Mining University of Leoben, Austria
Department of Mineral Resources and Petroleum Engineering
Institute of Reservoir Engineering
Laboratory of Applied Computational Thermofluiddynamics

DEVELOPMENT AND APPLICATION OF A RESERVOIR SIMULATOR BASED ON THE OPEN SOURCE SOFTWARE OPENFOAM

A

Thesis

Submitted to the

Department of Mineral Resources and Petroleum Engineering,
Institute of Reservoir Engineering at Mining University of Leoben, Austria

in partial fulfillment of the requirements
for the degree of

Master of Science (M.Sc.)

By:

Emmanuel TCHATCHOUA
Bachelor's Degree in Physics
B.Sc. in Petroleum Engineering

Under the supervision of:

Prof. Dr.Dipl.-Ing. Wilhelm Brandstätter

Leoben, December 2007

I declare herewith that this thesis is entirely my own work and that I have only consulted the references quoted herein.

E. TCHATCHOUA

Leoben, December 2007/Austria.

To my splendid wife

Carole Aimee TCHATCHOUA

and my loved sons

Alfred Miguel Nana TCHATCHOUA

and

Dominik Gael Kamgaing TCHATCHOUA.

Acknowledgement

Special thanks to my supervisor Professor Wilhelm Brandstätter for his valuable supervision, his constant advices and guidance throughout this work.

I would like to thank particularly DDr Hofmann Herbert and the Afro-Asiatic Institute in Graz for their financial supports during all my studies in Austria.

I would like to thank Dipl.Ing. Michael Klug for his technical support with OpenFOAM software.

Special thank to Adel Ragab PhD student in the same laboratory for his assistance. It was nice to work in the same office with you .Thank you Adel.

I would also like to thank all members of the Mineral Resources and Petroleum Engineering Department, all members of the Institute of Reservoir Engineering and finally of the Laboratory of Applied Computational ThermofluidDynamics at the Mining University of Leoben.

I would also like to thank both companies Heinemann Oil GmbH and ICE (Innovative Computational Engineering) Strömungsforschung GmbH for provides me data for this work and for their technical support according to the development of this simulator.

Special thanks to my parents Mr Tchatchoua Jeremie (not more in live) and Mrs.Tchatchoua Pauline for given me the life, for their encouragements, financial supports during all my life.

Special thanks to my brother Mr Nana Raymond and his wife Mrs. Nana Lysette for their financial supports and affections. Without you, I would not have known who I am.

Special thanks to my wife Carole Aimee Tchatchoua and my sons Alfred Miguel Nana Tchatchoua and Gael Dominik Kamgaing Tchatchoua for all. Without you I couldn’t reach this goal. I will just say you are angels.

Acknowledgement

Special thank to my brother Elvis Fleury Doutio Nana who studies Mechanical Engineering in the same university with me for his councils. Elvis you were every week in my office to know what is going on. You knew how to recomfort me during difficult moments. Thanks Elvis.

Special thanks to my family, my brothers Roger, Christian, my sisters Angele Claire and Adele Sidonie. My Aunt Mrs Nguepdjop Odette and all members of my family.

Finally I would like to thank all my friends, particularly, Bismarck Owusu Fosu from Ghana, Serge Tako form Ivory Coast, Rubis Bazira from Burundi, Jean Luc Bala, Sonja Berghöfer, Martina Ehgartner, Bettina Schmitt from Austria and all students, colleges,and co-inhabitants in my student's dormitory. Thanks!

Contents

1.0 Introduction.

1.1 Background of the study
1.2 Chapters preview

2.0 Explanation of important concepts and general formulation of reservoir simulation.

2.1 Explanation of important concepts
2.2 General formulation of reservoir simulation
2.2.1 General formulation
2.2.2 Models and components of reservoir simulator
2.2.2.1 Models to simulate a reservoir
2.2.2.2 Components of a reservoir simulator

3.0 OpenFOAM software.

3.1 OpenFOAM overview
3.2 Software characteristics and description
3.3 Applications and libraries
3.4 The programming language of OpenFOAM
3.4.1 Object-orientation and C++
3.4.2 Equation representation
3.4.3 Solver codes
3.4.4 Compiling applications and libraries
3.5 OpenFOAM cases
3.5.1 File structure of OpenFOAM cases
3.6 Post-processing
3.6.1 Overview of paraFoam

4.0 Development of the simulator for a single phase flow.

4.1 Assumptions
4.2 Classification of 2D and 3D problems
4.3 Model formulation and fluid flow equations
4.3.1 Model formulation
4.3.2 Fluid flow equations of single phase: Mathematical model
4.3.2.1 Darcy's law
4.3.2.2 Law of mass conservation
4.3.2.3 General flow equation
4.3.2.3.1 Elastic porous medium with constant permeability
4.3.2.3.2 Equation for low compressible oil and elastic porous medium

4.4 Discretization of the non partial differential equation: Numerical model
4.4.1 Introduction
4.4.2 Spatial discretization: Finite volume method (FVM)
4.4.3 Temporal discretization
4.4.4 Equation discretization
4.5 Linearization of the non-linear algebraic equations system and solution-Method
4.6 Algorithm of the numerical solution based on OpenFOAM: computer model
4.6.1 The *case* directory
4.6.1.1 The subdirectory *(0)*
4.6.1.2 The subdirectory *(constant)*
4.6.1.3 The subdirectory *(system)*
4.6.2 The *solver* directory
4.6.2.1 The main code *laplacianFoamSimulator.C*

5.0 Description and application of the 2D model.

5.1 Description of the model called *Reservoir2D*
5.1.1 Geometry of the model
5.2 Simulation of different scenarios, results and analysis
5.2.1 1st Scenario: Case *Reservoir2D1* for no flow boundaries
5.2.1.1 Results from OpenFOAM version 1.4 simulator
5.2.1.2 Results from Eclipse simulator
5.2.1.3 Comparison of results and analysis
5.2.2 2nd Scenario: Case *Reservoir2D2* for one flow boundary with constant pressure
5.2.2.1 Results from OpenFOAM version 1.4 simulator and Eclipse
5.2.2.2 Comparison of results and analysis
5.2.2.3 Field pressure comparison between *Reservoir2D1* and *Reservoir2D2*
5.2.3 3rd Scenario: Case *Reservoir2D2Incompressible* for incompressible flow and one flow boundary with constant pressure
5.2.3.1 Results from OpenFOAM version 1.4 simulator
5.2.3.2 Field pressure comparison between *Reservoir2D2* and *Reservoir2D2Incompressible*

6.0 Description and application of the 3D model.

6.1 Description of the model called *Reservoir3D*
6.1.1 Geometry of the model
6.2. Scenario: Case *Reservoir3D* for low compressible flow with closed boundaries

6.2.1 Results from OpenFOAM version 1.4 simulator and analysis

7.0 Conclusion, recommendations and future research.

7.1 Conclusion
7.2 Recommendations and future research

References

Appendices

Figure 2.1 Workflow of a reservoir simulator
Figure 3.1 Overview of OpenFOAM structure
Figure 3.2 Case directory structure
Figure 3.3 The paraFoam main window
Figure 4.1 Linear flow in a porous rock of length L
Figure 4.2 Flow though a volume element in a cartesian coordinate system
Figure 4.3 Example of volume field defined on a mesh with 2 boundaries patches (in 2D)
Figure 4.4 Parameters in finite volume discretization

Case *Reservoir2D1*: No flow boundaries

Figure 5.1 2D-reservoir geometry

OpenFOAM solutions

Figure 5.2 Initial pressure distribution
Figure 5.3 Pressure distribution after 1 time step (0.02 day)
Figure 5.4 Pressure distributios after 10 time steps (0.2 day)
Figure 5.5 Pressure distribution after 20 time steps (0.4 day)
Figure 5.6 Pressure distribution after 50 time steps (1 day)
Figure 5.7 Pressure distribution after 75 time steps (1.5 days)
Figure 5.8 Pressure distribution after 100 time steps (2 days)
Figure 5.9 *Reservoir2D1* performance plot (red-average field pressure, green- bottom hole flowing pressure) after 2 days of production

Eclipse solutions

Figure 5.10 Initial pressure distribution
Figure 5.11 Pressure distribution after 100 time steps (2 days)
Figure 5.12 Field performance plot (red-average field pressure, green- bottom hole flowing pressure) after 2 days of production from Eclipse
Figure 5.13 *Reservoir2D1* performance plot (red-average field pressure, green- bottom hole flowing pressure) after 2 days of production from OpenFOAM
Figure 5.14 Final field pressure comparison between Eclipse and OpenFOAM for case 1

Final pressure distribution of case 1 OpenFOAM-Eclipse

Figure 5.15 Pressure distribution after 1 time step (0.02 day)
Figure 5.16 Pressure distribution after 10 time steps (0.2 day)
Figure 5.17 Pressure distribution after 20 time steps (0.4 day)
Figure 5.18 Pressure distribution after 50 time steps (1 day)
Figure 5.19 Pressure distribution after 75 time steps (1.5 day)
Figure 5.20 Pressure distribution after 100 time steps (2 days)

Case *Reservoir2D*: One flow boundary with a constant pressure OpenFOAM –Eclipse solutions

Figure 5.21 Initial pressure distribution
Figure 5.22 Pressure distribution after 1 time step (0.02 days)
Figure 5.23 Pressure distribution after 5 time steps
Figure 5.24 Pressure distribution after 10 time steps
Figure 5.25 Pressure distribution after 20 time steps
Figure 5.26 Pressure distribution after 30 time steps
Figure 5.27 Pressure distribution after 50 time steps
Figure 5.28 Pressure distribution after 75 time steps
Figure 5.29 Pressure distribution after 100 time steps (2 days)
Figure 5.30 *Reservoie2D2* performance plot (red-average field pressure, green- bottom hole flowing pressure) after 2 days of production from OpenFOAM
Figure 5.31 Performance plot showing differences between closed boundaries model and constant boundary model from Eclipse
Figure 5.32 Final field pressure comparison between Eclipse and OpenFOAM for case 2
Figure 5.33: Field pressure comparison between *Reservoir2D1* and *Reservoir2D2*

Case *Reservoir2D2Incompressible:* Incompressible flow and one flow boundary with a constant pressure

Figure 5.34 Initial pressure distribution
Figure 5.35 Pressure distribution after 1 time step
Figure 5.36 Pressure distribution after 10 time steps
Figure 5.37 Pressure distribution after 20 time steps
Figure 5.38 Pressure distribution after 50 time steps
Figure 5.39 Pressure distribution after 75 time steps
Figure 5.40 Pressure distribution after 100 time steps
Figure 5.41 *Reservoir2D2Icompressible* performance Plot (red-average field pressure, green- bottom hole flowing pressure) after 2 days of production
Figure 5.42 Field pressure comparison between *Reservoir2D2* and *Reservoir2D2Incompressible*

Case *Reservoir3D:* low compressible flow with closed boundaries

Figure 6.1 3D-reservoir geometry
Figure 6.2 Initial pressure distribution

Figure 6.3 Pressure distribution after 100 time steps (2 days) (plane z=0.5)
Figure 6.4 Pressure distribution after 100 time steps (2 days) (plane y-z)
Figure 6.5 Pressure distribution after 100 time steps (2 days) (plane x-z)
Figure 6.6 Pressure distribution after 400 time steps (8 days) (plane z=0.5)
Figure 6.7 Pressure distribution after 400 time steps (8 days) (plane y-z)
Figure 6.8 Pressure distribution after 400 time steps (8 days) (plane x-z)
Figure 6.9 Pressure distribution after 1000 time steps (20 days) (plane z=0.5)
Figure 6.10 Pressure distribution after 1000 time steps (20 days) (plane y-z)
Figure 6.11 Pressure distribution after 1000 time steps (20 days) (plane x-z)
Figure 6.12 Pressure distribution after 2000 time steps (40 days) (plane z=0.5)
Figure 6.13 Pressure distribution after 2000 time steps (40 days) (plane y-z)
Figure 6.14 Pressure distribution after 2000 time steps (40 days) (plane x-z)
Figure 6.15 Pressure distribution after 4000 time steps (80 days) (plane z=0.5)
Figure 6.16 Pressure distribution after 4000 time steps (80 days) (plane y-z)
Figure 6.17 Pressure distribution after 4000 time steps (80 days) (plane x-z)
Figure 6.18 *Reservoir3D* performance plot (red-Average field pressure, green-bottom hole flowing pressure) after 160 days of production

Table B-1 Formation volume factor and viscosity as a function of pressure

Table B-2 Post-processing data from Eclipse for 2D

Table C-1 WBHFP, FPR data of the case *Reservoir2D1*

Table C-2 WBHFP, FPR data of the case *Reservoir2D2*

Table C-3 WBHFP, FPR data of the case *Reservoir2D2Incompressible*

Table C-4 Pressure-data comparison between Eclipse and OpenFOAM for *Reservoir2D1*

Table C-5 Pressure-data comparison between Eclipse and OpenFOAM for *Reservoir2D2*

Table C-6 WBHFP, FPR data of the case Reservoir3D

Table D-1 S.I. base units of measurement

A	Cross sectional area of the block	[m^2]
B	Formation volume factor at pressure p	[-]
B^0	Formation volume factor at a reference pressure p^0	[-]
c_o	Oil compressibility	[1/Pa]
c_ϕ	Rock compressibility	[1/Pa]
Dp	Piezometric conductivity or hydraulic diffusivity	[m^2s^{-1}]
g	Gravitational acceleration	[ms^{-2}]
k	Permeability	[m^2]
OOIP	Original oil in place	[m^3]
p	Pressure	[Pa]
PV	Pore volume	[m^3]
q	Flow rate	[m^3s^{-1}]
t	Time	[s]
Δt	Time interval	[s]
T	Temperature	[K]
u	Superficial or Darcy velocity	[ms^{-1}]
V	Rock volume	[m^3]
ΔV	Control volume	[m^3]
x	Distance in x- direction	[m]
y	Distance in y- direction	[m]
z	Vertical depth in z-direction	[m]
μ	Dynamic viscosity	[$kgm^{-1}s^{-1}$]

ϕ	=	Porosity	[-]
ρ	=	Oil density	[kgm^{-3}]
ρ_{STC}	=	Oil density at standard conditions	[kgm^{-3}]

CFD	-	Computational Fluid Dynamics
CV	-	Control Volume
FDM	-	Finite Difference Method
FEM	-	Finite Element Method
FPR	-	Field Pressure
FVM	-	Finite Volume Method
ICCG	-	Incomplete Cholesky Conjugate Gradient
OOIP	-	Original Oil In Place
OpenCFD	-	Open Source Computational Flow Dynamics
OpenFOAM	-	Open Field Operation and Manipulation
PDE	-	Partial Differential Equation
PV	-	Pore Volume
Res2D1	-	*Reservoir2D1*
Res2D2	-	*Reservoir2D2*
Res2D2Inc	-	*Reservoir2D2Incompressible*
Res3D1	-	*Reservoir3D1*
Res3D2	-	*Reservoir3D2*
WBHP	-	Well Bottom Hole Pressure
WBHFP	-	Well Bottom Hole Flowing Pressure
WOPR	-	Well Oil Production Rate
1D	-	One dimensional
2D	-	Two dimensional
3D	-	Three dimensional

Abstract

A reservoir simulator is a blend of engineering, physics, chemistry, mathematics, numerical analysis and computer programming and finally experiences and practices. It is an important tool for oil companies to forecast reservoir performance, to improve reservoir description (History Match), for the development of simple models, correlations, and number of commercial software packages such as Eclipse and Sure are available to perform this task.

Software licence costs are high and modifications to the software are limited due to the inaccessibility of the source codes. In contrast to this,OpenFOAM is freely available, open source and licensed under GNU General Public License.

However, the OpenFOAM Software CFD toolbox can simulate anything from complex fluid flows involving chemical reactions, turbulence and heat transfer, to solid dynamics and electromagnetics.

Hence the aim of this work was to investigate the potential of the open source software OpenFOAM which is a free software, to be able to work as a hydrocarbon reservoir simulator .The obtained results are compared with the results of Eclipse simulator software[4] which is the most widely used tool in the area of hydrocarbon reservoir simulation.

It is demonstrated in this thesis that OpenFOAM has the potential to substitute commercial software.

Zusammenfassung

Ein Kohlenwasserstoff-Lagerstätten-Simulator ist ein wichtiges Werkzeug für die Ölindustrie, um das Verhalten von Erdöl-Erdgas-Lagerstätten zu prognostizieren, um Lagerstättensbeschreibungen zu verbessern, und verschiedene kommerzielle Software Pakette wie Eclipse and Sure sind verfügbar um diese Aufgaben durchzuführen.

Software Lizenzkosten sind hoch und Modifizierungen der Software sind wegen der Unzugänglichkeit zum Quellcode beschränkt.
Im Gegensatz dazu ist OpenFOAM eine frei verfügbare, Open Source-Software, die unter der GNU „General Public License“ lizenziert ist.

Der OpenFOAM Software CFD Toolbox kann alle Arten von Strömungen simulieren, die chemische Reaktionen, Turbulenz und Wärmeübertragung, auf feste Dynamik und Elektromagnetismus.

Ziel dieser Diplomarbeit ist es OpenFOAM als Lagerstätten-Simulator zu verwenden. Dazu nötige Erweiterungen des Quellcodes werden implementiert.
Die Ergebnisse werden mit jenen von Eclipse vergleichen um die Richtigkeit der Simulation zu beweisen. Aufgrund der Übereinstimmung der Ergebnisse kann demonstriert werden das OpenFOAM das potenzial hat um kommerzielle Software zu ersetzen.

1. Introduction

1.1 Background of the study

The primary objective of petroleum reservoir study is to predict future performance of a reservoir and find ways and means of increasing ultimate recovery.

We distinguish in general two types of reservoir model which are the geological model (describes the static model of the reservoir) and the simulation model which describe the dynamic behavior of the reservoir (fluid flow through the reservoir).[1]

Our topic will focus on the second model which is a blend of engineering, physics, chemistry, mathematics, numerical analysis and computer programming and finally experiences and practices. It is an important tool for reservoir characterization and management and it allows us to generate different production prognoses with different depletion strategies during the life of a field.

Classical reservoir engineering deals with the reservoir on a gross average basis (tank model) and cannot account adequately for the variations in reservoir (pressure, saturation) and fluid parameters (density, viscosity, formation volume factor,....) in space and time.

Reservoir simulation by computers allows a more detailed study of the reservoir by dividing the reservoir into a number of blocks and applying fundamental equations for flow in porous media to each block.

Typical uses of reservoir simulation are:[4]

*Asset valuation

Accurate determination of recoverable reserves

*Asset managament:

Determine the most economical perforation method, well pattern, number of wells to drill, injection rates

Determine appropriate facilities

*Uncertainty management:

Estimate financial risk of exploration prospects and early life-cycle fields

Assess the effects of early gas or water breakthrough or coning

Estimate means of meeting gas deliverability contracts

A reservoir simulator is a computer program that solves equations of heat and mass flow in porous media, subject to appropriate initial and boundary conditions.[7]

Hence the goal of this topic will be to develop a reservoir simulator based on an open source software used for computational flow dynamics process, called OpenFOAM .It is first and foremost a C++ library, used primarily to create executables known as applications.[10]

The applications fall into two categories:

Solvers, that are each designed to solve a specific problem in continuum mechanics and *utilities* that are designed to perform tasks that involve data manipulation.
OpenFOAM contains numerous solvers and utilities covering a wide range of problems.

One of the strengths of OpenFOAM is that new solvers and utilities can be created by its users with some prerequisite knowledge in physics and programming techniques. In this thesis, we will deal with the most basic of all reservoir models which is the Black-Oil model or beta-model and we will limit the work to a single phase flow.

We will not cover the 1D model in this work because it can seldom be used for field wide reservoir studies because it cannot model areal and vertical sweep.

1.2 Chapters preview

The second chapter will deal with the explanation of important concepts related to reservoir simulators and the general workflow of a simulator.

Chapter three explain pre-processing, processing and post-processing of the OpenFOAM software.

In the fourth chapter, we will deal with the development of a simulator according to a single phase flow (oil) and the description of the development of the simulator itself, from the physical formulation to the computer model.

In chapter five and six the application of the developed simulator, post-processing and analysis of the results are discussed

The task of chapter seven consists of conclusions, recommendations and some words about future research that has to be done in order to apply OpenFOAM as software for reservoir simulations.

2. Explanation of important concepts and general formulation of reservoir simulation

2.1 Explanation of important concepts

A hydrocarbon reservoir is a three dimensional, heterogeneous, anisotropic formation saturated with fluids (oil, gas, and water) of different compositions.

A reservoir simulator is a computer program that solves equations for heat and mass flow in porous media, subject to appropriate initial and boundary conditions. Its main tasks are to forecast reservoir performance (full field studies for depletion planning and field development, assessment of uncertainty in forecasting reservoir performance, reservoir management), to improve reservoir description (History matching, identification of fluid units, barriers and aquifer influx, near well properties, Dual porosity behavior, unstable displacements, heterogeneities) and to develop simple models and correlations (Coning studies,..).

2.2 General formulation of reservoir simulation

2.2.1 General formulation

Numerical reservoir simulators are used widely for some reasons. Primarily because they solve problems that cannot be solved in any other way. Simulation is the only way to describe quantitatively the flow of multiple phases in a heterogeneous reservoir having a production schedule determined not only by the properties of the reservoir, but also by market demand, investment strategies and government regulations.

Secondly, because it asseses economic and technical risks; optimize well locations, type and spacing; it is also cheaper or more reliable than other methods and

it increases profitability by improved reservoir management.

Finally, simulators respond to safety, environmental and regulatory concerns.

The first reservoir simulators has been developed between 1960 and 1970 but with poor reliability and confidence in technology due to first generation of digital computers limited by speed and storage.

Modern reservoir simulators appeared between 1970 and 1985 with increasing confidence in technology, decreasing hardware costs, availability of supercomputers and could describe multi-components fluid.

Today, there are various types of real modern reservoir simulators with high confidence in technology and which are the main important tools for reservoir engineers in operating companies. The most confidence one are commercial and someone can be found free.

