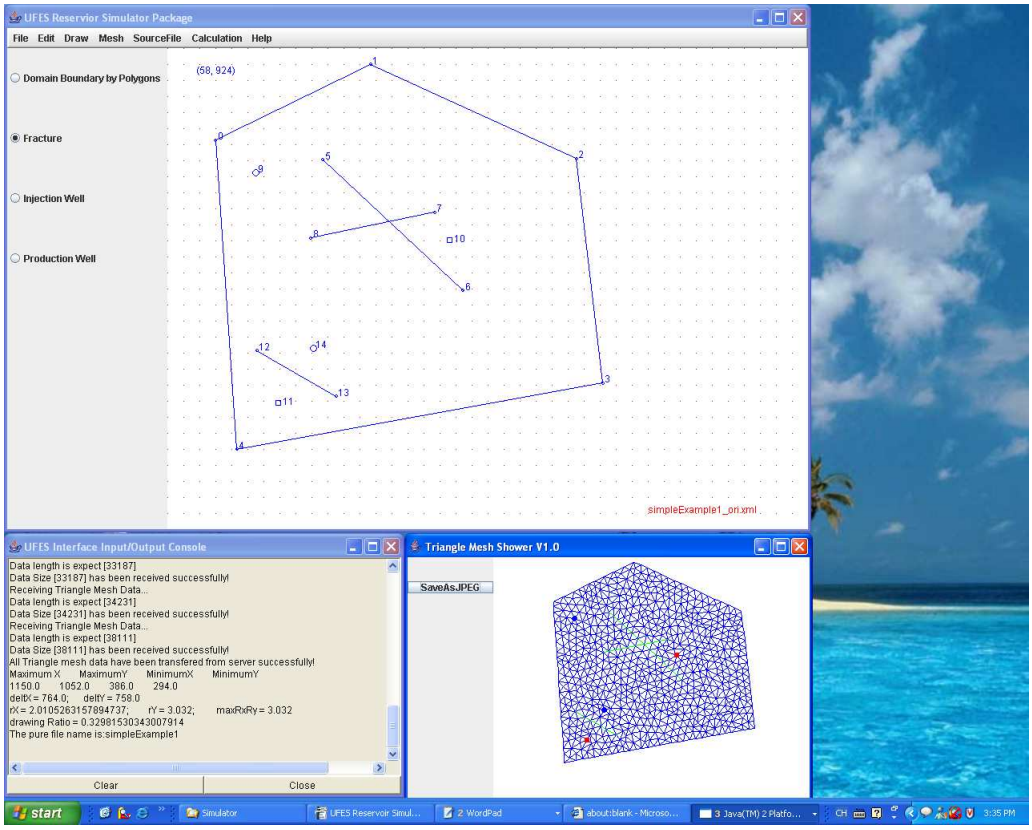

Figure 10.12: The snapshot of meshed "simpleExample1_ori.xml".

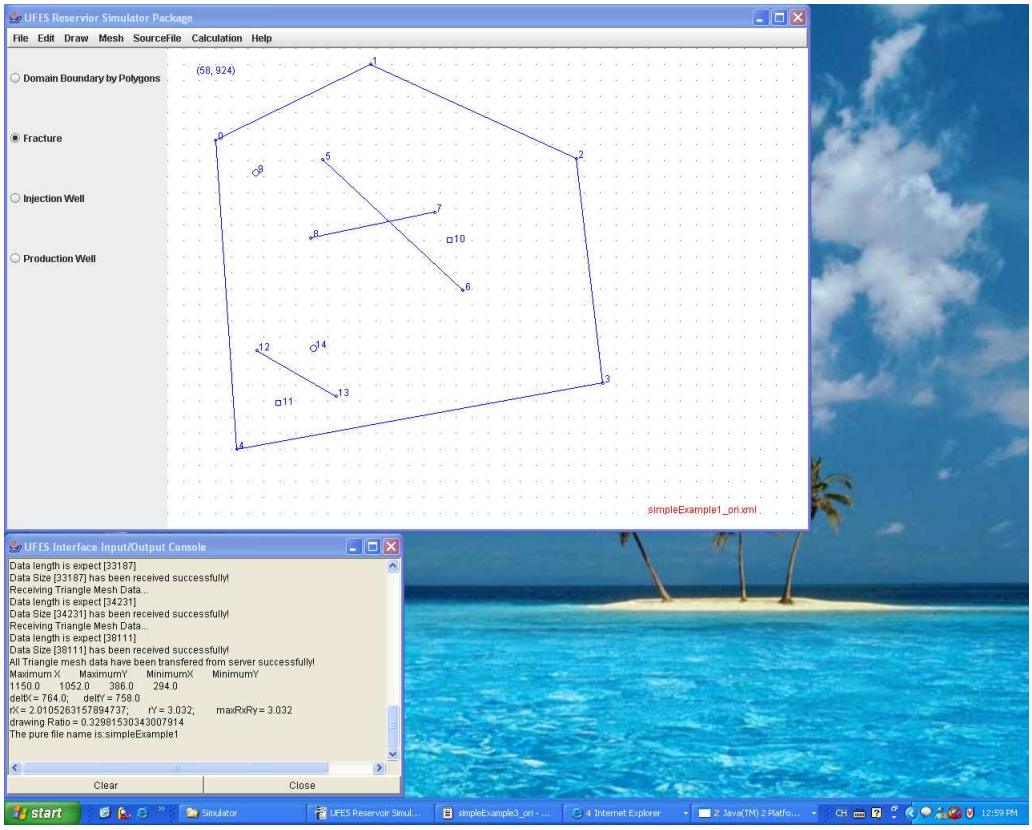

Figure 10.13: The snapshot of "simpleExample1_ori.xml".