2.2.2 Models and components of reservoir simulator

2.2.2.1 Models to simulate a reservoir

In reality there are three kinds of models involved in developing a program to simulate a reservoir.

Mathematical model

The physical system to be modeled must be expressed in term of appropriate mathematical equations. This process almost involves assumptions. The assumptions are necessary from a practical standpoint in order to make the problem tractable.

Numerical model

The equation constituting a mathematical model of the reservoir is almost always too complex to be solved by analytical methods. Approximations must be made to put the equations in a form that is amenable to solutions by digital computers.

Computer model

A computer program or a set of program written to solve the equations of the numerical model constitutes a computer model of the simulator.

2.2.2.2 Components of reservoir simulator

A reservoir simulator is composed of three main components:

Pre-processing consisting of grid generation, local grid refinement, aquifer modeling, flexible well modeling, fault modeling, data loading and importing from various sources and formats, data preparation modules (rock, PVT and well data)

Processing consisting of set up of the fluid model (initial and boundary conditions, fluid properties…) and the numerical calculations

Post-processing consisting of the Visualization of the calculated results on grid, visualization for results versus time, animation of results and exporting maps to third party products.

The main task of the simulator is carried out during the processing step. Hence a typical workflow of a reservoir simulator is carried out on the sketch below:

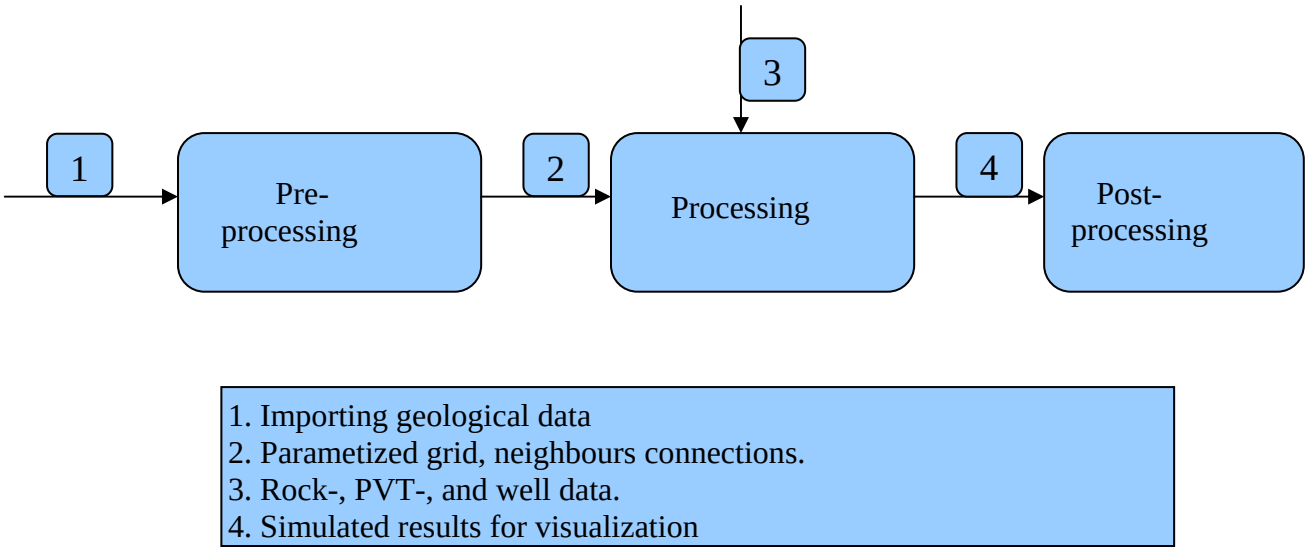


Figure 2.1 Workflow of a reservoir simulator[7]

3. Open Field Operation and Manipulation (OpenFOAM) software

3.1 OpenFOAM overview

OpenFOAM is an open source software. This means a software where the source code(the language programmers use to make computers do their jobs) is available to everyone. Anyone can see how the code works and can change it if they want to make it work differently.

The opposite of open source is closed source where the source is not available to everyone.

3.2 Software description and characteristics

OpenFOAM is first and foremost a C++ library, used primarily to create executables, known as applications. The applications fall into two categories: *solvers*, that are each designed to solve a specific problem in continuum mechanics; and *utilities*, that are designed to perform tasks that involve data manipulation. The OpenFOAM distribution contains numerous solvers and utilities covering a wide range of problems.[10]

One of the strengths of this software is that new solvers and utilities can be created by its users with some pre-requisite knowledge of physics and programming techniques involved.

OpenFOAM is supplied with pre- and post-processing environments. The interface to the pre- and post-processing are themselves OpenFOAM utilities, thereby ensuring consistent data handling across all environments.[10]

The overall structure of OpenFOAM is shown in figure 3.1 below:

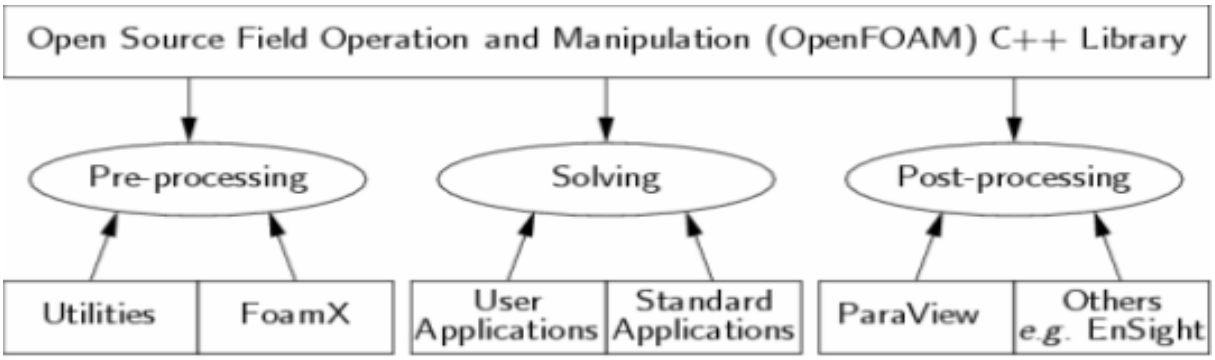


Figure 3.1 Overview of OpenFOAM structure.[10]

OpenFOAM is supplied with pre-processing tools like *FoamX* and *blockMesh,* processing solvers (see Appendix) and a post-processing tool (paraView).

The software always operates in a 3 dimensional cartesian coordinate system and all geometries are generated in 3 dimensions. OpenFOAM solves the case in 3 dimensions by default but can be instructed to solve in 2 dimensions by specify a "special" *empty* boundary condition on boundaries normal to the (3rd) dimension for which no solution is required.[10]

3.3 Applications and libraries

We should reiterate from the outset that OpenFOAM is a C++ library used primarily to create executables, known as *applications.[10]*

OpenFOAM is distributed with a large set of precompiled applications but users also have the freedom to create their own or modify existing ones. Applications are split into two main categories:

Solvers

Those are each designed to solve a specific problem in computational continuum mechanics;

Utilities

Those perform simple pre-and post-processing tasks, mainly involving data manipulation and algebraic calculations.

OpenFOAM is divided into a set of precompiled libraries that are dynamically linked during compilation of the solvers and utilities.
Libraries such as those for physical models are supplied as source code so that users may conveniently add their own models to the libraries.[10]

3.4 The programming language of OpenFOAM

In order to understand the way in which the OpenFOAM library works, some background knowledge of C++, the base language of OpenFOAM, is required; the necessary information will be presented in this chapter.

3.4.1 Object-orientation and C++

The clarity of having objects in programming that represents physical objects and abstract entities should not be underestimated.

The class structure concentrates code development to contained regions of the code, i.e. the classes themselves, thereby making the code easier to manage. New classes can be derived or inherit properties from other classes, e.g. the *vectorField* can be derived from a vector class and a *Field* class.

C++ provides the mechanism of template classes such that the template class *Field*<Type> can represent a field of any <Type>, e.g. scalar, vector, tensor. The general features of the template class are passed on to any class created from the template. Templating and inheritance reduce duplication of code and create class hierarchies that impose an overall structure on the code.

3.4.2 Equation representation

A central theme of the OpenFOAM design is that the solver applications, written using the OpenFOAM classes, have a syntax that closely resembles the partial differential equations being solved.

For example the equation

$$\frac{\partial \rho u}{\partial t} + \nabla . \phi u - \nabla . \mu \nabla u = -\nabla p \tag{1}$$

is represented by the code as

```
solve
(
    fvm::ddt (rho, u)
  + fvm::div(phi, u)
  - fvm::laplacian(mu, u)
    = =
  - fvc::grad(p)
);
```

This and other requirements demand that the principal programming language of OpenFOAM has object-oriented features such as inheritance, template classes, virtual functions and operator overloading.

These features are not available in many languages that purport to be object-orientated but actually have very limited object-orientated capability, such as FORTRAN-90.

C++, however, possesses all these features while having the additional advantage that it is widely used with a standard specification so that reliable compilers are available that produce efficient executables.
It is therefore the primary language of OpenFOAM.

3.4.3 Solver codes

Solver codes are largely procedural since they are a close representation of solution algorithms and equations, which are themselves procedural in nature.

3.4.4 Compiling applications and libraries

Compilation is an integral part of application development that requires careful management since every piece of code requires its own set instructions to access dependent components of the OpenFOAM library.

In Unix/Linux systems these instructions are often organized and delivered to the compiler using the standard UNIX make utility.

OpenFOAM, however, is supplied with the *wmake* compilation script that is based on *make* but is considerably more versatile and easier to use; *wmake* can, in fact, be used on any code, not simply the OpenFOAM library.

3.5 OpenFOAM cases

This chapter deals with the file structure and organization of OpenFOAM cases. Normally, we would assign a name to a case.

3.5.1 File structure of OpenFOAM cases

The basic directory structure for an OpenFOAM case, that contains the minimum set of files required to run an application, is shown in 3.2 and described as follows:

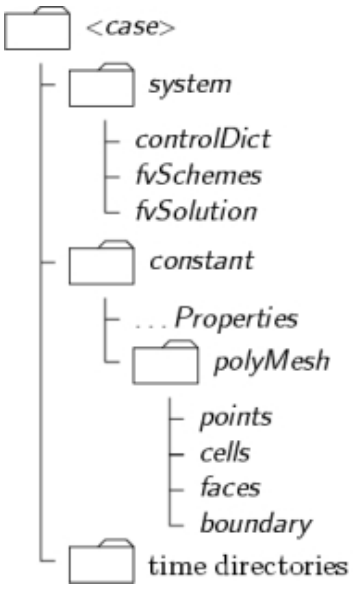


Figure 3.2 Case directory Structure[10]

A *constant* **directory**

That contains a full description of the case mesh in a subdirectory *polyMesh* and files specifying physical properties for the application concerned, *e.g. transportProperties*. *tuburlenceProperties, porousModel …*

A *system* **directory**

For setting parameters associated with the solution procedure itself. It contains at least the following 3 files:

controlDict where run control parameters are set including start/end time, time step and parameters for data output.

fvSchemes where discretization schemes used in the solution may be selected at run-time.

And,

fvSolution where the equation solvers, tolerances and other algorithm controls are set for the run.

The *time* directories

The *time* directories contain individual files of data for particular fields.
The data can be: either, initial values and boundary conditions that we must specify to define the problem; or results written to file by OpenFOAM.

Note that the OpenFOAM fields must always be initialized, even when the solution does not strictly require it, as in steady-state problems. The name of each time directory is based on the simulated time at which the data is written.

It is sufficient to say now that since we usually start our simulations at time $t=0$ the initial conditions are usually stored in a directory named *0* or *0.000000e+00*, depending on the name format specified.[10]

3. 6 Post-processing

OpenFOAM is supplied with post-processing utility *paraFoam* that uses *paraView*, an open source visualization application

3.6.1. Overview of paraFoam

paraFoam is strictly a script that launches *paraView* using the reader module supplied with OpenFOAM.

It is executed like any of the OpenFOAM utilities with the root directory path and the case directory name as arguments:

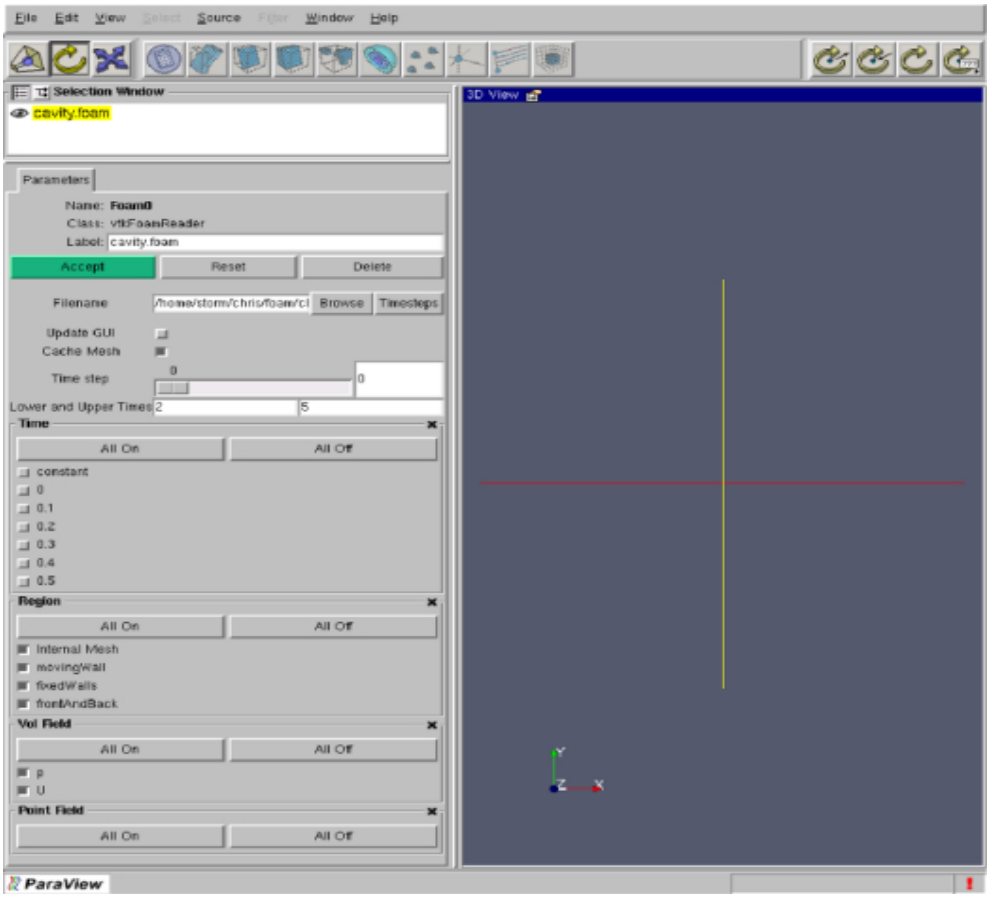


Figure 3.3 The paraFoam main window

Selection Window

The selection window lists the modules opened in *paraView*, where the selected modules are highlighted in yellow and the graphics for the given module can be enabled/disabled by clicking the eye button alongside.

Parameters panel

The parameters panel contains the input selections for the case, such as times, regions and fields.

Display panel

The display panel controls the visual representation of the selected module, e.g. colours.

Information panel

The Information panel gives case statistics such as mesh geometry and size . *paraView,* operates a tree-based structure in which data can be filtered from the top-level case module to create sets of sub-modules.

For example, a contour plot of pressure could be a sub-module of the case module which contains all the pressure data.

The strength of *paraView* is that the user can create a number of sub-modules and display whichever ones they feel to create the desired image or animation.

For example, they may add some solid geometry, mesh and velocity vectors, to a contour plot of pressure, switching any of the items on and off as necessary.

4. Development of the simulator for single phase flow

4.1 Assumptions

During this work, some assumptions have to be taken into consideration:
We will deal with an undersaturated (initial reservoir pressure is greater than the bubble point pressure of the reservoir fluid) single phase oil.

The flow will also be considered to be laminar (for the validity of Darcy's law), and the gravitational effect is negligible because we don't consider vertical flow across layers

4.2 Classification of 2D and 3D problems

All real reservoirs are, of course, three-dimensional; it is possible in many practical situations to assume that flow in one of the three coordinate directions is negligible compared to flow in other two directions. This is the concept of 2D problems.

Hence there are three classes of problems that can be handled in this manner.

***Areal problems (x-y)**

In thin reservoirs of large extent it is often possible to assume that the pressure gradients and hence flow in the z direction is negligible compared to flow in other two directions.[2]

***Cross-sectional problems(x-z)**

If instead of neglecting flow in the vertical direction, flow in one of the two horizontal directions is neglected, the resulting model is called cross-sectional.
This type of model can be used for cases where the flow is predominantly in the vertical direction and one of the two horizontal directions.[2]

***Single-well problems(r-z)**

One of the important applications of the fluid flow equations to petroleum engineering problems is in the field of well testing. Oil and gas wells are tested to determine their productivity.
Most well tests involve the production of a well at a sequence of constant flow rates and the observation of bottom hole pressure during the test period.

A comparison of test data with theory allows to predict the basic reservoir data and hence the productivity of the well.

3D problems are the most important one used for reservoir management.

4.3 Model formulation and fluid flow equations

4.3.1 Model formulation

Consider a finite system (reservoir) and the flow of a single fluid (single component or a homogeneous mixture through it) .The system exists in x, y, z space and time t.

Observations:

Anything that enters or leaves the system must cross the boundary (Boundary conditions).[7]

At some initial time, we can describe the system state (Initial conditions).[7]

The process occurring within the system obeys some physical laws.
The final observation will define the fluid flow equation through the system.

This flow is based on two majors laws which are:

Darcy's law for single phase flow.

Law of mass conservation

4.3.2 Fluid flow equations of single phase: Mathematical model

4.3.2.1 Darcy's law

In addition to the equation of mass conservation (equation of continuity), we have to derive a relationship between flow rate and pressure gradient of the phase.
Such a relationship was discovered by Darcy (1856) for single phase flow as illustrated in the figure 4.1.

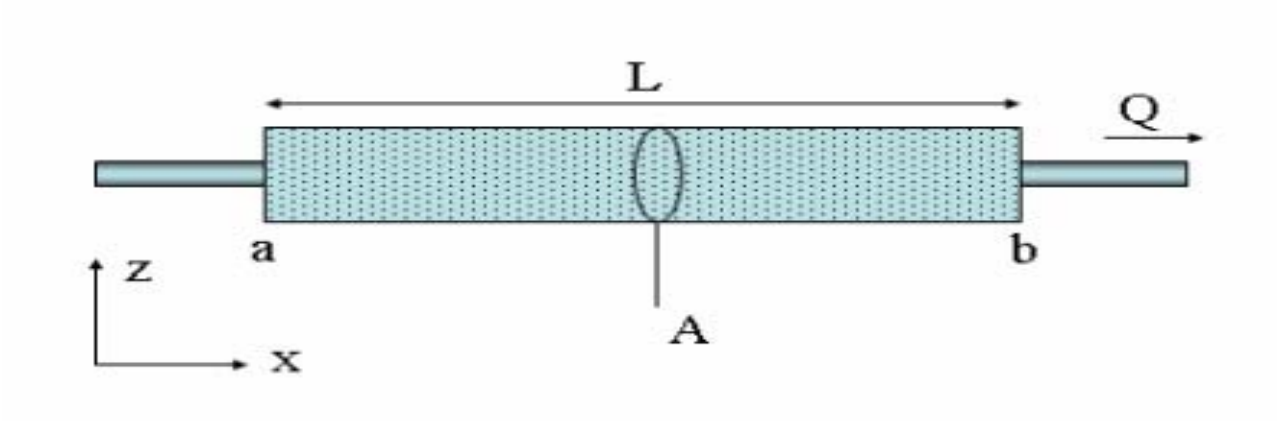


Figure 4.1: Linear flow in a porous rock of length L [16]

Darcy's law is a simple proportional relationship between the instantaneous discharge rate through a porous medium, the viscosity of the fluid and the pressure drop over a given distance.[16]

$$Q = \frac{-kA}{\mu}\frac{\Delta p}{L} \qquad (2)$$

The total discharge, Q (units of volume per time, e.g., $m^3.s^{-1}$) is equal to the product of the permeability k of the medium, the cross-sectional area to flow A, and the pressure drop ∇p, all divided by the viscosity μ and the length L the pressure drop is taking place over.

The negative sign is needed because a fluid flows from high pressure to low pressure. However, if the change in pressure is negative (in the x-direction) the flow will be positive (in the x-direction).

Dividing both sides of the equation by the area and using more general notation leads to

$$u = \frac{Q}{A} = -\frac{k}{\mu}\nabla p \tag{3}$$

where u is the flux (discharge per unit area, with units of length per time, m.s-1) and Δp is the pressure gradient vector.

This value of flux, often referred to as the Darcy flux, is not the real velocity with which the oil traveling through the pores.This expression for flux is only valid for horizontal flow or flow in the absence of gravity.

To express the full 3D general formulation of Darcy's law, one must subtract the hydrostatic fluid pressure, in essence stating that a hydrostatic pressure gradient does not lead to flow.

The differential forming 3D of this relation is

$$\vec{u} = -\frac{k}{\mu}\left(\nabla p + \rho\vec{g}\right) \tag{4}$$

where k is the absolute permeability tensor of the porous medium, μ is the fluid viscosity, $\vec{g}$ is the gravitational acceleration vector

The permeability tensor used in equation (4) is determined experimentally. In most practical problems it is possible (or necessary) to assume that k is a diagonal tensor given in 3D by

$$\begin{pmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{pmatrix} \tag{5}$$

In the following, We will introduce a so called Potential function instead of pressure

$$\psi = gz + \int_{p0}^{p} \frac{dp}{\rho} \tag{6}$$

with

$$\rho = \rho(p) \tag{7}$$

where

$p0$ is the pressure at the reference depth.