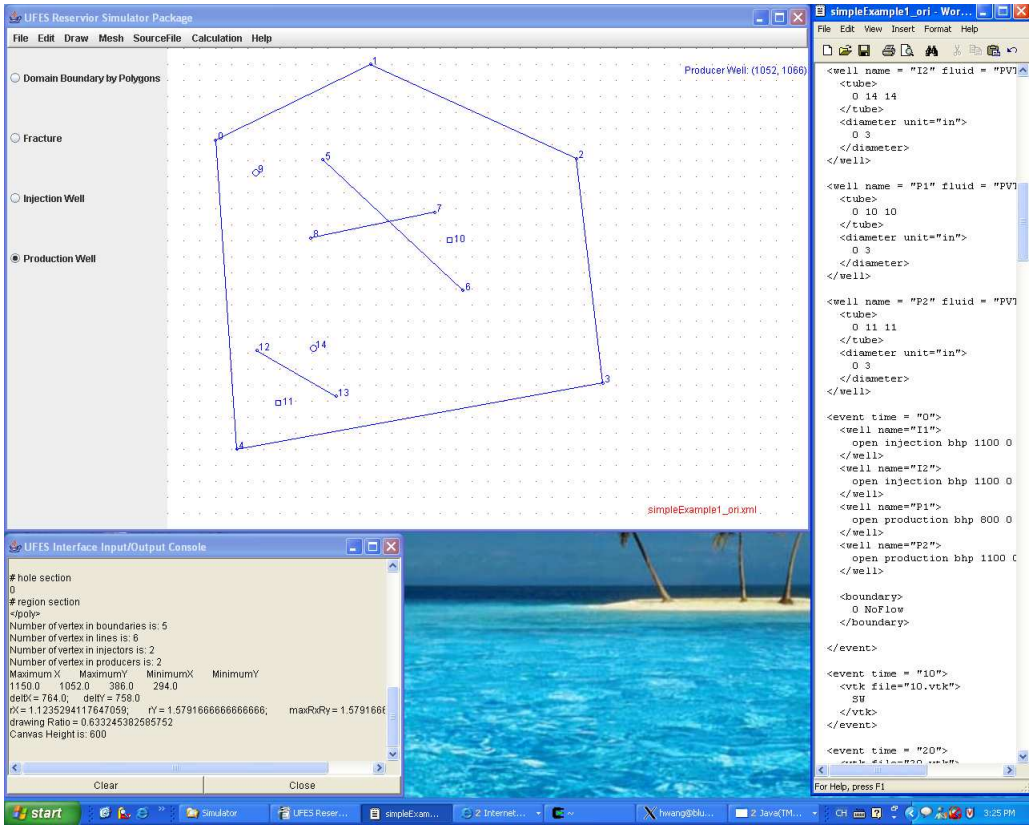

Figure 10.14: The snapshot of “simpleExample1_ori.xml” with its editor.

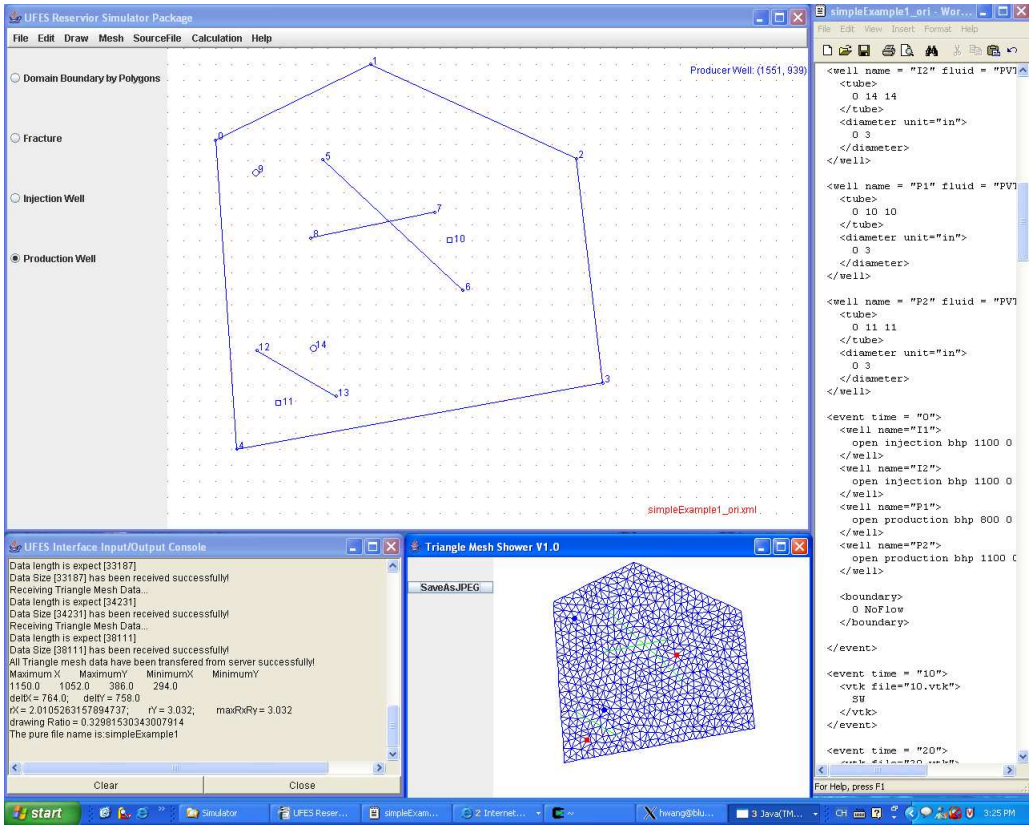

Figure 10.15: The snapshot of meshed "simpleExample1_ori.xml" with its editor.

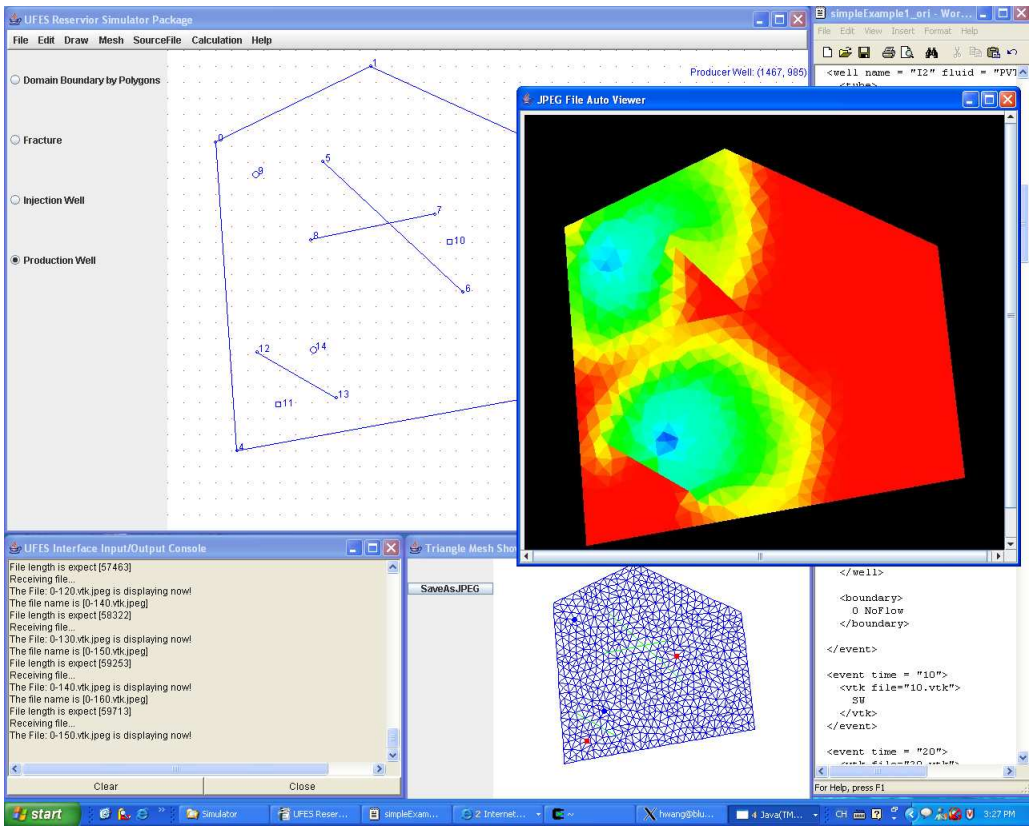

Figure 10.16: The middle snapshot of meshed "simpleExample1_ori.xml" simulation (110days).