Then equation (4) becomes

$$\vec{u} = -\frac{k\rho}{\mu} \nabla \psi \tag{8}$$

This equation is a linear vector-vector equation. In an isotropic porous medium the permeability k is a scalar but in an anisotropic medium it is a tensor.
Thus

$$\overline{u} = -\frac{\rho}{\mu} \overline{k} \nabla \psi \tag{9}$$

Note that $\overline{u}$ is not the true velocity but the Darcy velocity or average velocity which is the fluid velocity at core scale and not at pore scale

4.3.2.2 Law of mass conservation

The equation of continuity describes the law of mass conservation.

Consider the flow trough a parallelepiped control volume core with the porosity ϕ as shown in Figure 4.3, the control volume must be representative of the porous medium. It should be large compared to the size of the pores but small compared to the size of the core.

Hence the physical property of the porous medium, like the porosity may be associated with the control volume.

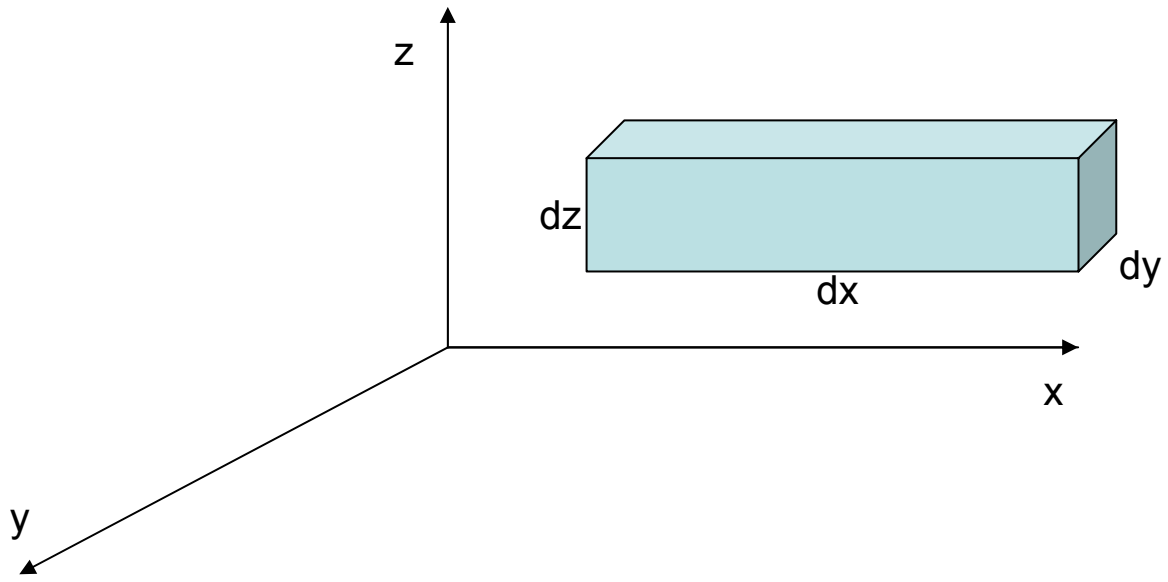


Figure 4.2: Flow though a volume element in a cartesian coordinate system

The pore volume of the volume element is then:

$$\phi dxdydz \tag{10}$$

and the fluid mass content in the control volume is:

$$\rho\phi dxdydz \tag{11}$$

The change of quantity during a time interval dt is:

$$\frac{\partial(\rho\phi)}{\partial t} dxdydzdt \tag{12}$$

On the other side, the quantity of fluid flowing trough the surface at x and x+dx is:

$$(\rho u_x)_x \, dydzdt \tag{13}$$

and

$$(\rho u_x)_{x+dx} \, dydzdt \tag{14}$$

The change of flowing quantity results in :

$$-\frac{\partial}{\partial x}(\rho u_x) dxdydzdt \tag{15}$$

Also considering the filtration in the direction of other coordinates, we have the following:

$$-\frac{\partial}{\partial y}(\rho u_y) dxdydzdt \tag{16}$$

$$-\frac{\partial}{\partial z}(\rho u_z) dxdydzdt \tag{17}$$

Combining equations (15) (16) and (17) leads to:

$$-\frac{\partial}{\partial t}(\phi\rho)=\sum_{i=1}^{3}\frac{\partial}{\partial x_i}(\rho u_i) \tag{18}$$

where, the term dxdydz was already cancelled, and with

$$\begin{aligned} x_2 &= y \\ x_3 &= z \end{aligned} \tag{19}$$

after writing equation (4) in a vector form, it becomes

$$-\frac{\partial}{\partial t}(\phi\rho)=\nabla(\rho\vec{u}) \tag{20}$$

Equation (20) is the equation of continuity describing mass flow through our reservoir

4.3.2.3 General flow equation

The substitution of equation (9) into equation (20) yields to:

$$\nabla(\frac{\bar{k}}{\mu}\rho^2\nabla\psi)=-\frac{\partial}{\partial t}(\phi\rho) \tag{21}$$

or in details form in 3D :

$$\frac{\partial}{\partial x}\left[\frac{k_x}{\mu}\rho^2\frac{\partial\psi}{\partial x}\right]+\frac{\partial}{\partial y}\left[\frac{k_y}{\mu}\rho^2\frac{\partial\psi}{\partial y}\right]+\frac{\partial}{\partial z}\left[\frac{k_z}{\mu}\rho^2\frac{\partial\psi}{\partial z}\right]=\frac{\partial}{\partial t}(\phi\rho) \tag{22}$$

From equation (6)

$$\psi = gz+\int_{p0}^{p}\frac{dp}{\rho} \quad \text{(equation (6))}$$

we get,

$$\nabla\psi = \frac{1}{\rho}\nabla p + g\nabla z \tag{23}$$

and

$$\rho^2\nabla\psi = \rho\nabla p + \rho_2 g\nabla z \tag{24}$$

Assuming the effect of gravity negligeable, the second term in equation (24) cancels And we have

$$\rho^2\nabla\psi = \rho\nabla p \tag{25}$$

Hence, equation (21) yields to

$$\nabla\left(\frac{\bar{k}}{\mu}\rho\nabla p\right) = -\frac{\partial}{\partial t}\left(\phi\rho\right) \tag{26}$$

Equation (26) ist one of the most important equations in this work which describe the flow through our reservoir with negligeable gravity effect.

In this work, we assume an isotropic medium hence k is a scalar

Moreover, oil viscosity is independent of pressure, hence equation (26) yields to:

$$\frac{\bar{k}}{\mu}\nabla(\rho\nabla p) = -\frac{\partial}{\partial t}\left(\phi\rho\right) \tag{27}$$

Equation (27) is the general flow equation for single phase oil ,low compressible with constant flow viscosity flowing through an isotropic elastic porous medium.[6]

4.3.2.3.1 Elastic porous medium with constant permeability

The variation of pore volume with pressure leads to the rock compressibility

$$c_\phi = -\frac{1}{V_\phi}\left[\frac{\partial V_\phi}{\partial p}\right]_T = -\frac{1}{\phi}\left[\frac{\partial \phi}{\partial p}\right]_T \tag{28}$$

Where ϕ is the porosity, V_ϕ is the pore volume and c_ϕ is the rock compressibility.

The integration of this equation yield to

$$\phi = \phi^0 \exp\left[c_\phi \left(p - p^0\right)\right] \tag{29}$$

And by applying Taylor's series, we obtain

$$\frac{\phi}{\phi^0} = 1 + c_R(p - p^0) + \theta(p - p^0) \tag{30}$$

Where ϕ^0 is the formation porosity at reference pressure p^0

By assuming that c_ϕ is small we can assume that the error term or truncation error $\theta(p - p^0) \to 0$ and hence from equation (30) we have

$$\phi = \phi^0\left[1 + c_R\left(p - p^0\right)\right] \tag{31}$$

4.3.2.3.2 Equation for low compressible oil and elastic porous medium

Compressible fluid means that the oil compressibility is different from zero

For low compressible oil one may assume that the fractional change volume of the fluis as pressure changes at constant temperature is constant.

This constant is called the coefficient of isothermal compressibility which is defined by:

$$c_o = -\frac{1}{V_0}\left[\frac{\partial V_0}{\partial p}\right]_T \tag{32}$$

By substituting :

$$V_0 = \frac{m}{\rho} \tag{33}$$

where, m is the mass of oil, a constant, into equation (32), we obtain

$$c_o = \frac{-1}{\frac{m}{\rho}} \frac{d\left(\frac{m}{\rho}\right)}{dp} = \frac{1}{\rho}\frac{d\rho}{dp} \tag{34}$$

After integration of equation (34) between a reference Pressure p^0 and a certain pressure p we obtain the following equation:

$$\rho = \rho^0 \exp\left[c_o\left(p - p^0\right)\right] \tag{35}$$

Where ρ^0 is the oil density at the reference pressure p^0

By applying Taylor's series, we obtain

$$\frac{\rho}{\rho^0} = 1 + c_o(p - p^0) + \theta(p - p^0) \tag{36}$$

By assuming that c_o is small we can assume that the error term or truncation error $\theta(p - p^0) \to 0$ and hence from equation (36) we have

$$\rho \approx \rho^0\left[1 + c_o\left(p - p^0\right)\right] \tag{37}$$

Moreover, from the definition of oil formation volume factor (B) it is obvious that

$$\frac{B^0}{B}=\frac{\rho^0}{\rho}\approx 1+c_o\left(p-p^0\right) \tag{38}$$

where B^0 is he oil formation volume factor at reference pressure p^0

According to Equation (35),

$$\nabla\rho=\nabla\left[\rho^0 \exp c_o\left(p-p^0\right)\right]=\rho^0 c_o \exp\left(p-p^0\right)\nabla p \tag{39}$$

Hence

$$\rho\nabla p=\frac{1}{c_o}\nabla\rho \tag{40}$$

Equation(27) yields to:

$$\frac{\bar{k}}{\mu c_o}\nabla(\rho)=-\frac{\partial}{\partial t}(\phi\rho) \tag{41}$$

Differentiation of equation(40) leads to:

$$\nabla\rho=\rho\, c_o\nabla p \tag{42}$$

And substituting equation (42) to equation (41) we obtain:

$$\frac{\bar{k}}{\mu}\nabla\left(\rho\nabla p\right)=-\frac{\partial}{\partial t}(\phi\rho) \tag{43}$$

Substituting equation (29) and equation (35) into equation (43) yields to:

$$\frac{\bar{k}}{\mu}\nabla\left(\rho\nabla p\right)=\frac{\partial}{\partial t}\left[\phi^0\rho^0\exp\left[\left(c_o+c_\phi\right)\left(p-p^0\right)\right]\right] \tag{44}$$

From equation (44) and after differentiation and simplification we obtain:

$$\nabla^2\left[\exp(c_o(p-p^0)\right]=\frac{\phi^0\mu(c_o+c_\phi)}{k}\exp(c_\phi(p-p^0)\frac{\partial}{\partial t}\left[\exp(c_o(p-p^0)\right] \quad (45)$$

Some approximations have to be made.

Since c_ϕ is mall, it can be assumed:

$$\exp(c_\phi(p-p^0))=1 \quad (46)$$

and also by already assuming(above) that

$$\exp(c_0(p-p^0))=1+c_o(p-p^0) \quad (47)$$

Equation(45) yields to:

$$\nabla^2 p=\frac{\phi^0\mu(c_o+c_\phi)}{k}\frac{\partial p}{\partial t} \quad (48)$$

Equation (48) is the equation of single phase(oil) filtration of a low compressible fluid with contant dynamic viscosity and permeability through an elastic porous medium.

To be in the language of OpenFOAM, this equation can also be writte in the following form:

$$\frac{\partial p}{\partial t}=\nabla^2(Dp,p) \quad (49)$$

with

$$Dp=\frac{k}{\phi^0\mu(c_o+c_\phi)} \quad (50)$$

Dp is the so called piezometric conductivity or hydraulic conductivity in porous media. It defines the transport properties of the reservoir.

4.4 Discretization of the non partial differential equation: Numerical model

4.4.1 Introduction

Equation (48) describes the fluid flow through our reservoir for low compressible flow without gravitational effect.

This equation is a non-linear partial differential equation which is difficult to solve analytically. It may sometimes be solved using Bäcklund Transformation characteristics, Green's function, integral transform, Lax pair, separation of variables, or-when all else fails (which it frequently does)-numerical methods such as finite volume , element or differences.[16]

In this work we will solve this equation numerically. This system of equations must be solving for every point of the reservoir as a function of time for pressure determination. We have no interest on oil saturation because in this work we suppose a single oil phase.

The discretization consists of the replacement of non linear PDEs with boundary conditions to non-linear algebraic equations that can be linearized. This means instead of searching a continuous solution, we will look for approximated values of the solutions on a finite set of grid points at discrete time levels (discrete solutions at discrete time).

We distinguish between *spatial discretization and temporal discretization.*

Spatial discretization: Defining the solution domain by a set of points that fill and bound a region of space when connected. Some methods used in this purpose are

Finite VolumeMethod (FVM)
Finite Element Method (FEM)
Finite Difference Method (FDM)
Method of Weighted Residuals (Explain in details by Finlayson)

OpenFOAM utilizes the control volume finite method as the discretisation method.

Temporal discretization *(For transient problems):* which is the division of the time domain into a finite number of time intervals or steps.

Both leads to

Equation discretization which generates a system of algebraic equations in terms of discrete quantities defined at specific locations in the domain, from the PDEs that characterize the problem.

4.4.2 Spatial discretization: Control Volume Finite Method (FVD)

For the finite volume the solution domain is subdivided into a finite number of contiguous control volumes (CVs) and the equations of fluid flow are applied to each control volumes. At the centroid of each CV lies a computational node (grid point) at which the pressure values are to be calculated. Interpolation is used to express variable values at the CV surface in terms of the nodal (CV-center) values. (see Figure 4.4)

Surface and volume integrals are approximated using suitable quadrature formulae.
As a result, one obtains an algebraic equation for each CV,in which a number of neighboor nodal values appear.

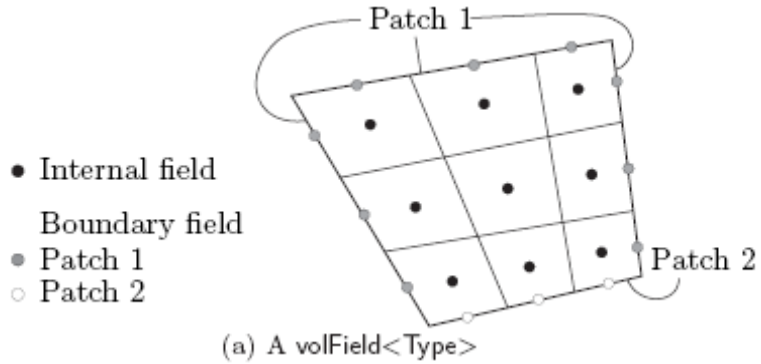


Figure 4.3: Example of volume field defined on a mesh with 2 boundaries patches (in 2D)[11]

The equation (49) as seen below

$$\frac{\partial p}{\partial t} = \nabla(Dp\nabla p)$$

is the equation to be spatially discretized. Dp is a scalar

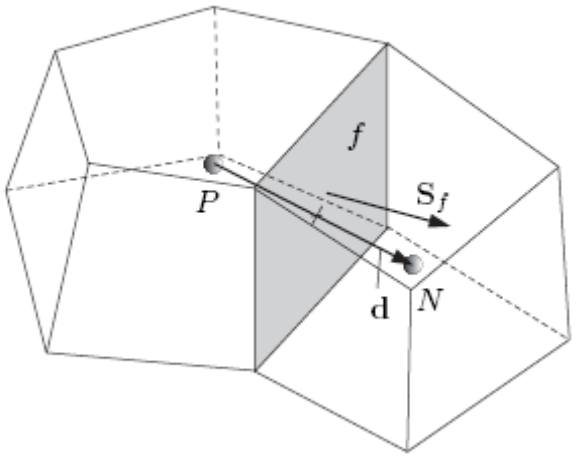


Figure 4.4: Parameters in finite volume discretization[11]

The cell is bounded by a set of flat faces, given the generic label f

The laplacian term is integrated over a control volume and linearised as follows:

$$\iiint_V \nabla(Dp\nabla p)dV = \iint_s dS.(Dp\nabla p) = \sum_f Dp_f S_f.(\nabla p)_f \tag{51}$$

The face gradient discretization is implicit when the length vector d between the centre of the cell P and the centre of the neighbouring cell N is orthogonal to the face plane. We deal with this case. Otherwise in case of non orthogonal meshes, an additional explicit term is introduced, which is evaluated by interpolating cell centres gradients, themselves calculated by central differencing cell centre values.

In OpenFOAM ,

$\nabla.(Dp\nabla p)$ is witten as $laplacian(Dp, p)$

4.4.3 Temporal discretization

Temporal discretization is the division of the time domain into a finite number of time intervals, or steps. The time is broken into a set of time step Δt that may change during numerical simulation, perhaps depending on some condition calculated during the simulation.[11]

The first time derivative $\partial/\partial t$ is integrated over a control volume as follow:

$$\frac{\partial}{\partial t}\iiint_V p dV \tag{52}$$

The term is discretized by simple differencing in time using the following:

new values

$$p^n \equiv p(t+\Delta t) \tag{53}$$

at the time step we are solving for

In this work we use the Euler implicit scheme that is the first order accurate in time defines as the following:

$$\frac{\partial}{\partial t}\iiint_V p dV = \frac{(p_p V)^n - (p_p V)^0}{\Delta t} \tag{54}$$

In OpenFOAM the first time discretization

$$\frac{\partial p}{\partial t} \quad \text{is written as} \quad ddt(p)$$

4.4.4 Equation discretization

After spatial and temporal discretization, we obtain from equation(51) and equation (54) the following discretized equation:

$$\frac{(p_p V)^n - (p_p V)^0}{\Delta t} = \sum_f Dp_f S_f . (\nabla p)_f \tag{55}$$

Equation (55) is the equation which will be solved by the solver *laplacianFoamSimulator,* to provide discrete pressure at every time step in each grid point

This equation (55) provides at every time step a system of algebric non linear equations that are commonly expressed in matrix form.

Equation (55) is written in OpenFOAM as:

$$ddt(p) = laplacian(Dp, p) \tag{56}$$

4.5 Linearization of the non-linear algebraic equations system and solution-method

For each time interval Δt equation (54) is a system of n non linear equations if n is the number of grid blocks of our spatial domain. The linearization of the equations is controlled by the *fvSchemes* subdirectory *of the System* directory defined in chapter 3. Because of severe restriction in time step size which can greatly cause instability with *Explicit-Method*, the solution-method used for pressure computation shall be the *Implicit* one.[11]

4.6 Algorithm of the numerical solution based on OpenFOAM: Computer model

A computer program or a set of program written to solve the equations of the numerical model constitute the computer model of the simulator.

The computer model used to solve our numerical model consists of two main directories:

A case directory which consists of several utilities and specifies our hydrocarbon reservoir.

A solver directory named laplacianFoamSimulator which contents all the programs necessary to solve our partial differential equation.

4.6.1 The case directory *(Reservoir)*

The structure of a case in OpenFOAM has already been defined in chapter 3.5.1.
It consists of subdirectories and files.

The main subdirectories of one case before running any simulation are:

- The subdirectory *(0)*
- The subdirectory *(constant)*
- The subdirectory *(system)*

4.6.1.1. The subdirectory *(0):*

The subdirectory *(0)* defines the initial condition at *startTime* and contents the file which describes the boundary conditions of the pressure field. (See the file on the appendix).

4.6.1.2 The subdirectory *(constant):*

The subdirectory *constant* consists of all data which don't change during the simulation. It may mainly consist of a subdirectory *polyMesh* and different files describing the transport properties of our system like the fluid viscosity, fluid density, rock porosity, rock permeability and rock compressibility.
The files which describe all these properties are in the Appendix.

The *polyMesh* directory is responsible for meshing our geometry.

The utility responsible for mesh generation in our geometry is the *blockMesh* utility.
The blockMesh utility creates parametric meshes with grading and curves edges. The mesh itself is generated from a dictionary file named *blockMeshDict* located in *constant/polyMesh* directory of the considered case (see the *blockMeshDict* file in the Appendix*).*

Within a *terminal window, blockMesh* can be run (creating the mesh of the geometry) with the command

blockMesh <path> caseName

or

blockMesh . caseName

ps: The *blockMeshDict* file must exist in the subdirectory *constant/polyMesh*

And,

The visualization of the meshing geometry in *paraView* is obtained with the following command on a terminal window

paraFoam <path> caseName* or *paraFoam . caseName

In our case the only directory used in the *constant* –subdirectory is:

.transportProperties

***transportProperties:**

It defines the piezometric or hydraulic conductivity which will be read by the program The entire *transportProperties* dictionary is available in the appendix.

4.6.1.3 The subdirectory *(system):*

The subdirectory *system* consists of data which change during simulation and are usually reread to each time step. It consists on three dictionaries which are:

fvSchemes
fvSolution
controlDict

***fvSchemes:**

This dictionary sets the numerical schemes for terms that appear in applications being run.(see 4.4.4 Laplacian schemes in OpenFOAM-Userguide).

fvSolution:

fvSolution controls the linear equation solvers and algorithm used in the solution during resolution of the linear equations system. To investigate pressure change in our reservoir during production, we will use linear solvers and the numerical process for simulation is based on the ICCG or AMG algorithm.

controlDict:

This dictionary contains information relating to the control of the solution procedure. (start time of the simulation, end time, writte solutions interval, solution format etc..)

4.6.2 The Solver directory *(laplacianFoamSimulator)*

This directory consists of a subdirectory named *Make,* the source codes *createFields.H* and *write.H* and *the* main code *laplacianFoamSimulator.C.*

4.6.2.3 The main code *laplacianFoamSimulator.C*

The main code *laplacianFoamSimulator.C* performs the following tasks:

increment the time step by *runTime++;*
generates the analytical solution for pressure field using the tensor arithmetic;
writes the solution to file by *runTime.writeObjects().*

The main code *laplacianFoamSimulator.C* is available on the Appendix and the core of the code which solves our equation is given on the next page:

laplacianFoamSimulator.C

```
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
Info << “\ nCalculating pressure distribution\n” << endl;
for  (runTime++ ; !runTime.end() ; runTime++)
{
        Info<< “Time = ”  <<runTime.timeName() << nl << endl;
        include “readSIMPLEControls.H”
        for (int nonOrth=0; nonOrth<=nNonOrthCorr; nonOrth++)
        {
            solve
            (
                fvm: :ddt(p) – fvm: :laplacian(Dp, p)
            )
        }
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  //
```

5. Description and application of a 2D model

5.1 Description of the model called *Reservoir2D*

The model is a one layer reservoir that has a simple 2D geometry with no dipping or faults.

The grid model used is a two dimensional one. The size of the model is 6 times 6 time 0.5 meters. A cartesian grid with a resolution of 0.2 meters is used to discretize the model.This results in 30 blocks in x and 30 blocks in y-direction leading to a total model size of 900 blocks.

To keep issues simple the model is set up with a homogeneous permeability and porosity distribution.

The density of oil is 800[kgm^{-3}] and its viscosity is 1.14 [cp].

The compressibility of the oil is modelled by the oil formation volume factor. Its numerical values are found on the appendix B.

The whole model is initialized with a pressure of 150E+05[Pa] valid at a reference depth of 1000.25[m], which corresponds the depth of the grid blocks of the 2D model.

Since only one phase (oil) is present the oil saturation equals unity for all cells.

Production takes place in one corner of the block model. The well bore diameter measures 0.1[m]. The well is operated in a way that it is bottom hole flowing pressure controlled.

The bottom hole pressure target is 50E+05[Pa].

The properties of the fine grid model are as follows:

Porosity $\phi = 0.3$; (constant throughout the run)

Viscosity (dynamic) μ = 1.14cp = 1.14E-03 [$kgs^{-1}m^{-1}$] (constant throughout the run)

Oil density ρ = 800 [kgm^{-3}]

Permeability k = 1[mD] = E-15[m^2] (constant throughout the run)

Rock compressibility c_ϕ = E-09[1/Pa]

Oil compressiblity:Because our oil is low compressible the compressibility is constant during pressure change. Due to the fact that the oil compressibility is modelled by the oil formation volume factor, according to equation (38) and using Table B-1 on the appendix B, we have the following:

$$c_o = \frac{1}{p - p^0}\left[\frac{B^0}{B} - 1\right] \tag{57}$$

With

p^0 = 150E+05[Pa] and p = 50E+05[Pa]

we obtain:

c_o = 4.21E-09[1/Pa]

According to the model described above and from the formula

$$^\circ API = \frac{141.5}{\rho} - 131.5 \tag{58}$$

The gravity of the oil is around 45.37° API which is a *light crude Oil.*

5.1.1 Geometry of the model

At the starting point, the fine scale grid is 30x30 cartesian grids with uniform size for each of the grid blocks in each block. The first investigation of this case considers to produce through a ¼ well located a the left corner of the geometry as seen in figure 5.1 below:

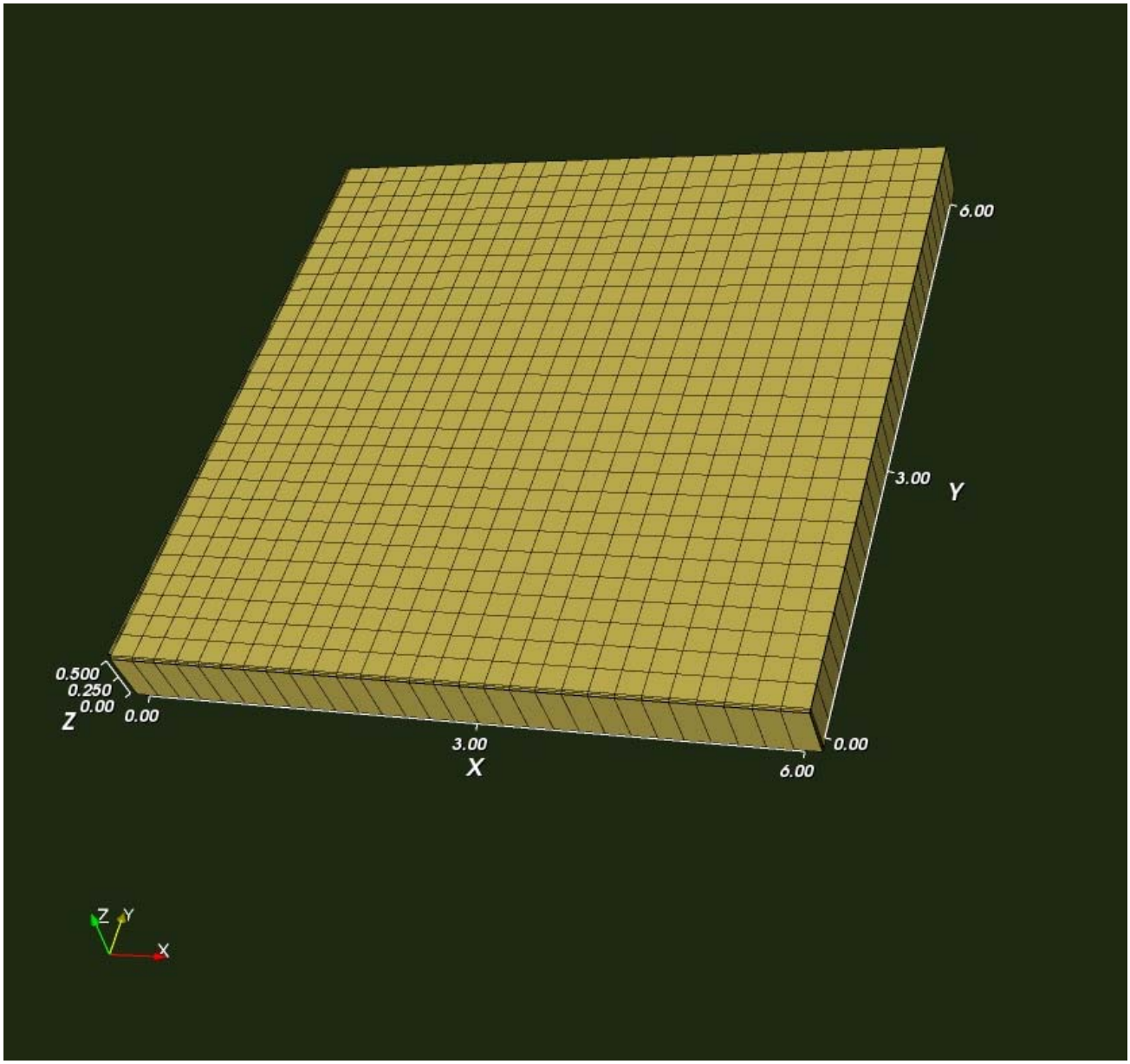


Figure 5.1 2D-Reservoir geometry

5.2 Simulation of different scenarios, results and analysis

The main task of the simulation was to investigate reservoir performance over time. In this case we will investigate the performance of the reservoir by generating some plots like:

Average pressure of the field
Bottom hole flowing pressure

to be able to compare the obtained results with those obtained from Eclipse.

To ensure a stable run small time steps are chosen. The time step lentgh is 0.02 days which correspond to 1728[s]. 100 time steps were calculated.
During this time no numerical problems occurred.

5.2.1 1st Scenario: Case *Reservoir2D1 for* no flow boundaries

According to the defined parameters and to equation (50), we have

$$Dp = 5.61\text{E-}04[m^2s^{-1}]$$

Dp is constant during simulation because of light compressibility property of the oil.

5.2.1.1 Results from OpenFOAM version 1.4 simulator

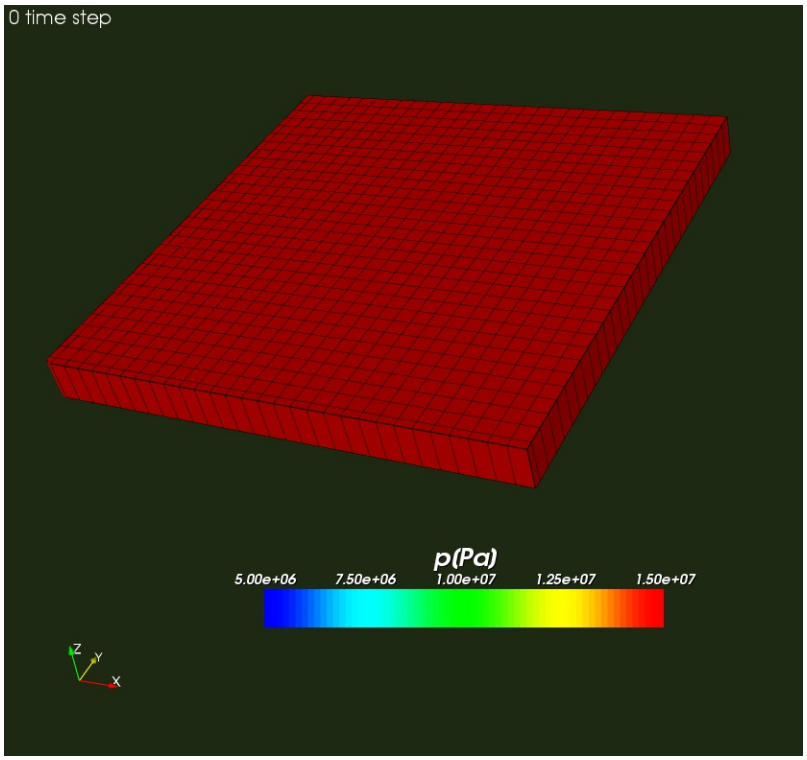


Figure 5.2: Initial pressure distribution

Figure 5.2 shows the initial pressure distribution for the two dimensional grid model. All cells have the same initial pressure of 150E+05[Pa]. (Assuming reference depth equal to top depth). The production well is located at the left corner.

For the next time steps this well was operated with a constant bottom hole flowing pressure of 50E+05[Pa].

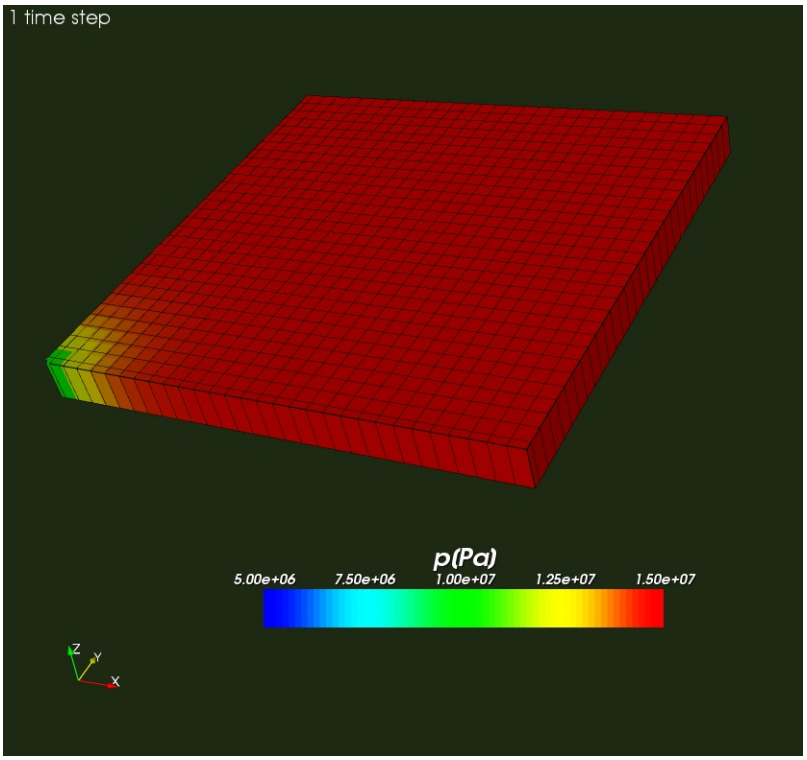


Figure 5.3: Pressure distribution after 1 time step (0.02 day)

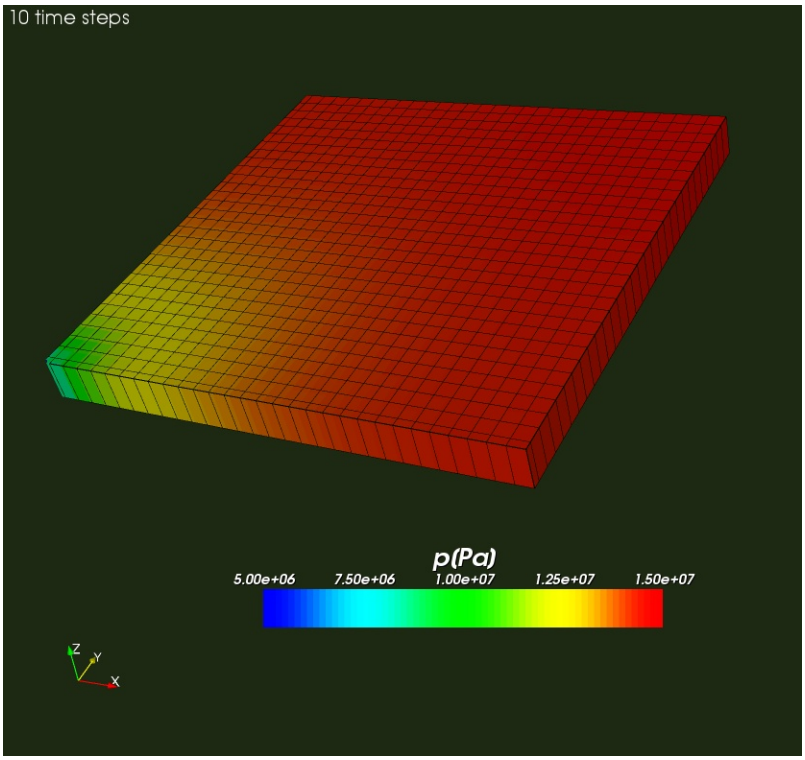


Figure 5.4: Pressure distribution after 10 time steps (0.2 day)

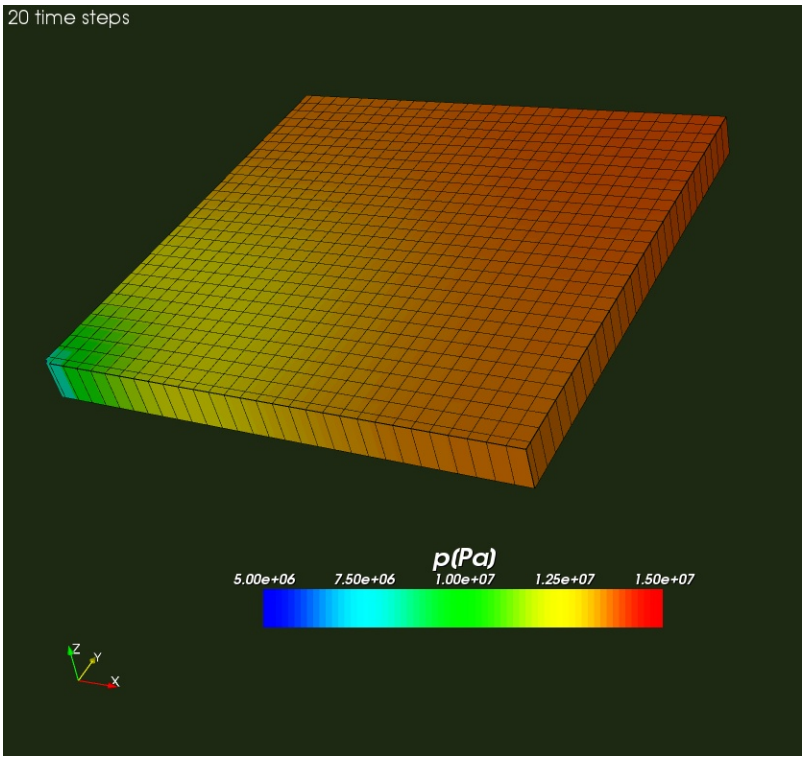


Figure 5.5: Pressure distribution after 20 time steps (0.04 day)

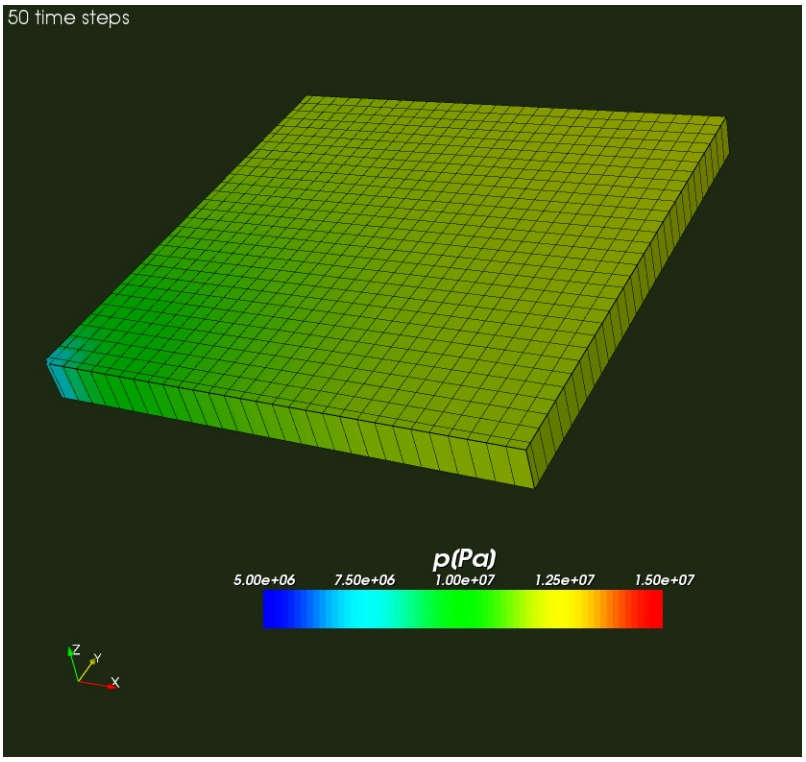


Figure 5.6: Pressure distribution after 50 time steps (1 day)

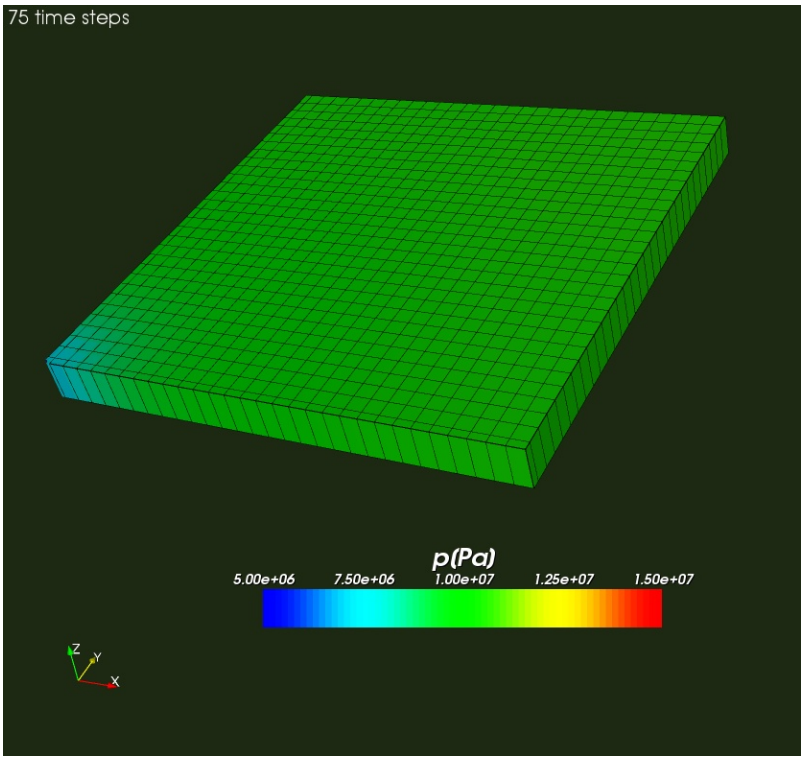


Figure 5.7: Pressure distribution after 75 time steps (1.5 days)

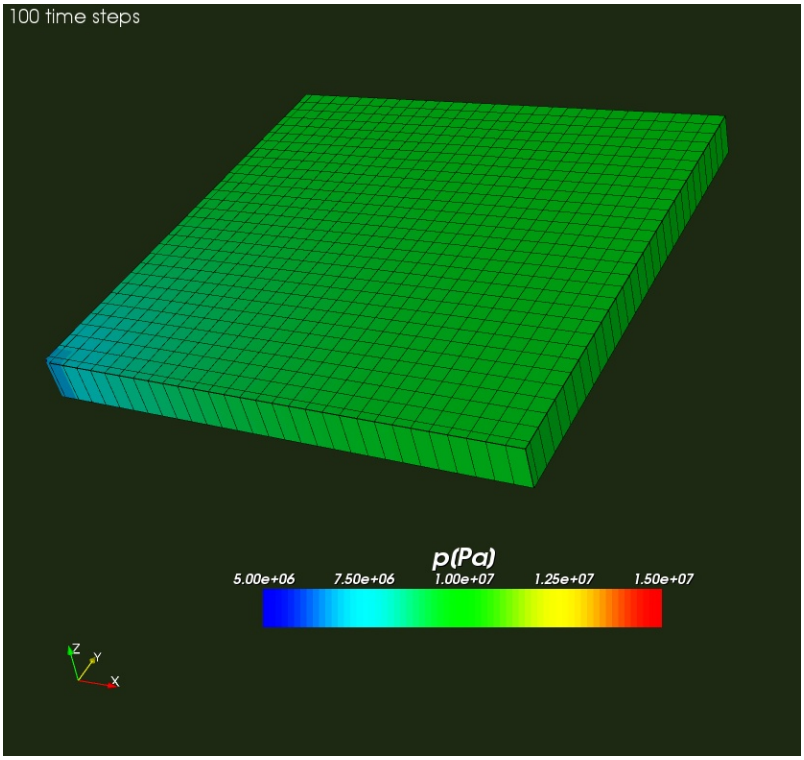


Figure 5.8: Pressure distribution after 100 time steps (2 days)

Performance plot.

The performance plot shows after 2 days of production, how the average field pressure decrease (red line) and how we control the well bottom hole flowing pressure (green line).