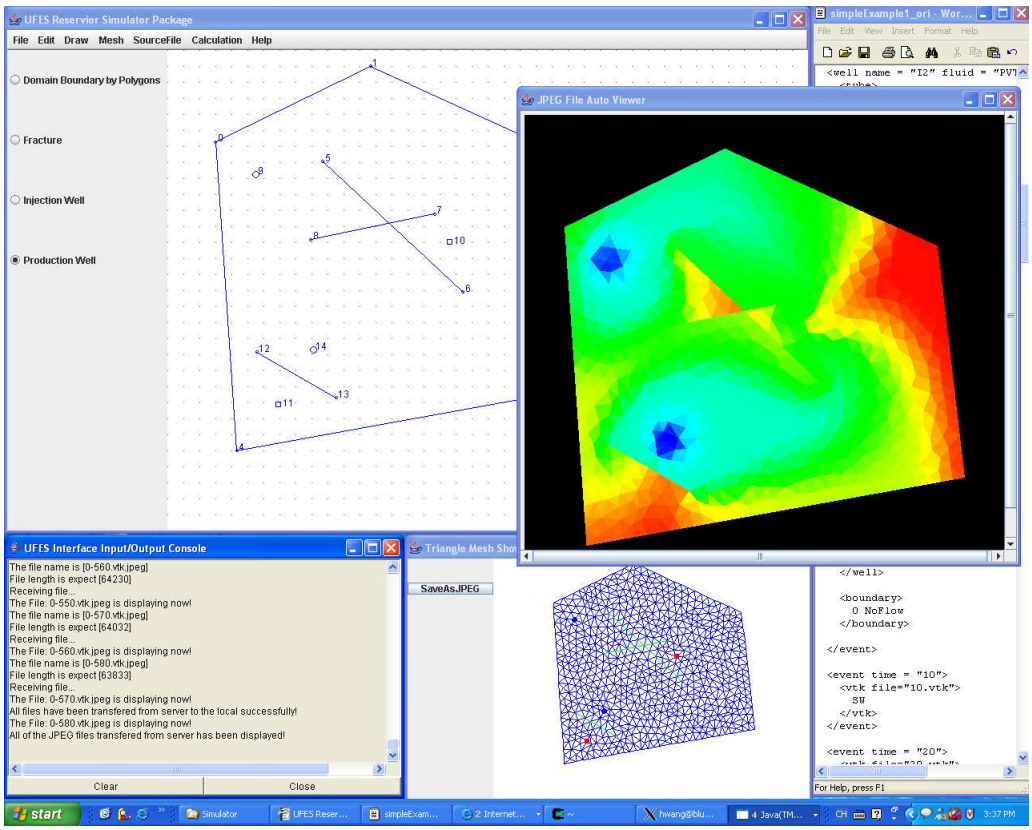

Figure 10.17: The final snapshot of meshed "simpleExample1_ori.xml" simulation (580days).

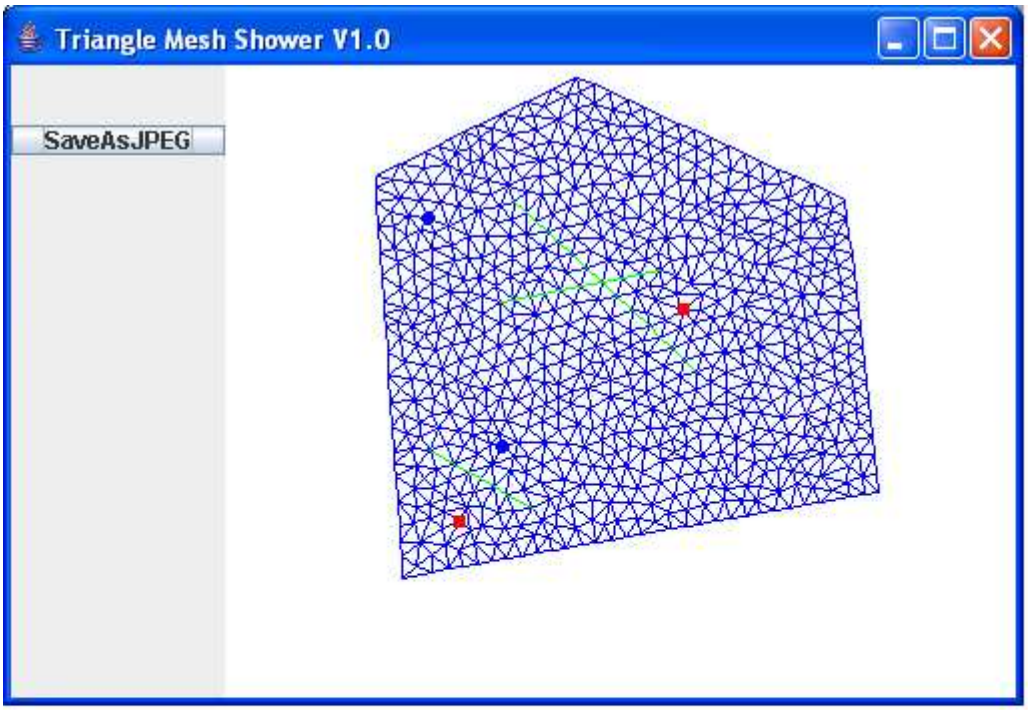

Figure 10.18: Triangle Mesh Shower V1.0.