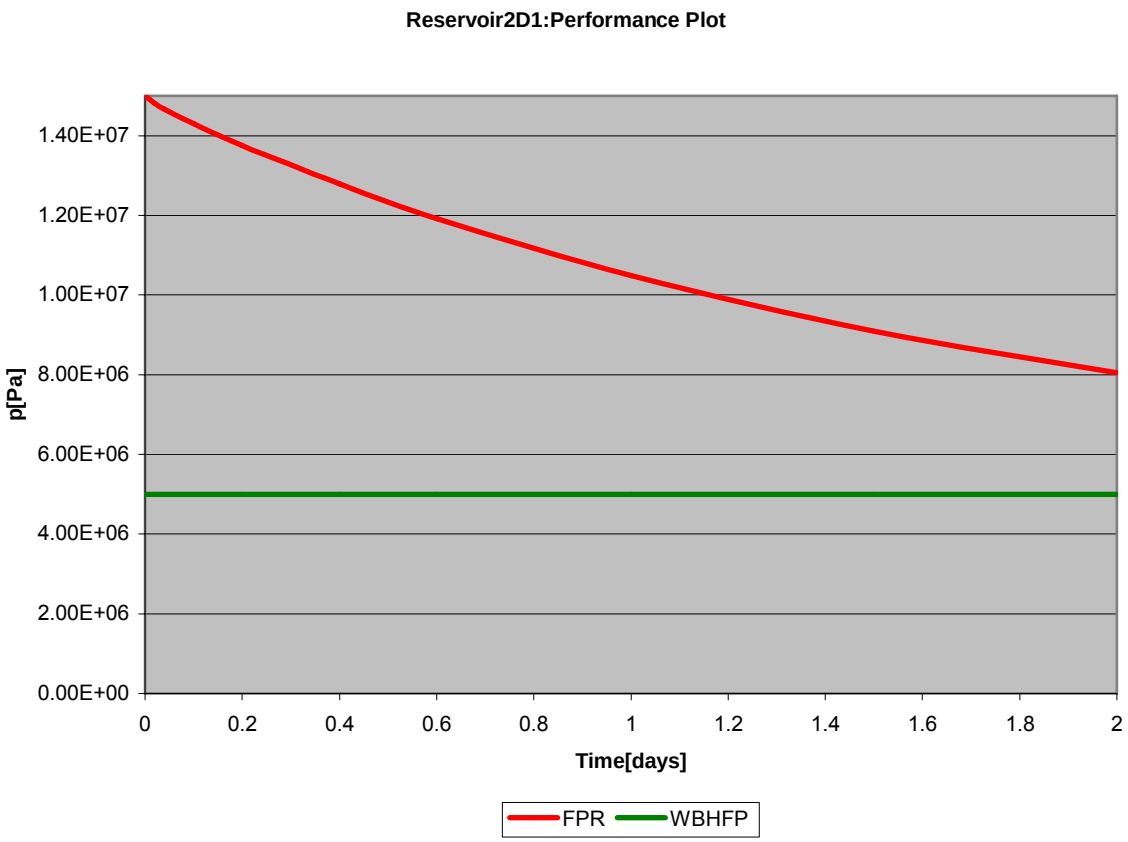


Figure 5.9: *Reservoir2D1* performance plot (red-average field pressure, green- bottom hole flowing pressure) after 2 days of production

5.2.1.2 Results from Eclipse simulator

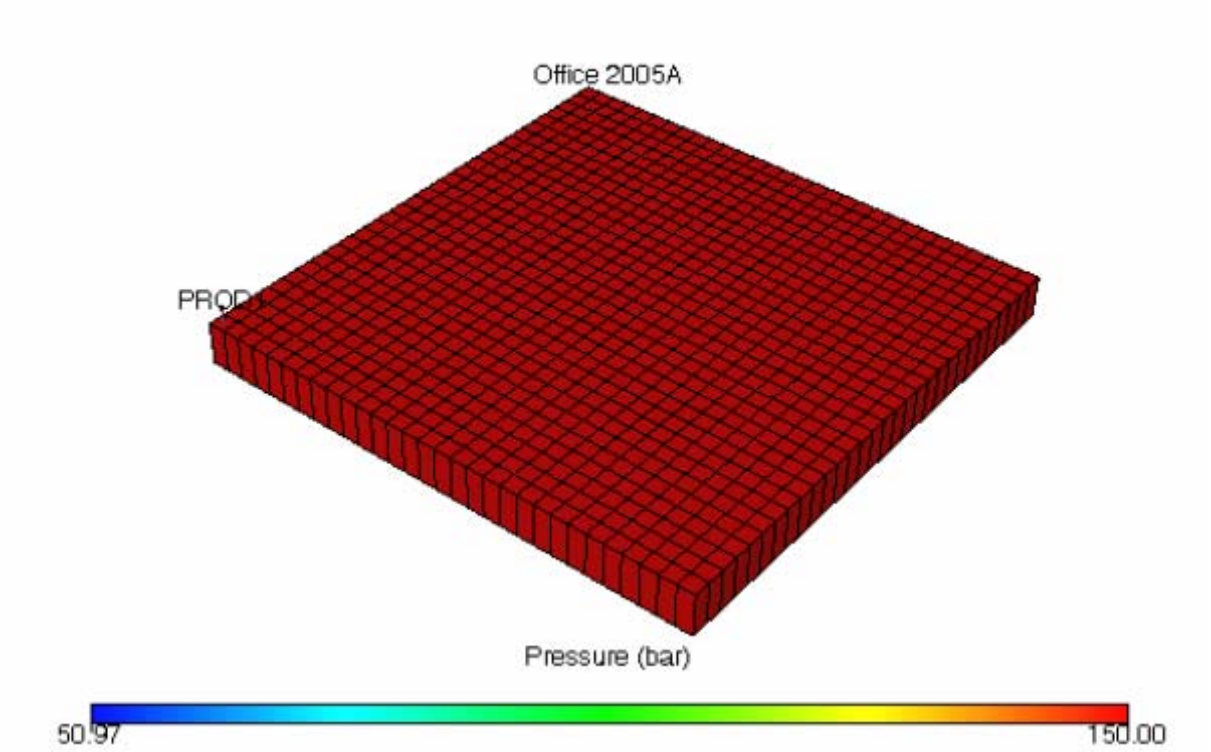


Figure 5.10: Initial pressure distribution

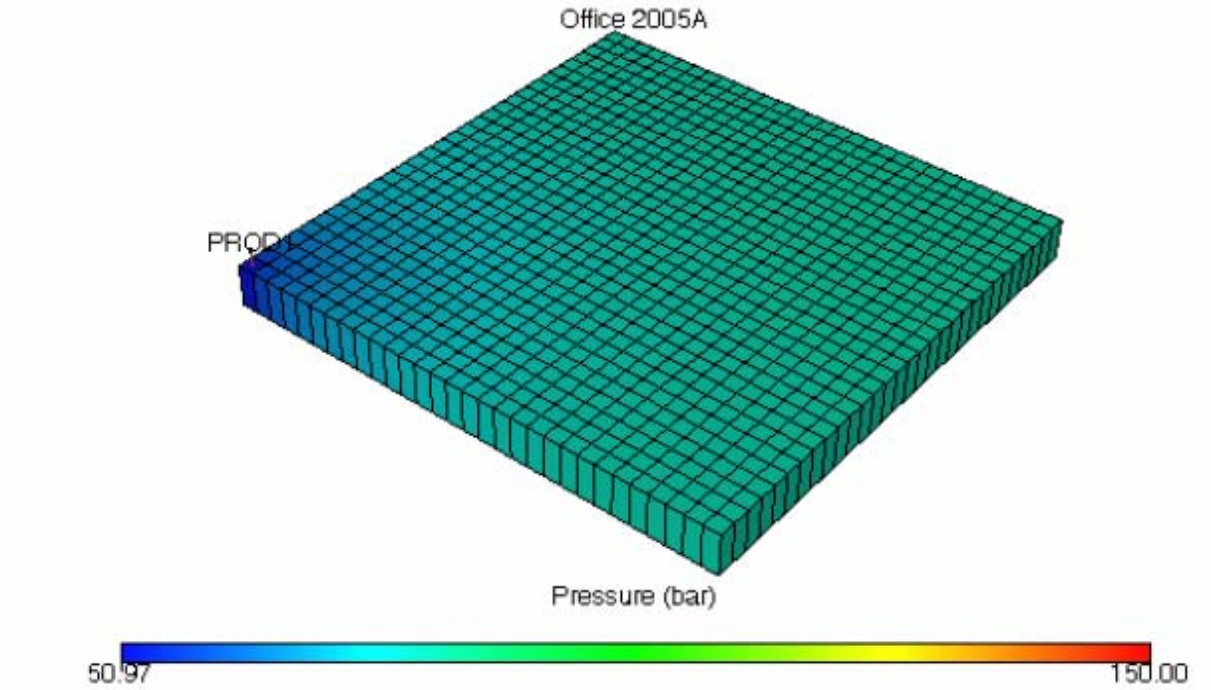


Figure 5.11: Pressure distribution after 100 time steps (2 days)

Performance plot.

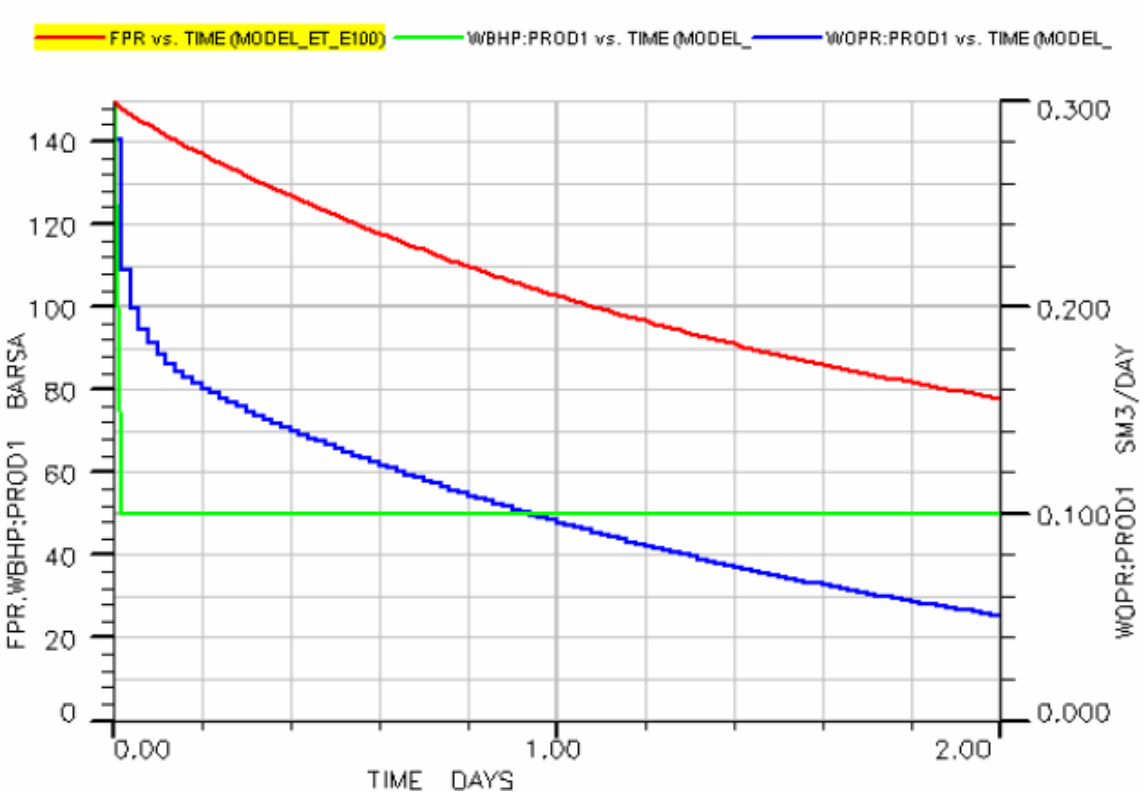


Figure 5.12: Field performance plot (red-average field pressure, green- bottom hole flowing pressure) after 2 days of production

5.2.1.3 Comparison of results and analysis

The first point comparing the results is to look out the pore volume and the original oil in place at initial pressure of both results.

Pore volume [m^3]

From Eclipse the initial pore volume is automatically generated as PV = 5.48106[m^3]

In OpenFOAM, we have the following:

$$PV = V.\phi = V\phi^0\left[1 + c_\phi \Delta p\right] = V\phi^0 \tag{59}$$

Because the the reference and the top depth are equals.
which leads to

PV = 5.4[m^3]

This corresponds to 98.53% of Eclipse results. This corresponds on a difference of around 1.47%.

Original oil in place (OOIP) [sm^3]

From Eclipse the original oil in place is automatically generated as
OOIP = 5.769536[Sm^3].

In OpenFOAM, we have the following:

$$OOIP = \frac{PV}{B^0} \tag{60}$$

Which leads to OOIP = 5.68 [Sm^3].
This difference is realistics because it is due to the 1.47% difference occuring in pores Volume.

Comparison of field pressure [Pa]

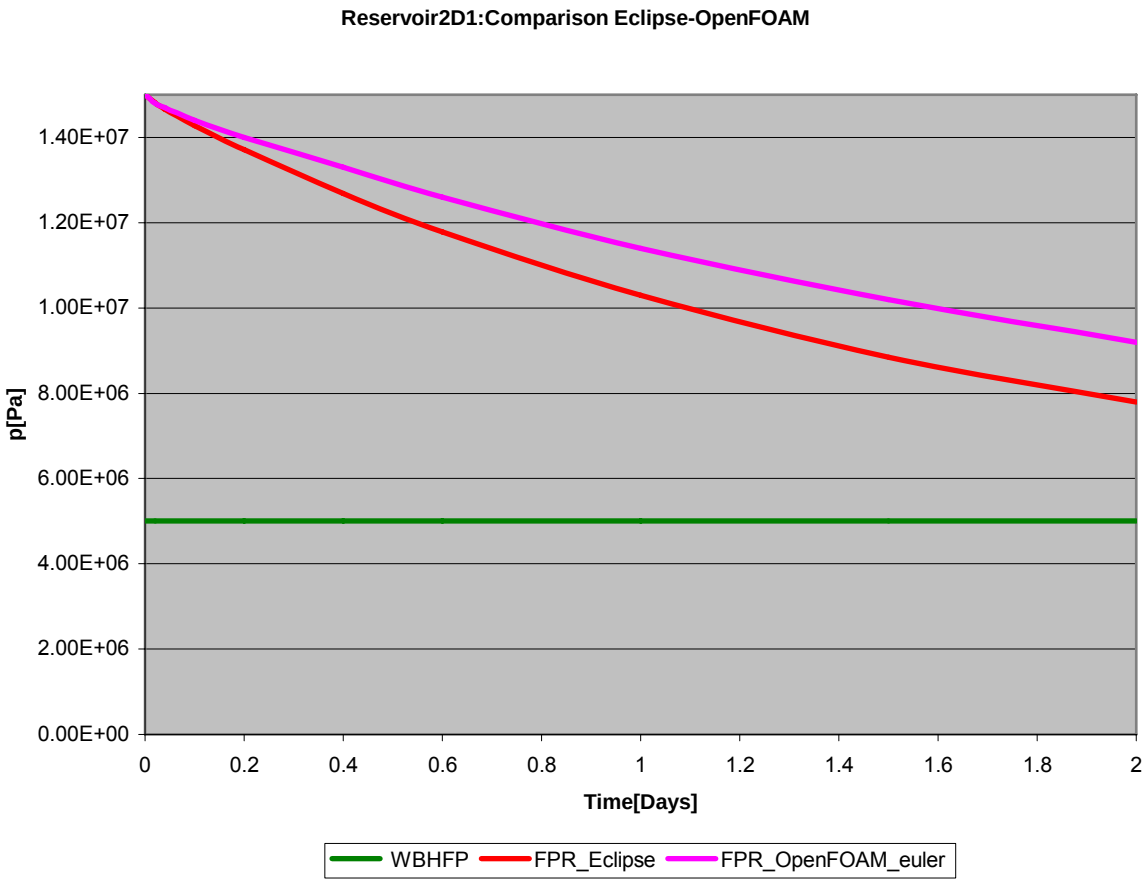


Figure 5.13: Field pressure comparison between Eclipse and OpenFOAM for case 1

It comes out from this first investigation that OpenFOAM and Eclipse field pressure decrease more or less in the same manner. But after a short time of production, Eclipse model depletes more quickly as OpenFOAM one.

The Field pressure of the Eclipse model will reach the Bottom Hole Flowing Pressure before that of OpenFOAM.

We observe a pressure difference up to 15% which is significant and recommands more investigation of this case.

In the last investigation, we used the firt order time dicretization(Euler schemes).

Now others investigations were performed which taken in consideration also the second time derivative (Backward schemes) and the fourth oder of the gradient discretization (fourth schemes). However it was difficult to improve the results.

The next investigation in this case concerned a litte change in the well geometry

Instead of producing through a ¼ of the well at the left corner of the reservoir, we produce in a small squared corner of 0.1[m](well diameter) which has approximately the same cross section with the well of 0.1 [m] diameter.
Now producing in a small corner on 0.1[m] (Well diameter), all reservoir parameters being the same, we obtain at the end the following results:

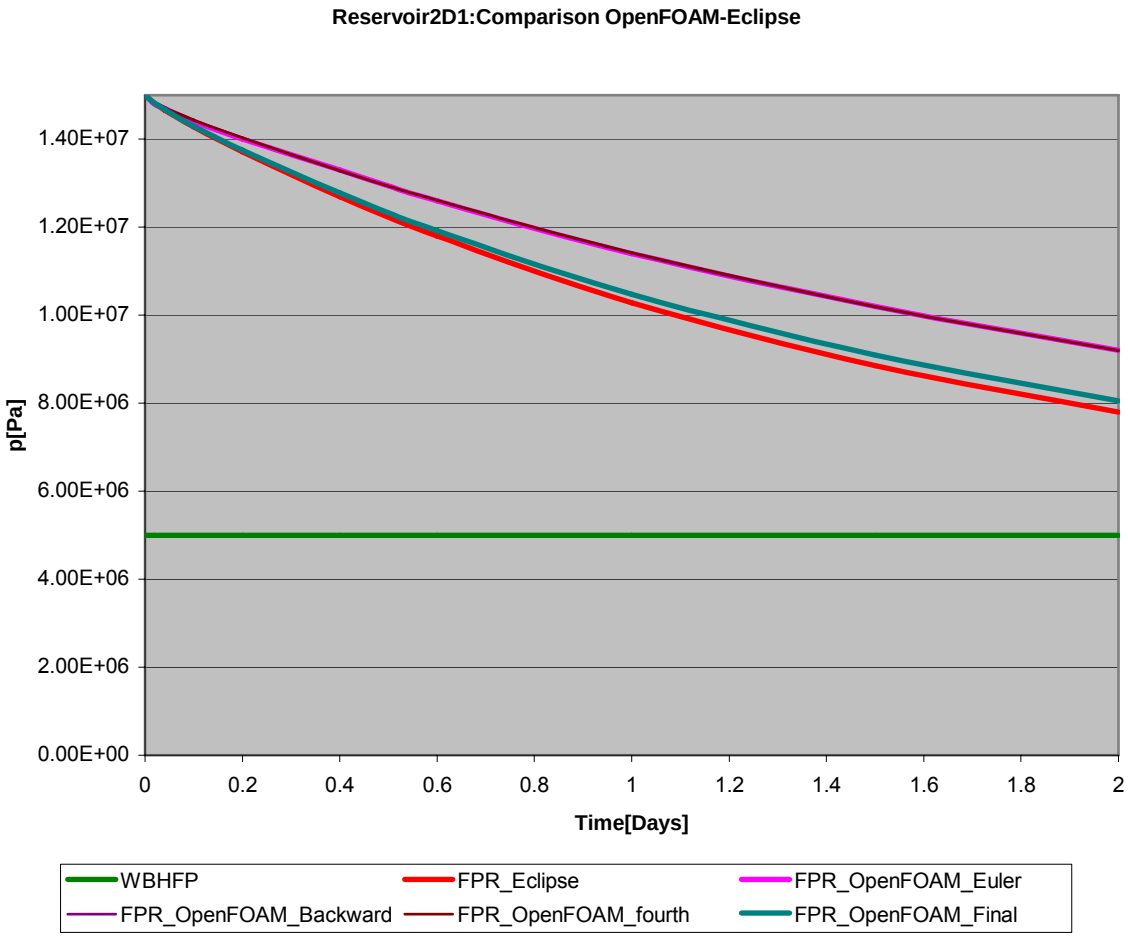
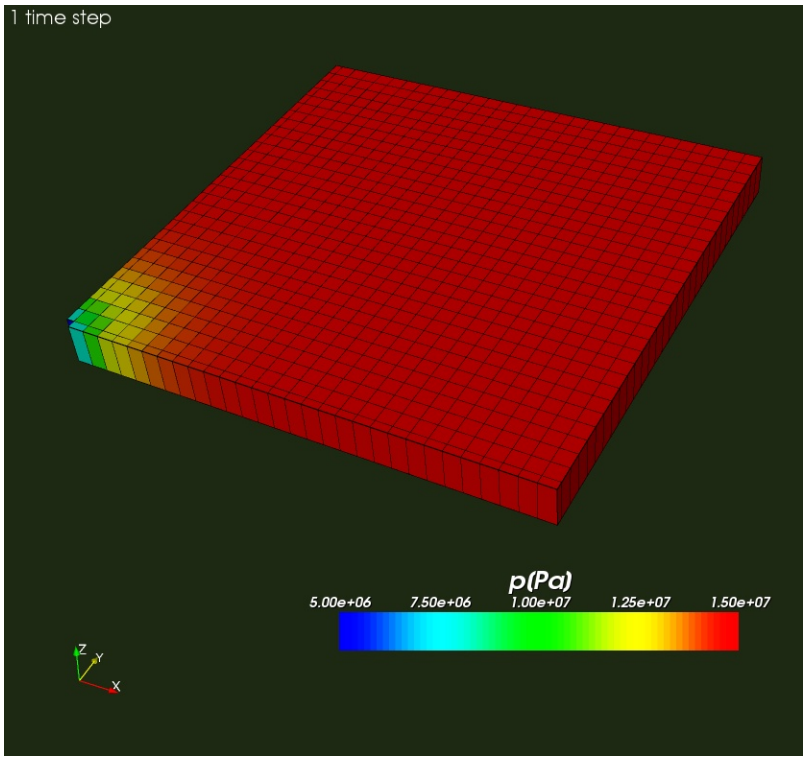


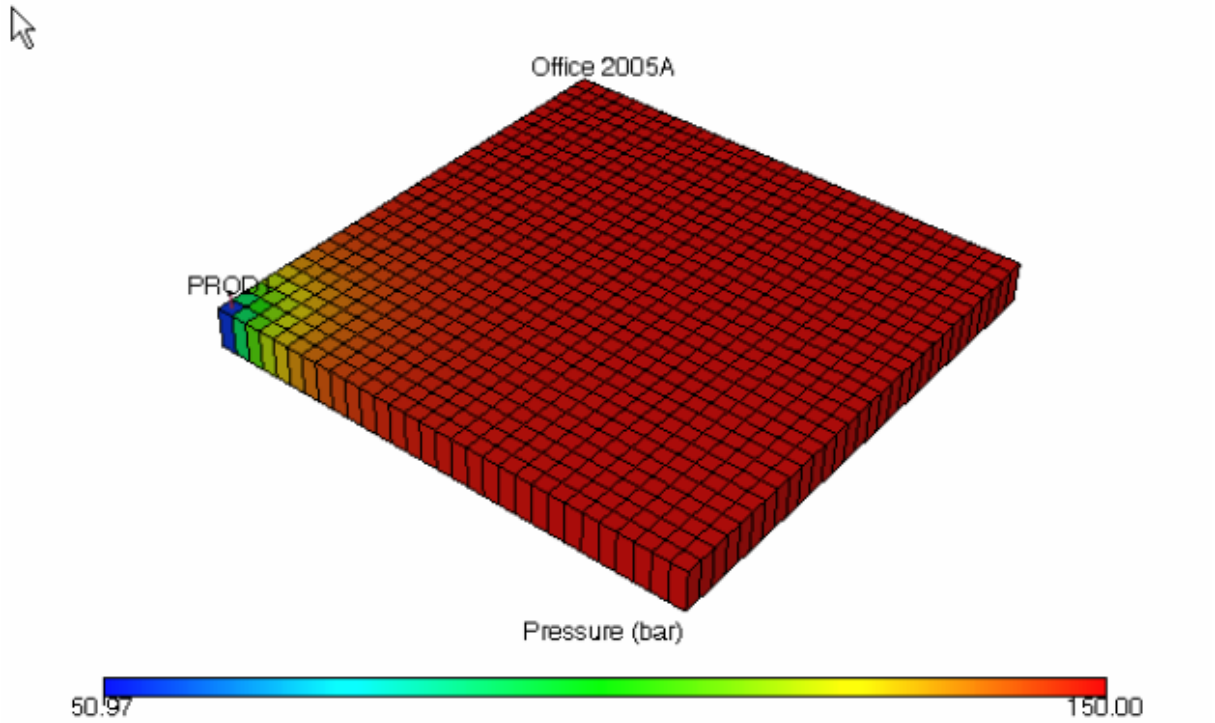
Figure 5.14:Final field pressure comparison between Eclipse and OpenFOAM for case 1

At the end we reach a pressure difference of around 2%. Which is realistic with the difference in pore volume at initial pressure.

Final pressure distribution of case 1 from OpenFOAM

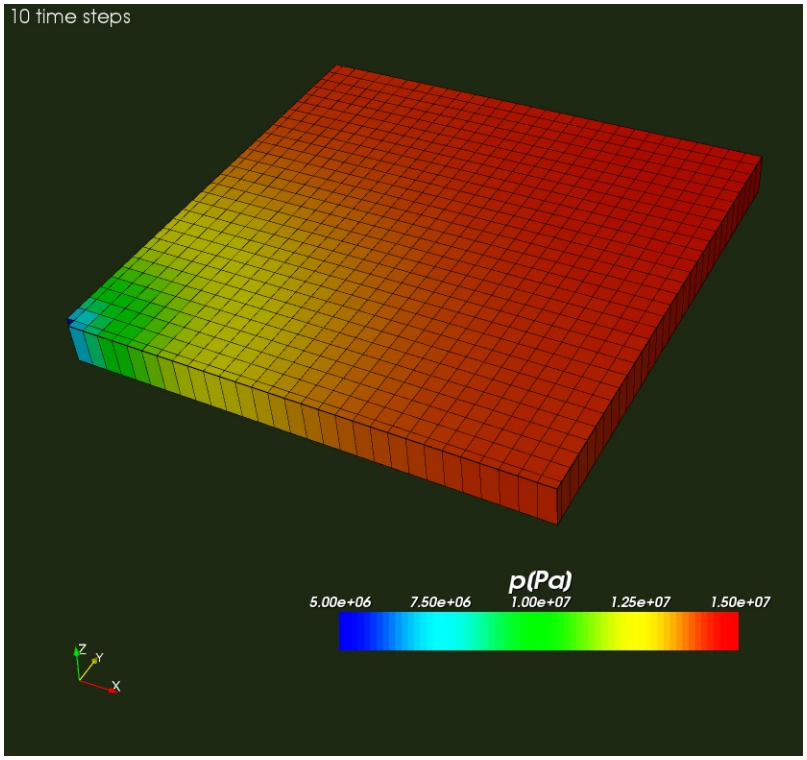


OpenFOAM

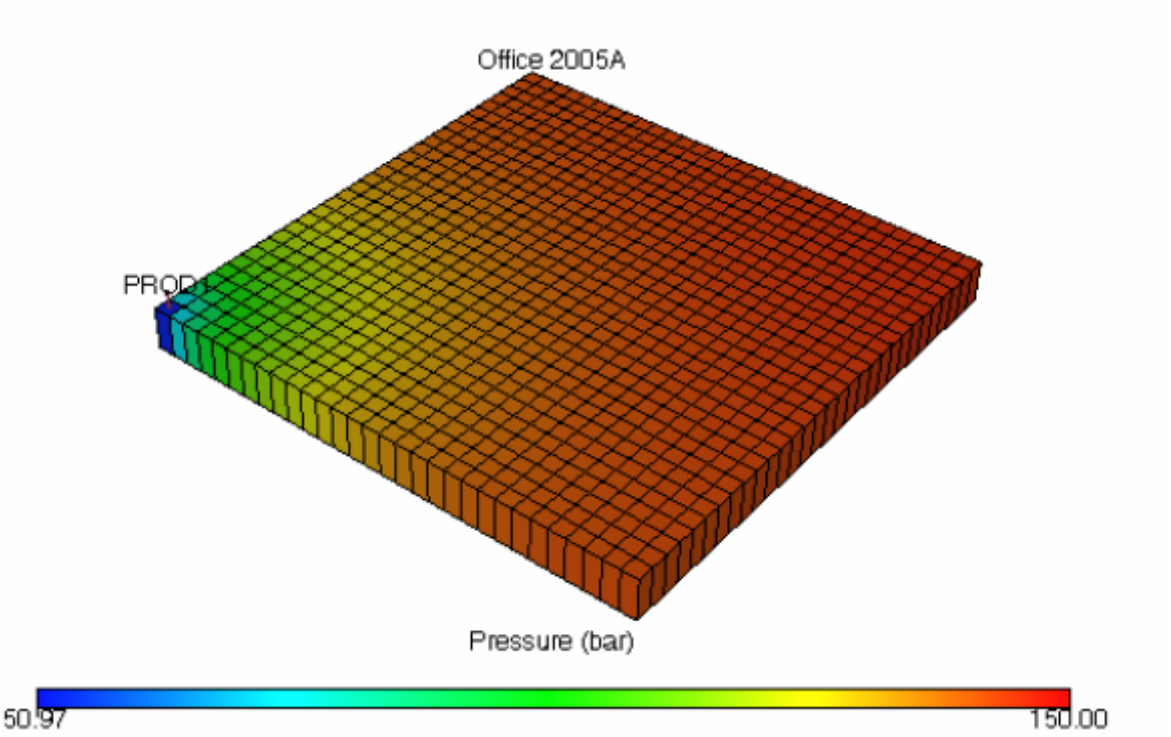


Eclipse

Figure 5.15 Pressure distribution after 1 time step (0.02 day)

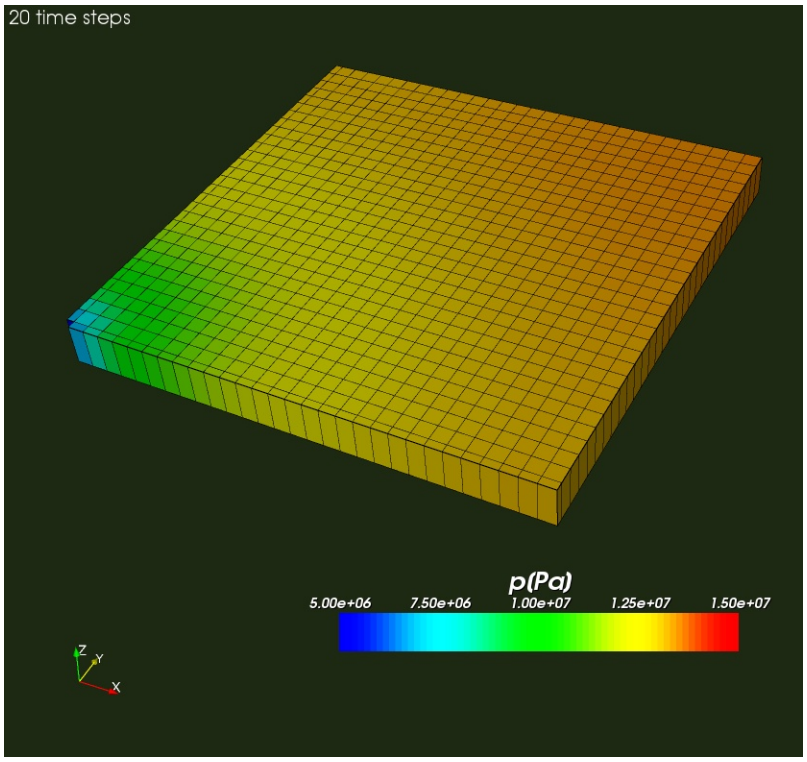


OpenFOAM

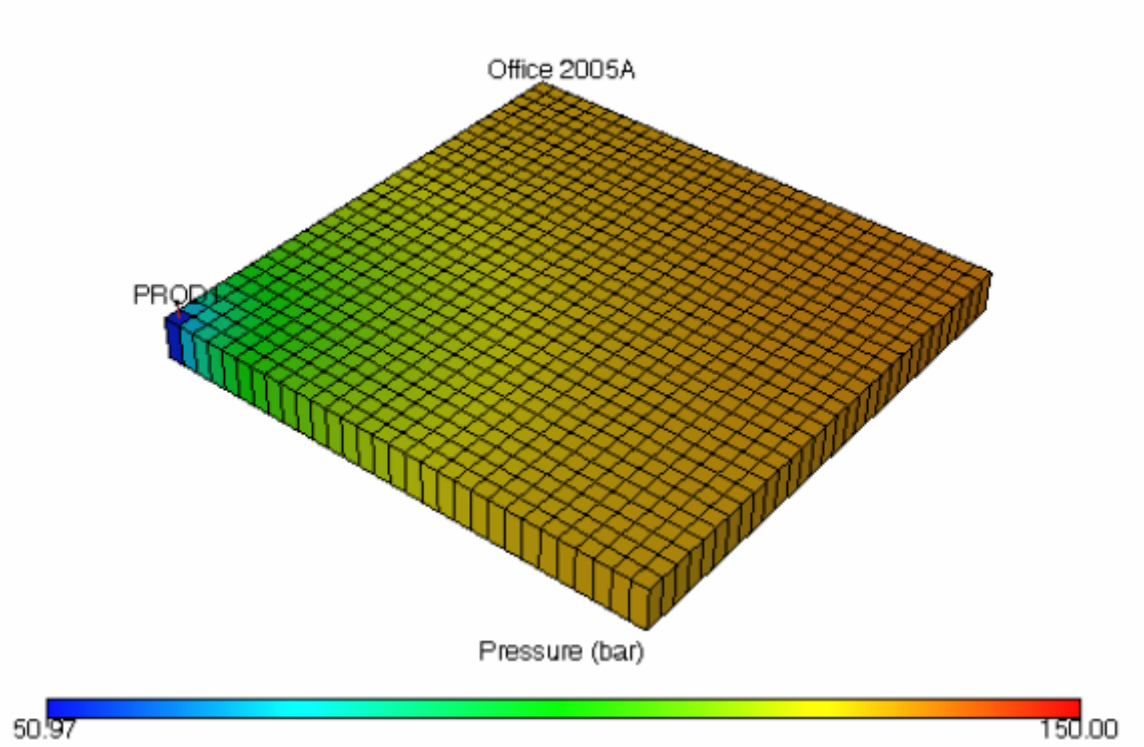


Eclipse

Figure 5.16 Pressure distribution after 10 time steps (0.2 day)

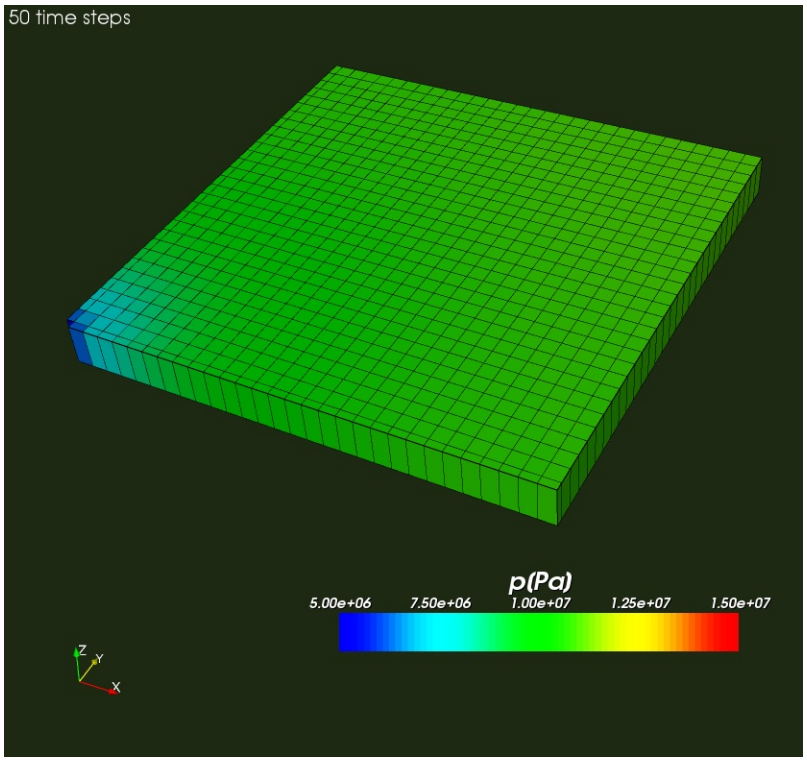


OpenFOAM

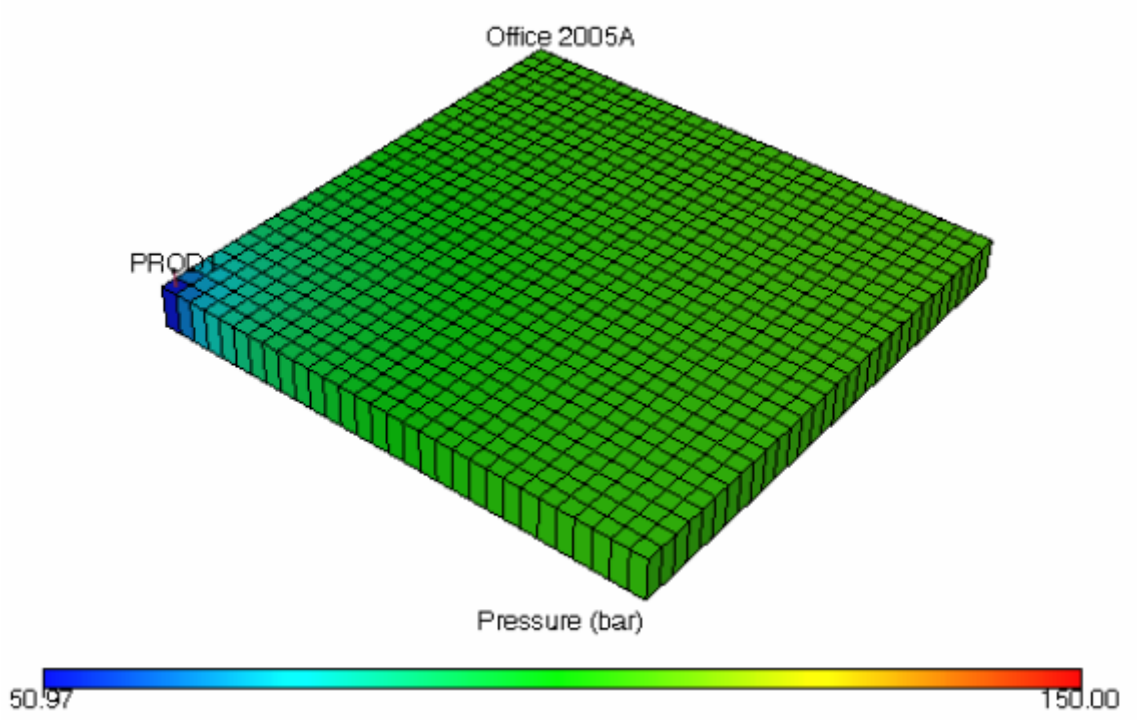


Eclipse

Figure 5.17 Pressure distribution after 20 time steps (0.04 day)

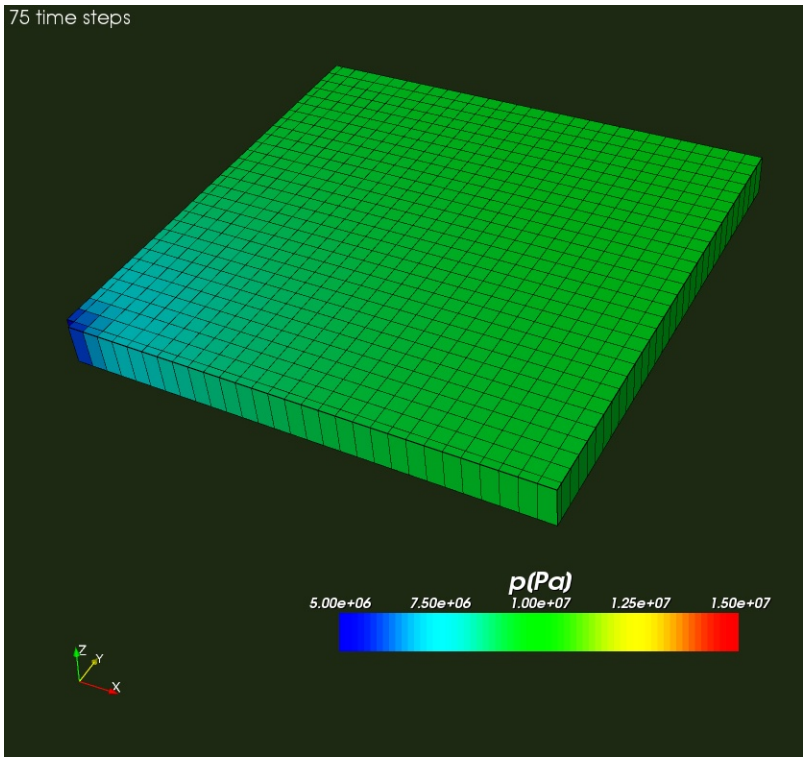


OpenFOAM

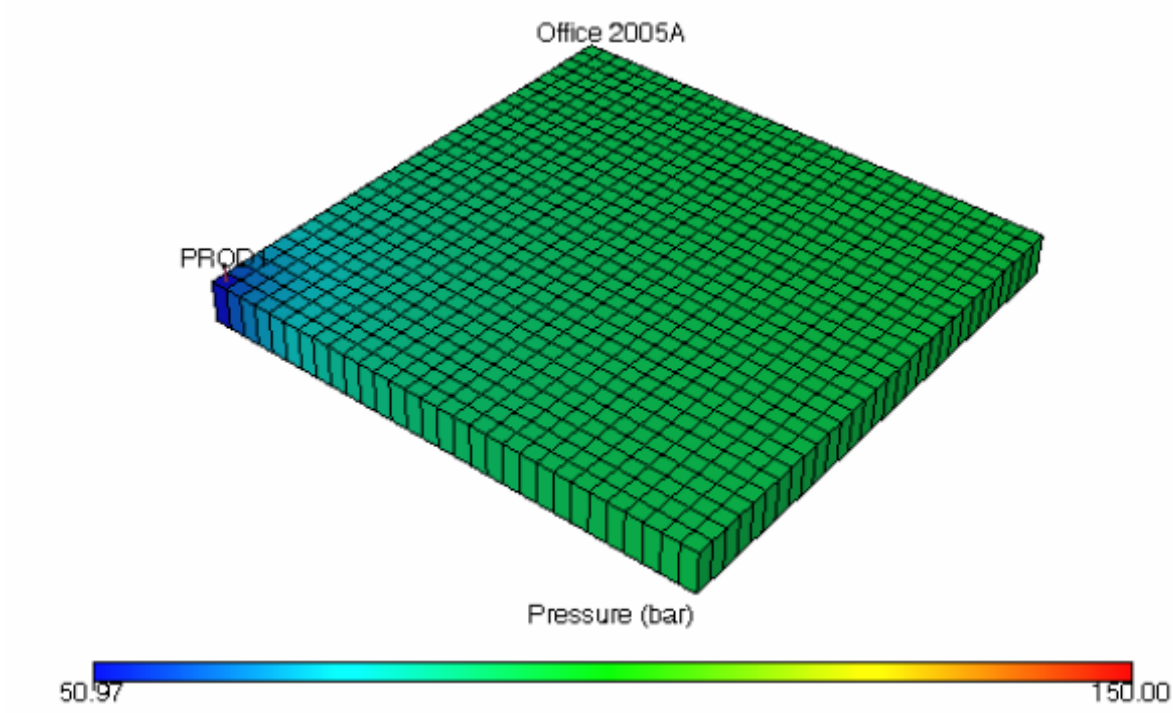


Eclipse

Figure 5.18 Pressure distribution after 50 time steps (1 day)