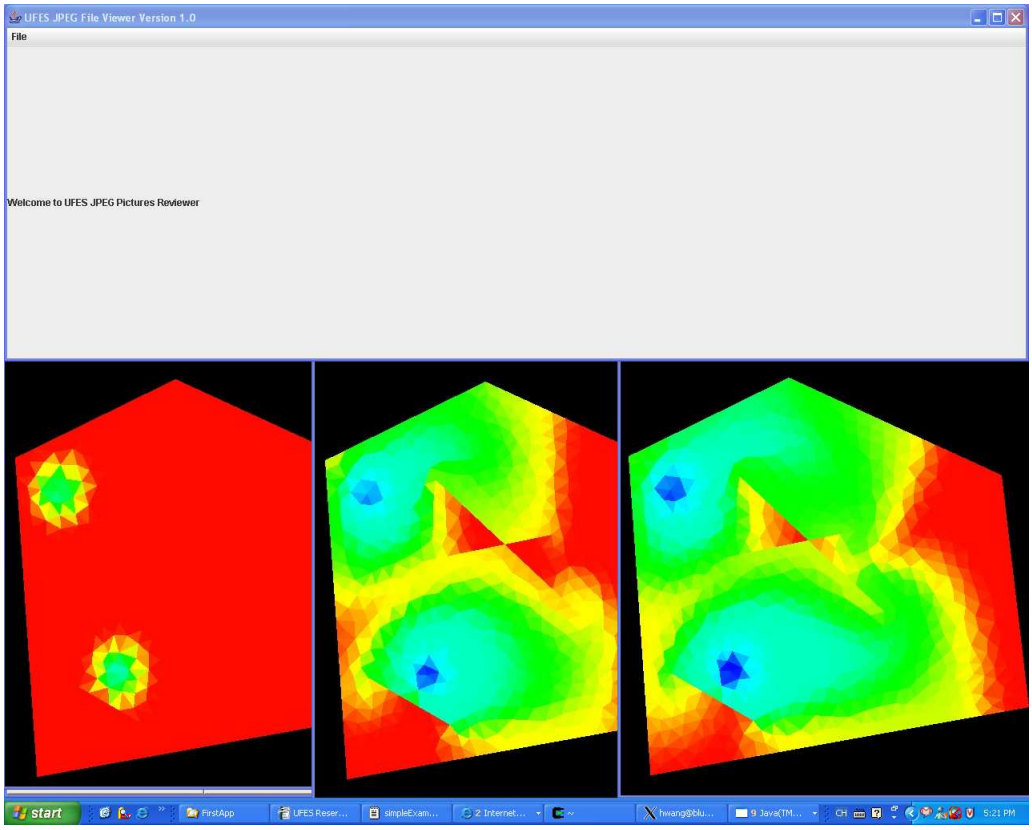

Figure 10.19: Result Images Viewer V1.0.

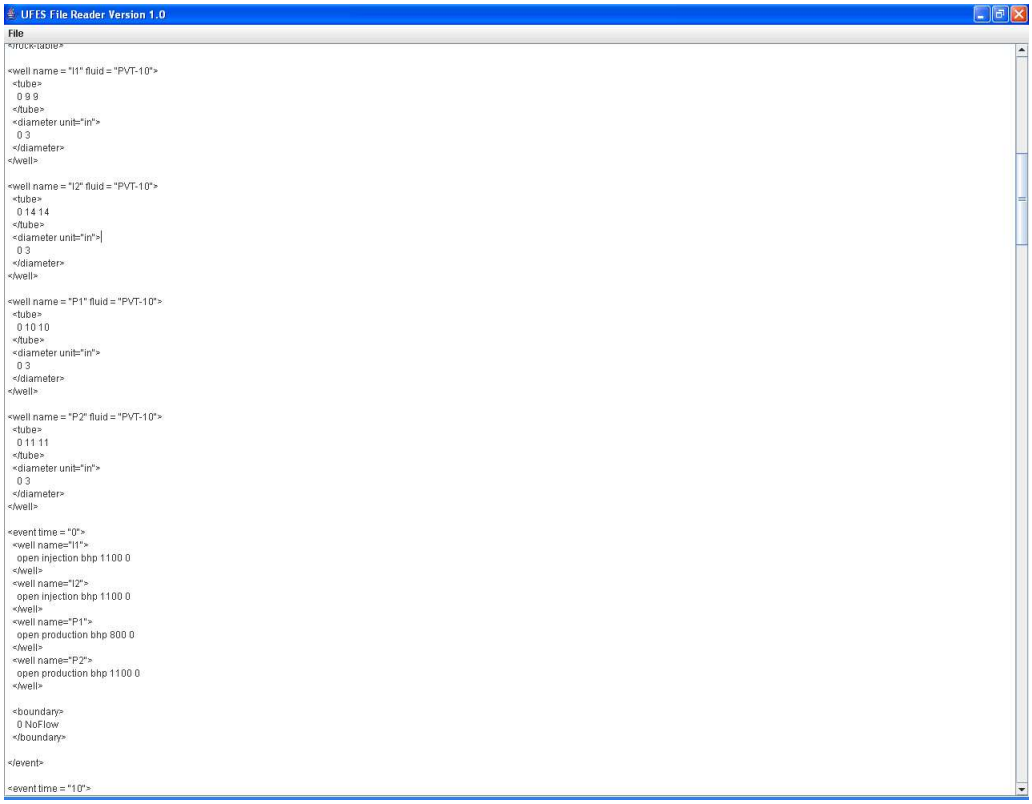

Figure 10.20: XML Source File Viewer V1.0.