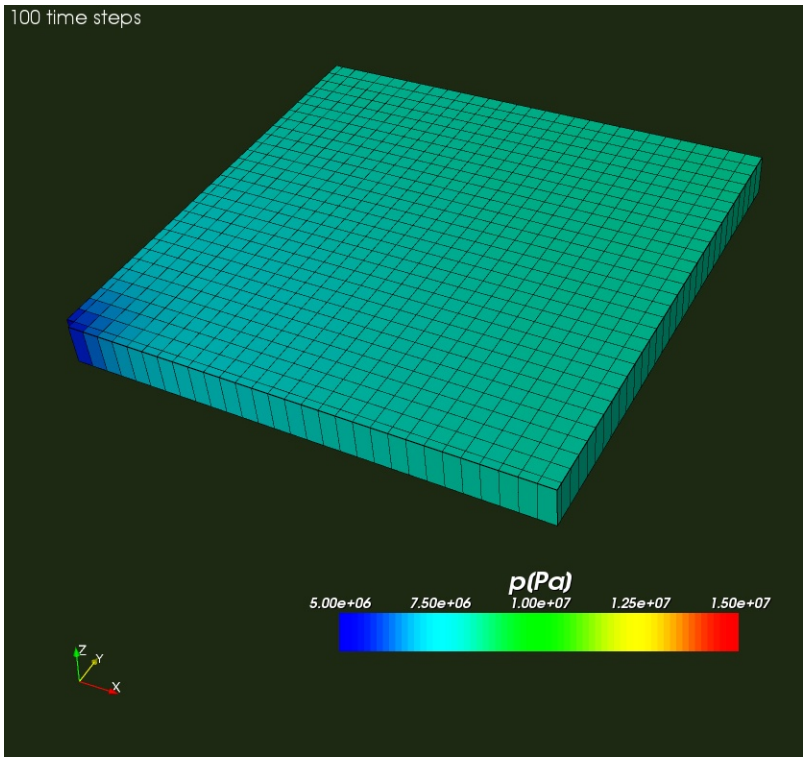


OpenFOAM

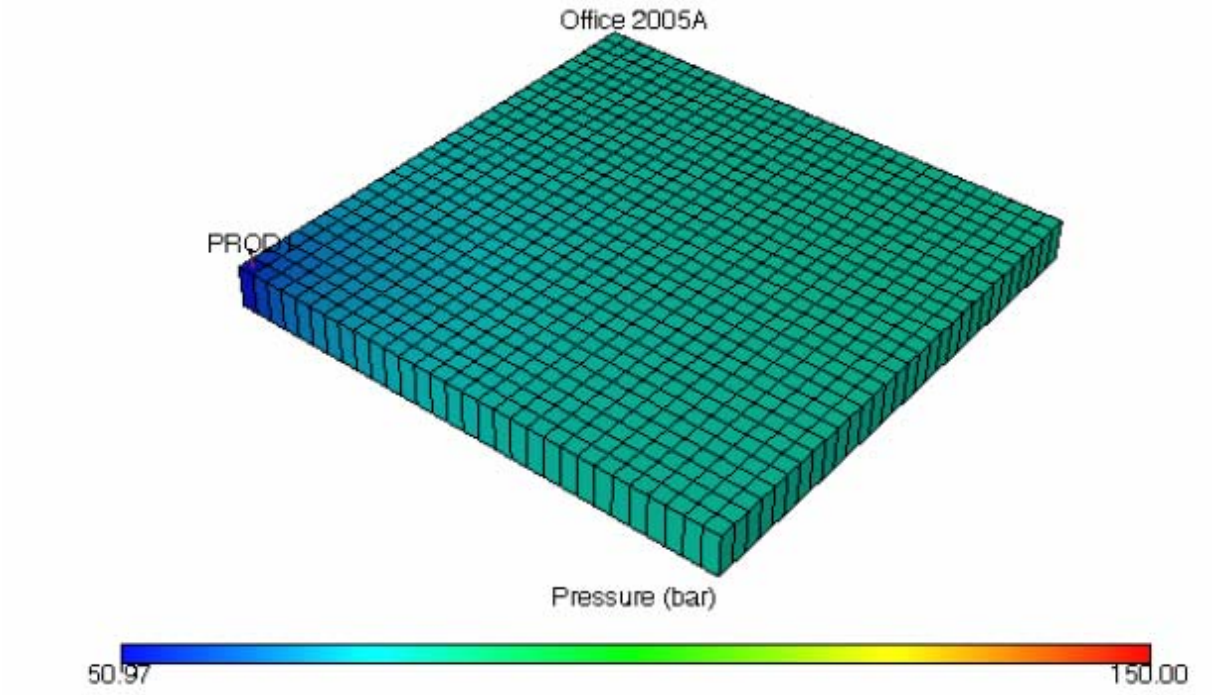


Eclipse

Figure 5.19 Pressure distribution after 75 time steps(1.5 days)



OpenFOAM



Eclipse

Figure 5.20 Pressure distribution after 100 time steps (2days)

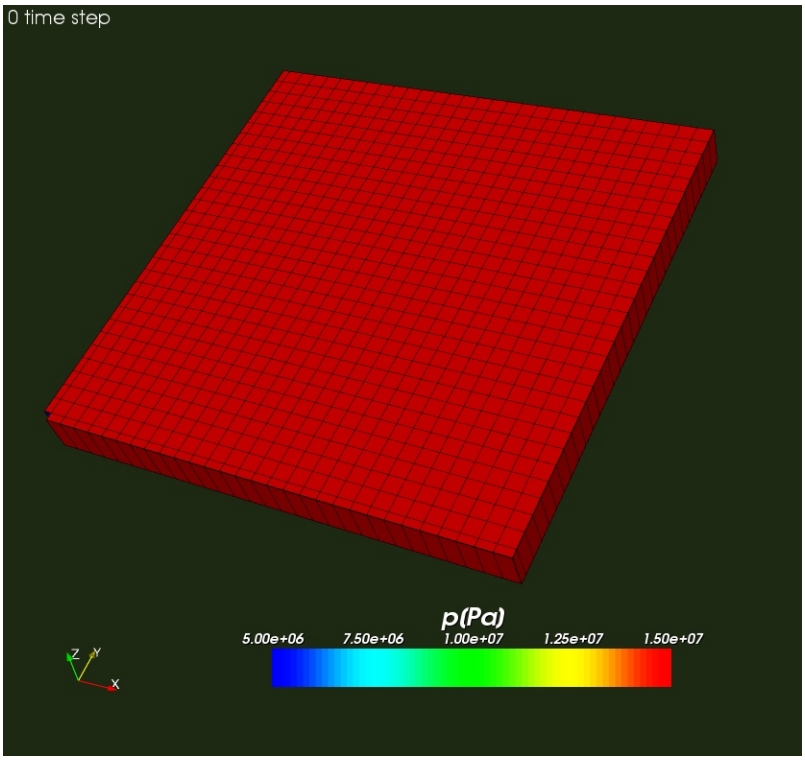
5.2.2 2nd Scenario:*Reservoir2D2 for one flow boundary with constant pressure*

A second scenario was created, called *Reservoir2D2*. Besides changing the boundary conditions, no changes were done to the model described above. We still produce at the left cartesian corner of the reservoir with 0.1m length in x and y direction. The difference to the first model is that we have three no flow boundaries and a constant pressure boundary at 150E+05[Pa] where in the first one all four boundaries were no flow boundaries.

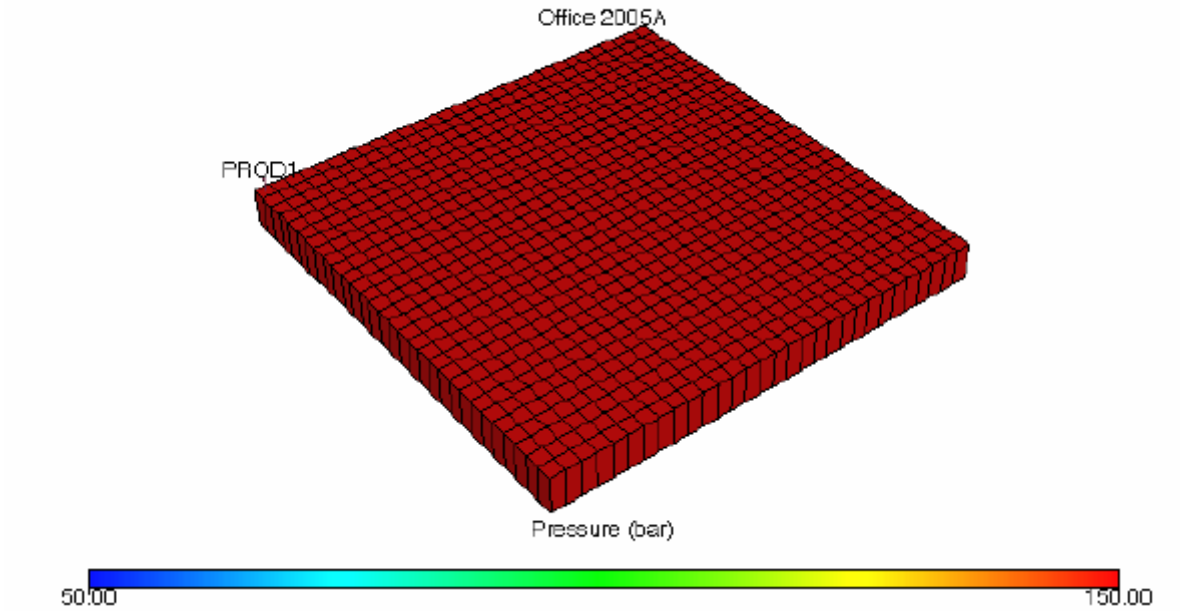
According to the defined parameters and to equation (50), we have again

$$Dp = 5.61\text{E-}04[m^2s^{-1}]$$

5.2.2.1 Results from OpenFOAM version 1.4 simulator and Eclipse



OpenFOAM

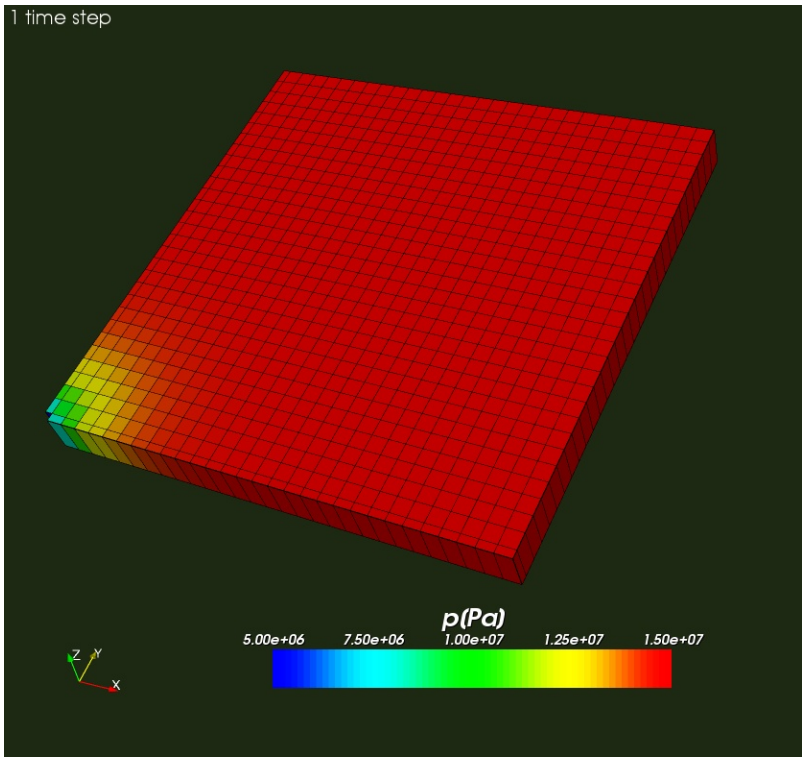


Eclipse

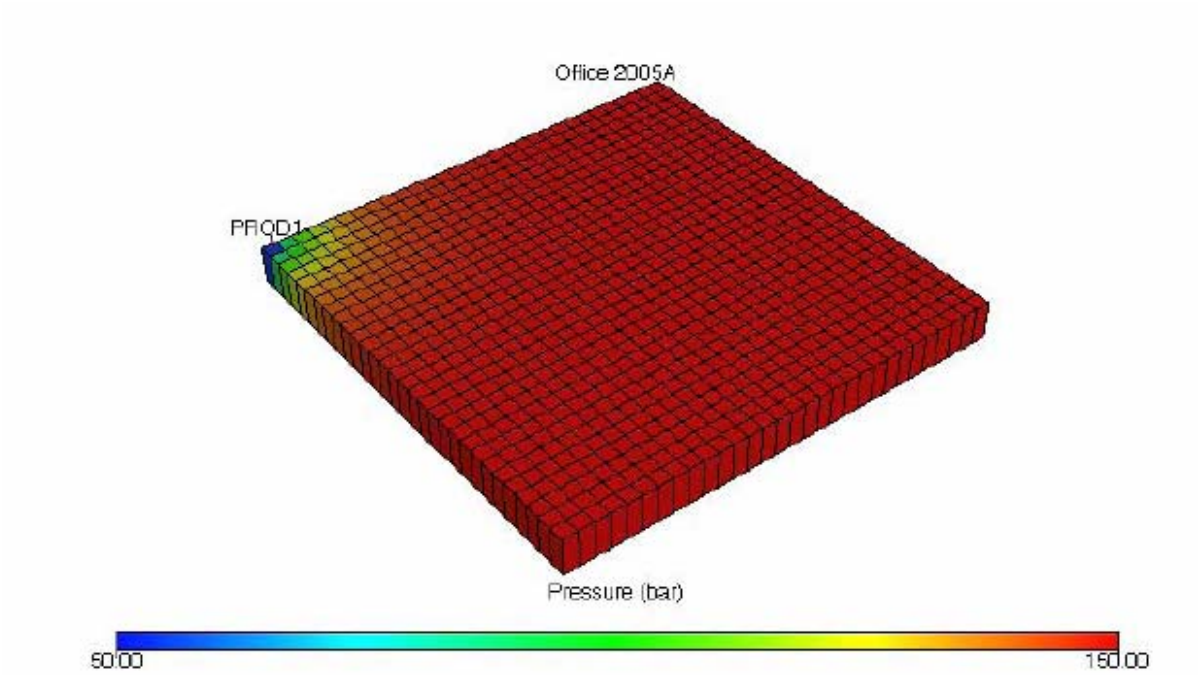
Figure 5.21: Initial pressure distribution

All cells have the same initial pressure of 150E+05[Pa]. The production well is located at the left corner.

During the whole calculation, the well was operated with a constant bottom hole flowing pressure of 50E+05[Pa].

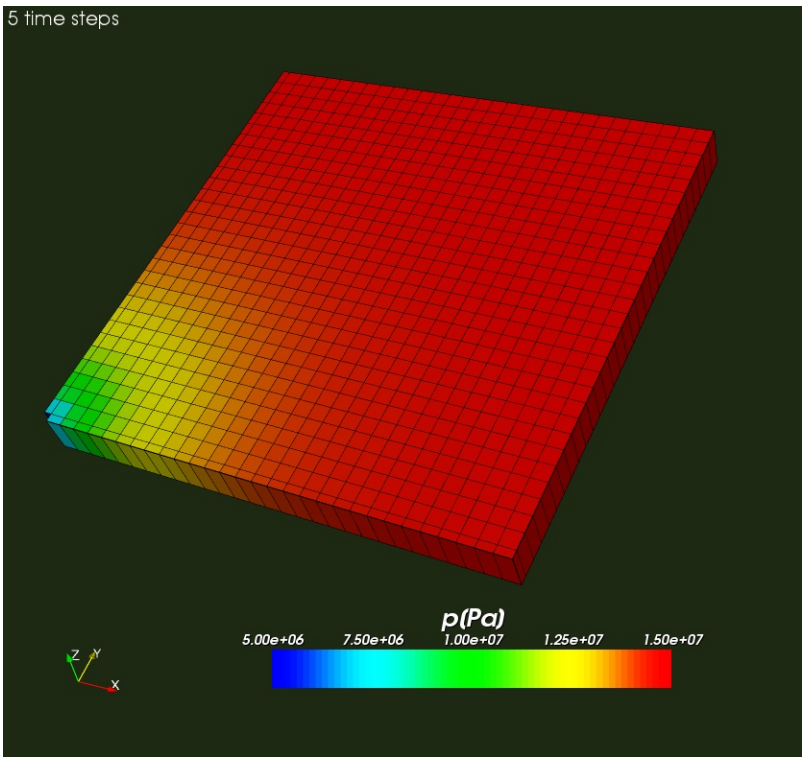


OpenFOAM

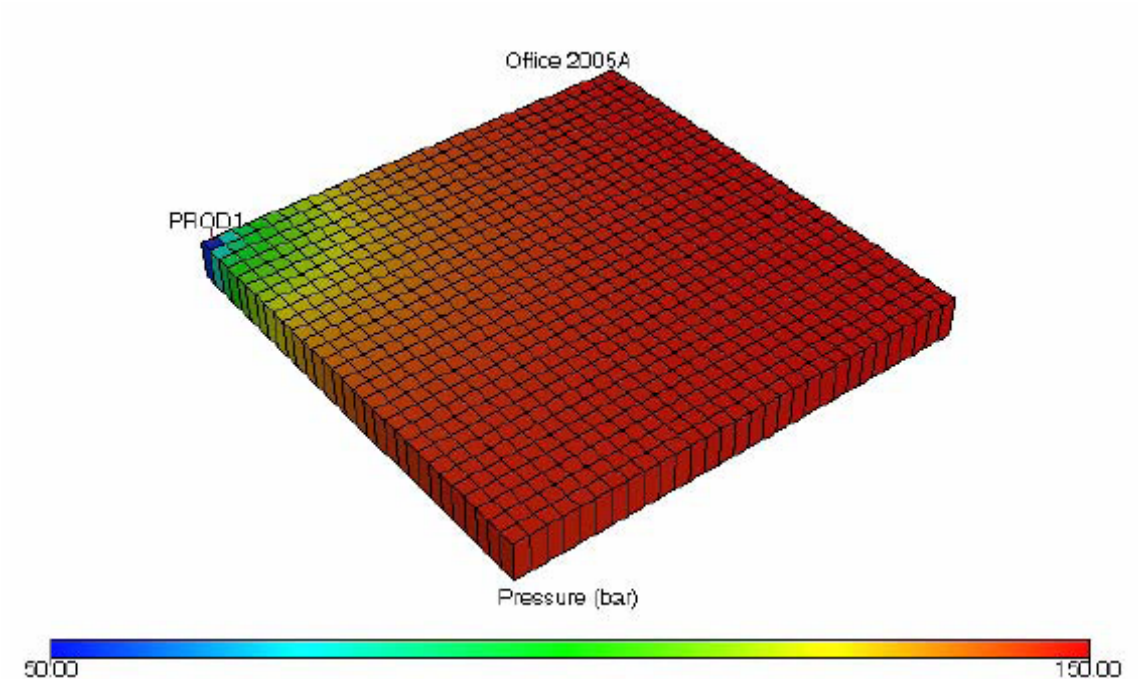


Eclipse

Figure 5.22: Pressure distribution after 1 time step (0.02 day)

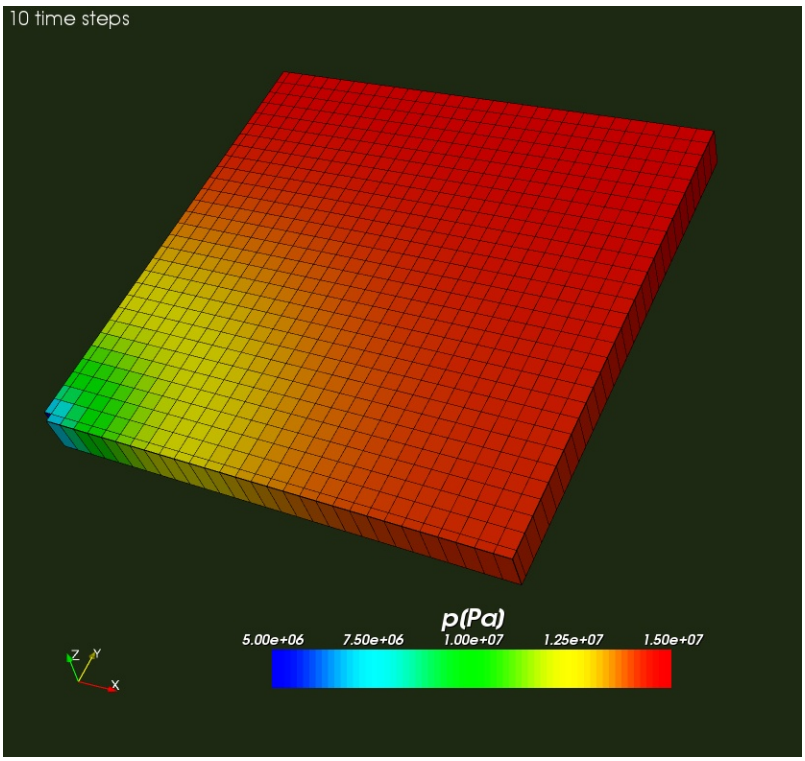


OpenFOAM

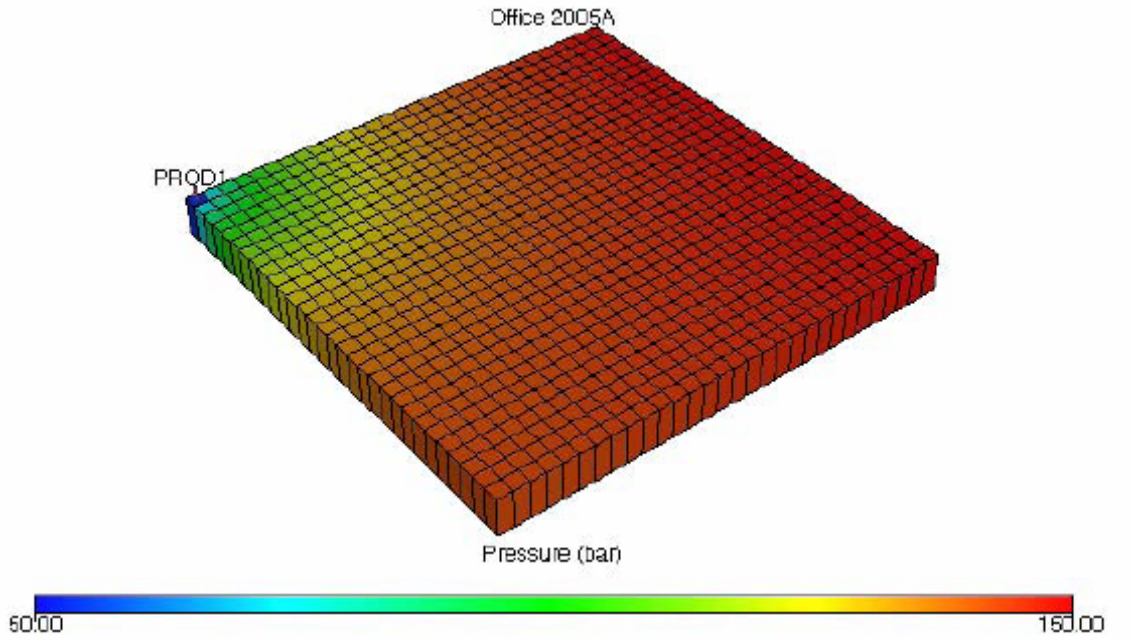


Eclipse

Figure 5.23: Pressure distribution after 5 time steps (0.1 day)