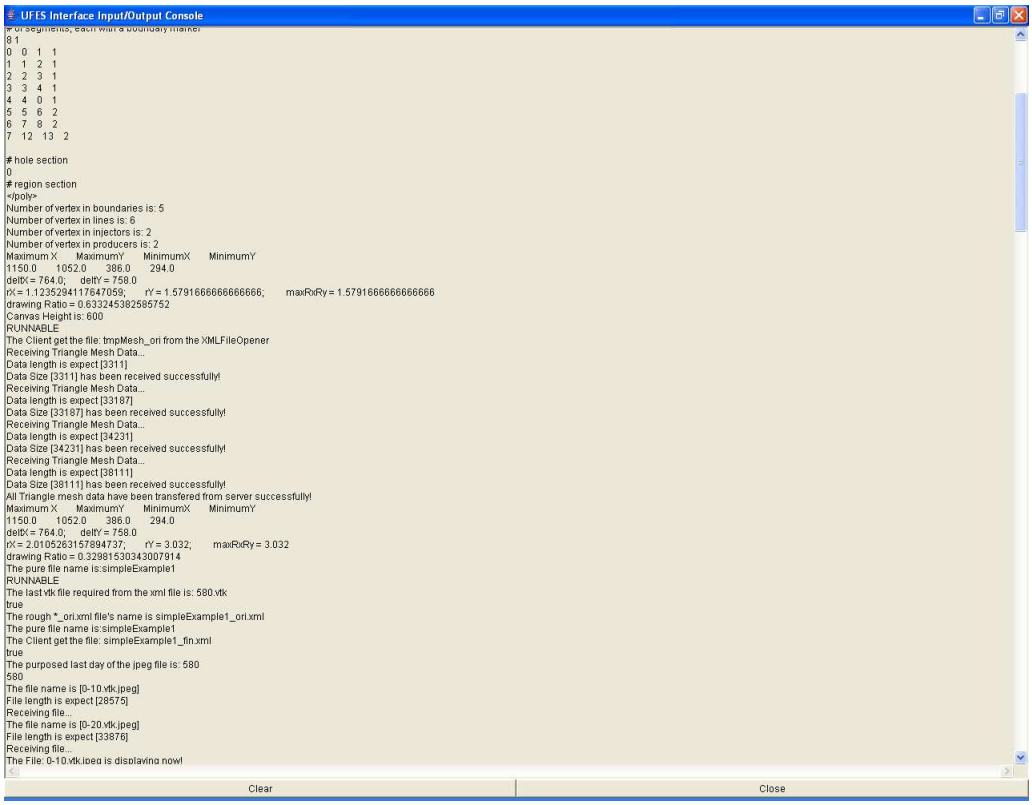

Figure 10.21: Interface Input/Output Console V1.0.

aren't much use unless they can talk to a server. There should be a server side socket opened to host the client socket by the same port number specified in both sockets. Here in this interface package, the server side socket is kept opened all the time listen for the client socket requesting. The client side socket will only be opened while end user want to submit jobs on the server through specified socket communicating protocol.

Java Swing: A graphic user interface (GUI) toolkit of Java. The main reservoir simulator interface frame is written by Java Swing such as all the panels, buttons, text fields, and so on.

Java 2-D Graphics: A two-dimensional graphic drawing toolkit of Java. In the current reservoir simulator interface, the reservoir domain drawing canvas was written by this technique. The closed reservoir domain is drawn by its polygons shape, fractures by its line shape, injection wells by its round shape and the production well by its square shape.

Java Threads: A very important topic for Java programming. For the current reservoir simulator interface developing, threads concepts are hired for the end user to run separate tasks at the same time within the same interface.

Bibliography

[1] Khalid Aziz and Antonin Settari, *Petroleum Reservoir Simulation*, Elsevier Applied Science Publishers, New York, NY, 1979.

[2] Jacob Bear, *Dynamics of Fluids in Porous Media*, Dover Publications, Inc., Mineola, New York, 1972.

[3] Peter H. Sammon, "An analysis of upstream differencing," *SPE Reservoir Eng.*, vol. 3, no. 3, pp. 1053–1056, 1988.

[4] Peter S. Huyakorn and George F. Pinder, *Computational Methods in Subsurface Flow*, Academic Press, Inc., London, England, 1983.

[5] Peter A. Forsyth, "Control-volume, finite-element method for local mesh refinement in thermal reservoir simulation," *SPE Reservoir Eng.*, vol. 5, no. 4, pp. 561–566, 1990.

[6] Iraj Javandel and P. A. Witherspoon, "Application of the finite element method to transient flow in porous media," *Soc. Petrol. Eng. J.*, vol. 8, pp. 241–251, 1968.

[7] C. L. McMichael and G. W. Thomas, "Reservoir Simulation by Galerkin's Method," in *46th SPE-AIME Annual Fall Meeting*, New Orleans, October 1971.

[8] Øyvind Langsrud, "Simulation of two-phase flow by finite element methods," in *Proceedings of Fourth Symposium of Numerical Simulation of Reservoir Performance of the Society of Petroleum Engineers of AIME*, Los Angeles, CA, 1976.

[9] Vilgeir Dalen, "Simplified finite-element models for reservoir flow problems," *Soc. Petrol. Eng. J.*, vol. 19, no. 5, pp. 333–343, 1979.

[10] M. Karimi-Fard and Abbas Firoozabadi, "Numerical simulation of water injection in 2d fractured media using discrete-fracture model," in *Proceedings of the 2001 SPE Annual Technical Conference and Exhibition*, New Orleans, LA, 2001.

[11] Jong Gyun Kim, *Advanced Techniques for Oil Reservoir Simulation: Discrete Fracture Model and Parallel Implementation*, Ph.D. thesis, University of Utah, Salt Lake City, UT, 1999.

[12] Jong Gyun Kim and Milind D. Deo, "Comparison of the performance of a discrete fracture multiphase model with those using conventional methods," in *Proceedings of the 1999 SPE Reservoir Simulation Symposium*, Houston, TX, 1999.

[13] Jong Gyun Kim and Milind D. Deo, "Finite element, discrete-fracture model for multiphase flow in porous media," *AIChE J.*, vol. 46, no. 6, pp. 1120–1130, 2000.

[14] Luiz C. N. Amado and Oswaldo A. Pedrosa Jr., "A finite volume approach with triangular grids in reservoir simulation," *SPE Advanced Technology Series*, vol. 2, no. 1, pp. 179–185, 1994.

[15] Robert Eymard and Fernand Sonier, "Mathematical and numerical properties of control-volume, finite-element scheme for reservoir simulation," *SPE Reservoir Eng.*, pp. 283–289, 1994.

[16] Peter A. Forsyth, "A control volume finite element approach to NAPL groundwater contamination," *SIAM J. Sci. Stat. Comput.*, vol. 12, no. 5, pp. 1029–1057, 1991.

[17] Larry S. K. Fung, Lloyd Buchanan, and Ravi Sharma, "Hybrid-cvfe method for flexible-grid reservoir simulation," *SPE Reservoir Eng.*, pp. 188–194, 1994.

[18] Larry S. K. Fung, A. D. Hlebert, and Long X. Nghiem, "Reservoir simulation with a control-volume finite-element method," *SPE Reservoir Eng.*, pp. 349–357, 1992.

[19] C. Prakash, "Examination of the upwind (donor-cell) formulation in control volume finite-element methods for fluid flow and heat transfer," *Numer. Heat Transf.*, vol. 11, pp. 401–416, 1987.

[20] Rainer Helmig and Ralf Huber, "Node-centered finite volume discretizations for the numerical simulation of multiphase flow in heterogeneous porous media," *Computat. Geosci.*, vol. 4, pp. 141–164, 2000.

[21] Randolph E. Bank and Donald J. Rose, "Some error estimates for the box method," *SIAM J. Numer. Anal.*, vol. 24, pp. 777–787, 1987.

[22] I. Aavatsmark, T. Barkve, Ø. Bøe, and T. Mannseth, "Discretization on unstructured grids for inhomogeneous, anisotropic media. Part II: Discussion and numerical results," *SIAM J. Sci. Comput.*, vol. 19, no. 5, pp. 1717–1736, 1998.

[23] Yi-Kun Yang, *Finite-Element Multiphase Flow Simulation*, Ph.D. thesis, University of Utah, Salt Lake City, UT, 2003.

[24] P. A. Raviart and J. M. Thomas, "A mixed finite element method for 2nd order elliptic problems," in *Mathematical Aspects of the Finite Element Method, Lecture Notes in Mathematics*, vol. 606, pp. 292–315. Springer, Berlin, 1977.

[25] B. L. Darlow, R. E. Ewing, and M. F. Wheeler, "Mixed Finite Element Method for Miscible Displacement Problems in Porous Media," *Soc. Petrol. Eng. J.*, pp. 391–398, August 1984.

[26] G. Chavent, G. Cohen, J. Jaffre, M. Dupuy, and I Ribera, "Simulation of Two-Dimensional Waterflooding By Using Mixed Finite Elements," *Soc. Petrol. Eng. J.*, pp. 382–390, 1984.

[27] L. J. Durlofsky and M. C. H. Chien, "Development of a Mixed Finite-Element-Based Compositional Reservoir Simulator," in *Proceedings of the 1993 SPE Symposium on Reservoir Simulation*, 1993.

[28] Todd Arbogast, Mary F. Wheeler, and Ivan Yotov, "Mixed Finite Elements for Elliptic Problems with Tensor Coefficients as Cell-Centered Finite Differences," *SIAM J. Sci. Comput.*, vol. 34, no. 2, pp. 828–852, April 1997.

[29] Theodor D. van Golf-Racht, *Fundamentals of Fractured Reservoir Engineering*, Elsevier Scientific Publishing Company, New York, NY, 1982.

[30] Roberto Aguilera, *Naturally Fractured Reservoirs*, PennWell Publishing Company, Tulsa, OK, 2nd edition, 1995.

[31] Norman J. Hyne, *Nontechnical Guide to Petroleum Geology, Exploration, Drilling, and Production*, PennWell Corporation, Tulsa, Oklahoma, 2001.

[32] W.F. Ramirez, *Application of Optimal Control Theory to Enhanced Oil Recovery*, Elsevier Science, Amsterdam, The Netherlands, 1987.

[33] L.S. Pontryagin, R.V. Boltyanskii, R.V. Gamkrelidze, and E.F. Mischenko, *The Mathematical Theory of Optimal Processes*, Wiley, New York, 1962.

[34] W.F. Ramirez, Z. Fathi, and J.L. Cagnol, "Optimal Injection Policies for Enhanced Oil Recovery: Part 1 - Theory and Computational Strategies," *SPE Journal*, pp. 328–332, June 1984.

[35] Z. Fathi and W.F. Ramirez, "Optimal Injection Policies for Enhanced Oil Recovery: Part 2 - Surfactant Flooding," *SPE Journal*, pp. 333–341, June 1984.

[36] W. Liu, W.F. Ramirez, and Y.F. Qi, "Optimal Control of Steamflooding," *SPE Advanced Technology Series*, vol. 1(2), no. 2, pp. 73–82, 1993.

[37] G. Mehos and W.F. Ramirez, "Use of Optimal Control Theory to Optimize Carbon Dioxide Miscible-Flooding Enhanced Oil Recovery," *Journal of Petroleum Science and Engineering*, vol. 2, pp. 247–260, 1989.

[38] B. Sudaryanto and Y.C. Yorstos, "Optimization of Fluid Front Dynamics in Porous Media Using Rate Control. I. Equal Mobility Fluids," *Physics of Fluids*, vol. 12(7), no. 7, pp. 1656–1670, 2000.

[39] D.R. Brouwer and J.D. Jansen, "Dynamic Optimization of Water Flooding with Smart Wells using Optimal Control Theory," in *Proceedings of the SPE European Petroleum Conference*, Aberdeen, UK, October 2002.

[40] R.E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[41] R.E. Bellman and S.E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, NJ, 1962.

[42] D. Murray, *Differential Dynamic Programming for the efficient solution of optimal control problems*, Ph.D. thesis, University of Arizona, 1978.

[43] D.M. Murray and S.J. Yakowitz, "Constrained Differential Dynamic Programming and its Application to Multireservoir Control," *Water Resour. Res.*, vol. 15(5), no. 5, pp. 1017–1027, 1979.

[44] D.M. Murray and S.J. Yakowitz, "Differential Dynamic Programming and Newton's Method for Discrete Optimal Control," *J. Optim. Theory Appl.*, vol. 43(3), no. 3, pp. 395–414, 1984.

[45] J.F.A. De O. Pantoja, "Differential Dynamic Programming and Newton's Method," *Int. J. Control*, vol. 47(5), no. 5, pp. 1539–1553, 1988.

[46] D.H. Jacobson and D.Q. Mayne, *Differential Dynamic Programming*, American Elsevier, New York, NY, 1970.

[47] S. Yakowitz and B. Rutherford, "Computational Aspects of Discrete-Time Optimal Control," *Appl. Math. Comp.*, vol. 15, pp. 29–45, 1984.

[48] L.-C. Chang, C.A. Shoemaker, and P.L.-F. Liu, "Optimal Time-Varying Pumping Rates for Groundwater Remediation: Application of a Constrained Optimal Control Algorithm," *Water Resour. Res.*, vol. 28(12), no. 12, pp. 3157–3173, 1992.

[49] L-C. Chang, *Application of Constrained Optimal Control Algorithms to Groundwater Remediation*, Ph.D. thesis, Cornell University, 1990.

[50] T.F. Edgar, D.M. Himmelblau, and L.S. Lasdon, *Optimization of Chemical Processes*, McGraw-Hill, New York, NY, 2001.

[51] M.T. Heath, *Scientific Computing: An Introductory Survey*, McGraw-Hill, New York, NY, 1997.

[52] P.E. Gill, W. Murray, and M.H. Wright, *Practical Optimization*, Academic Press, New York, NY, 1981.

[53] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer-Verlag, New York, NY, 1999.

[54] B. Yeten, L.J. Durlofsky, and K. Aziz, "Optimization of Smart Well Control," in *Proceedings of the SPE International Thermal Operations and Heavy Oil Symposium and International Horizontal Well Technology Conference*, Calgary, Canada, November 2002.

[55] J.E. Dennis and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, PA, 1983.

[56] N. Nishikiori, R.A. Redner, D.R. Doty, and Z. Schmidt, "An Improved Method for Gas Lift Allocation Optimization," in *Proceedings of the SPE Annual Technical Conference and Exhibition*, San Antonio, Texas, USA, October 1989.

[57] D.G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, 1984.

[58] K. Dutta-Roy and J. Kattapuram, "A New Approach to Gas Lift Allocation Optimization," in *Proceedings of SPE Western Regional Meeting*, Long Beach, California, USA, June 1997.

[59] Y-K. Yang, *Finite-Element Multiphase Flow Simulation*, Ph.D. thesis, University of Utah, 2003.

[60] J. Nocedal, "Updating Quasi-Newton Matrices with Limited Storage," *Mathematics of Computation*, vol. 35(151), no. 151, pp. 773–782, 1980.

[61] S.Benson and J.J. Moré, "A Limited Memory Variable Metric Method in Subspaces and Bound Constrained Optimization Problems," Technical Report ANL/MCS-P909-0901, Argonne National Laboratory, 2001.

[62] S.Benson, L.C. McInnes, and J.J. Moré, "TAO Users Manual," MCS Division Technical Report ANL/MCS-TM-242 - Revision 1.3, Argonne National Laboratory, March 2002.

[63] L.A. Saputelli, S. Mochizuki, L. Hutchins, R. Cramer, M.B. Anderson, J.B. Mueller, A. Escorcia, A.L. Harms, C.D. Sisk, S. Pennebaker, J.T. Han, A. Brown, C.S. Kabir, R.D. Reese, G.J. Nú nez, K.M. Landgren, C.J. Mckie, and C. Airlie, "Promoting Real-Time Optimization of Hydrocarbon Producing Systems," in *Proceedings of the SPE Offshore Europe Conference*, Aberdeen, UK, September 2003.

[64] C.K. Strayhorn, "Manual for Discounting Oil and Gas Income," http://www.window.state.tx.us/taxinfo/proptax/ogman/index.html.

[65] C.D.F. Ridders, "Accurate Computation of F'(x) and f'(x)f"(x)," *Adv. Eng. Software*, vol. 4(2), no. 2, pp. 75–76, 1982.

[66] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, New York, NY, 1992.