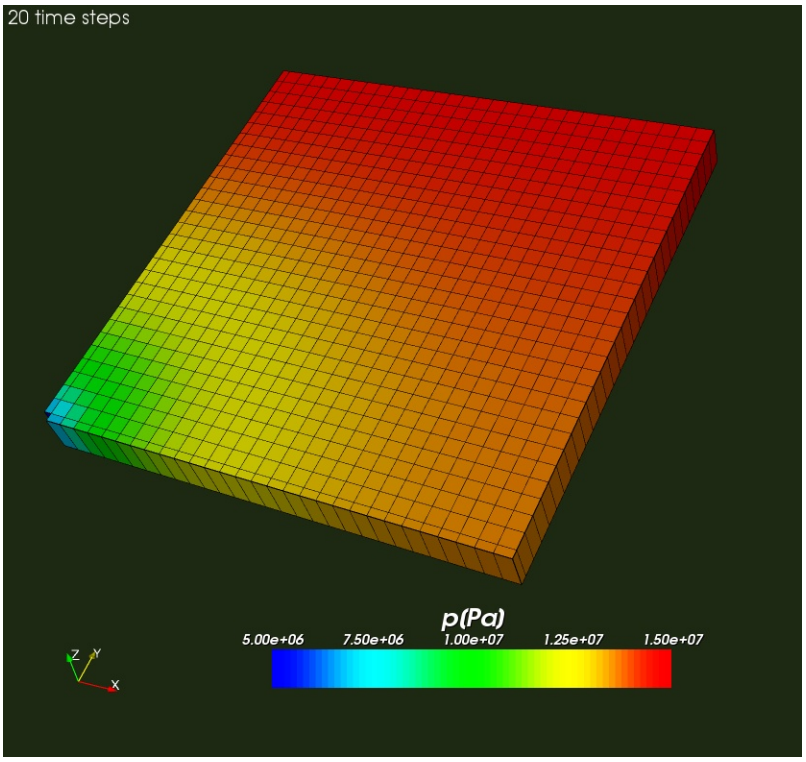


OpenFOAM

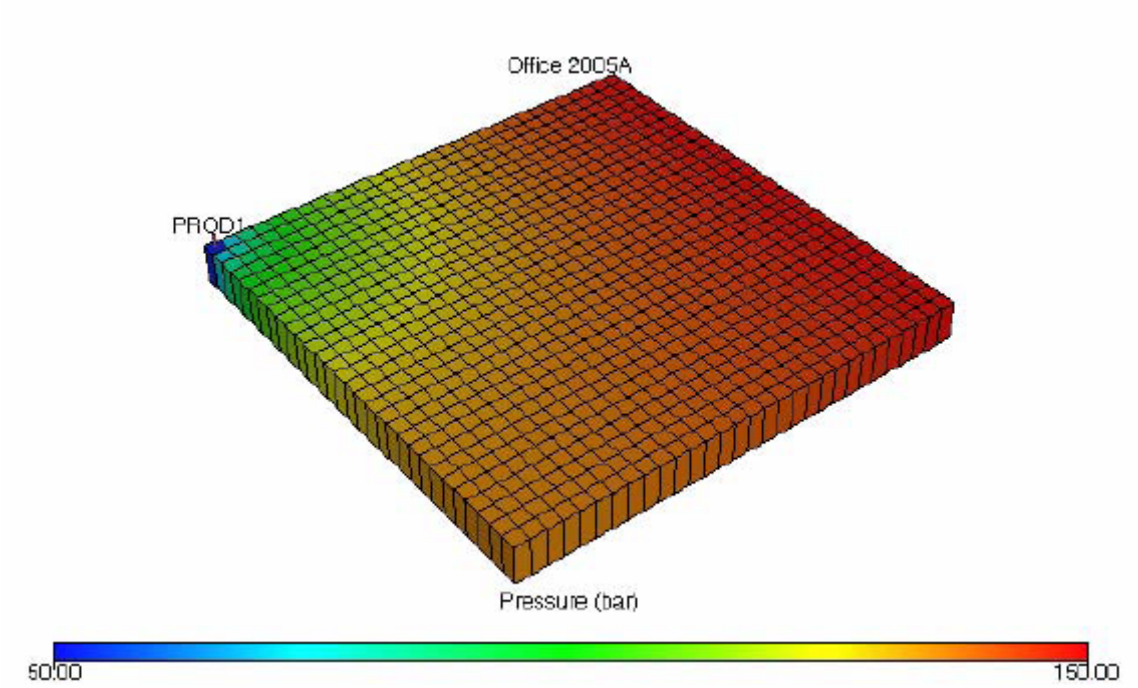


Eclipse

Figure 5.24: Pressure distribution after 10 time steps (0.2 day)

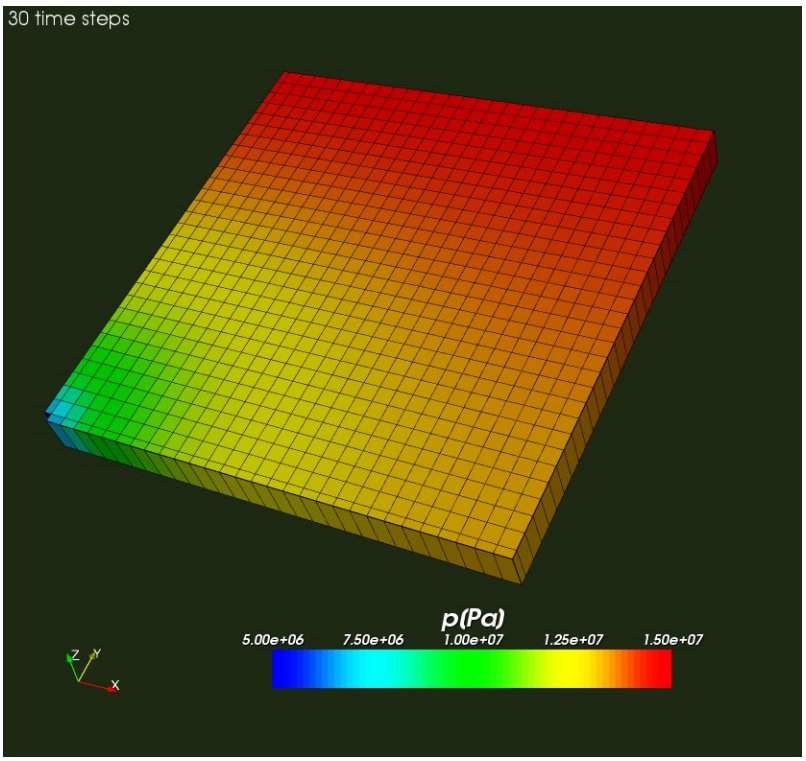


OpenFOAM

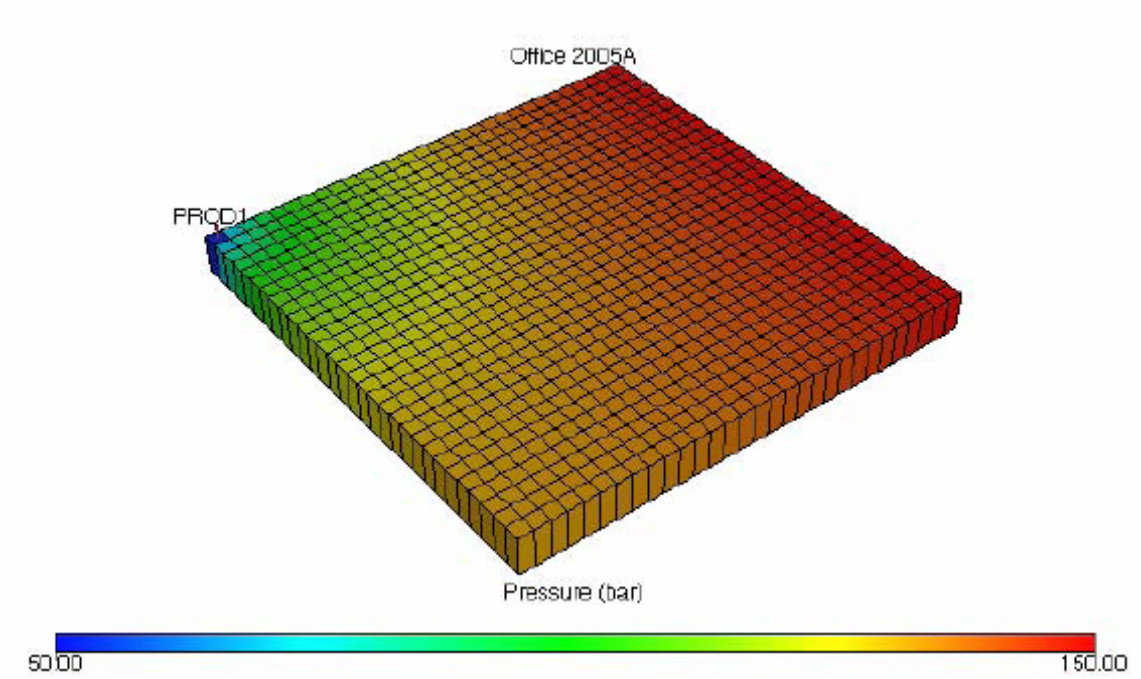


Eclipse

Figure 5.25: Pressure distribution after 20 time steps (0.4 day)

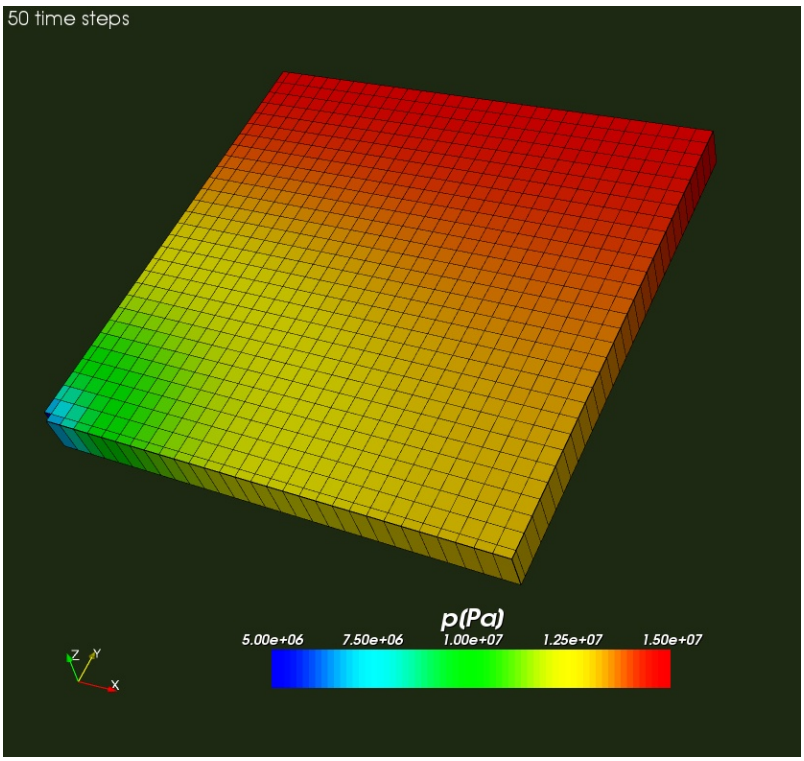


OpenFOAM

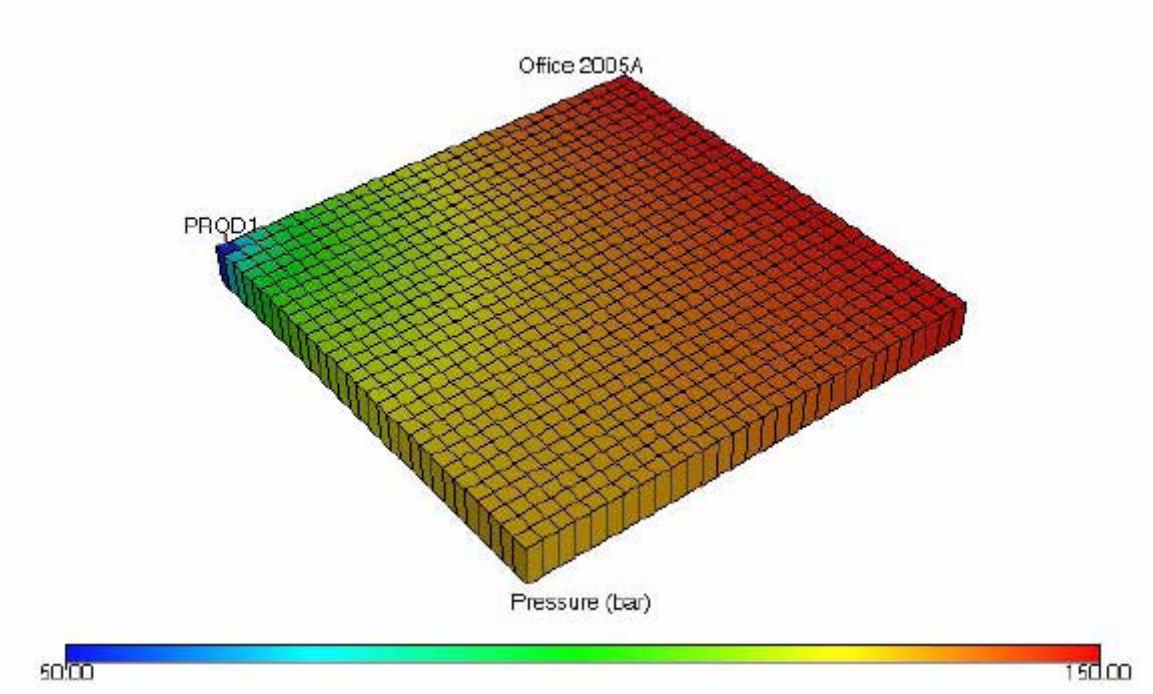


Eclipse

Figure 5.26: Pressure distribution after 30 time steps (0.6 day)

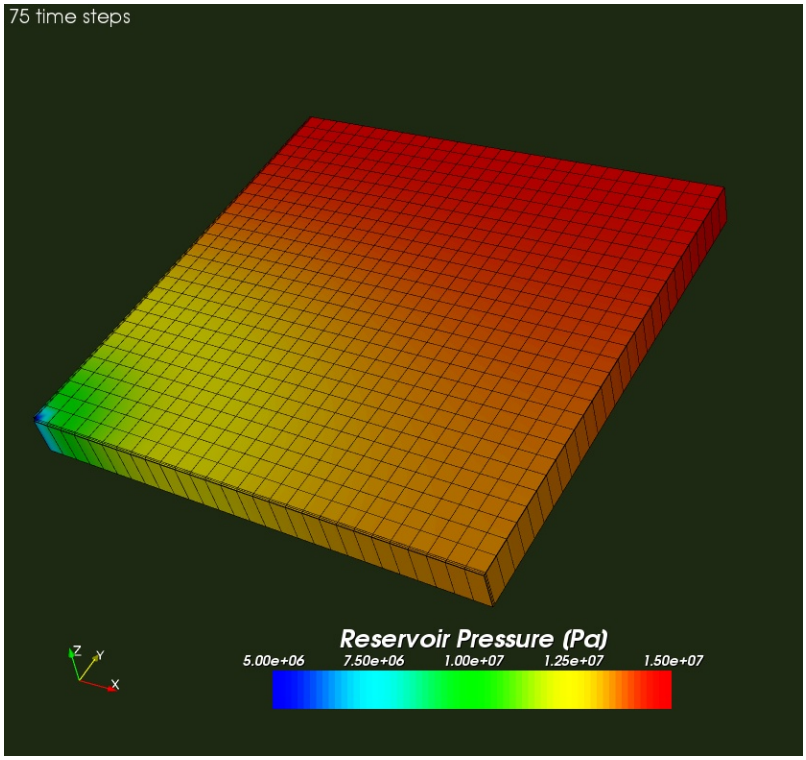


OpenFOAM

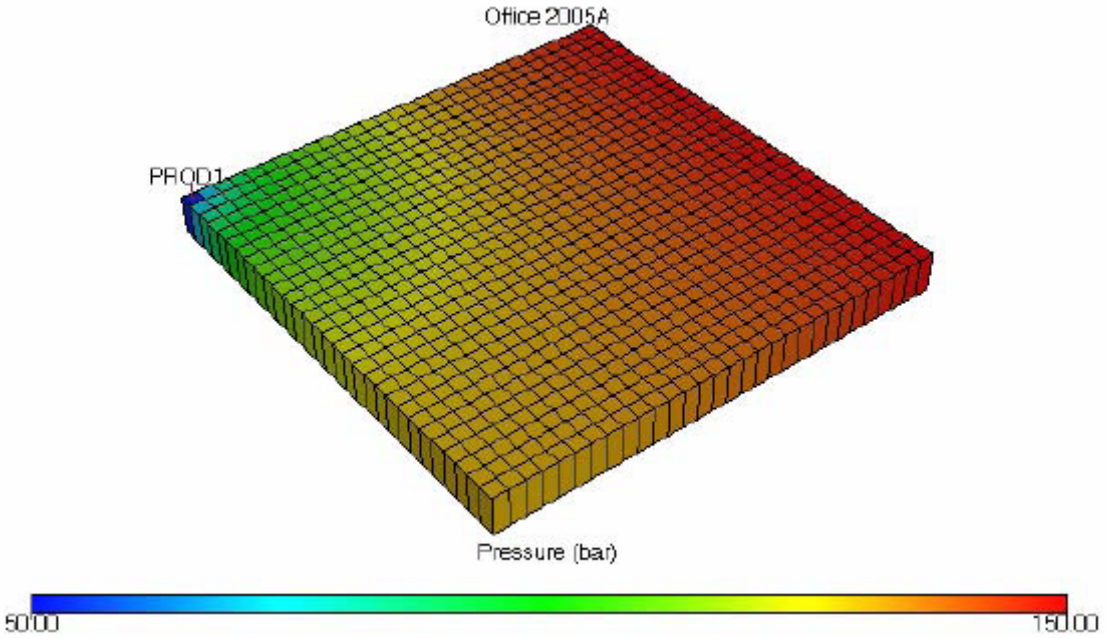


Eclipse

Figure 5.27: Pressure distribution after 50 time steps (1 day)

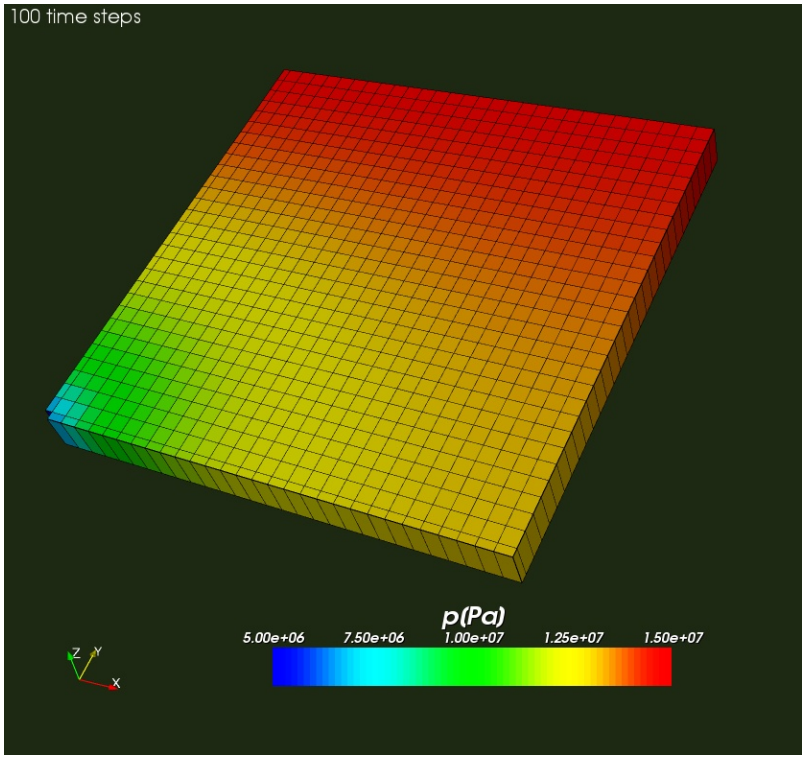


OpenFOAM

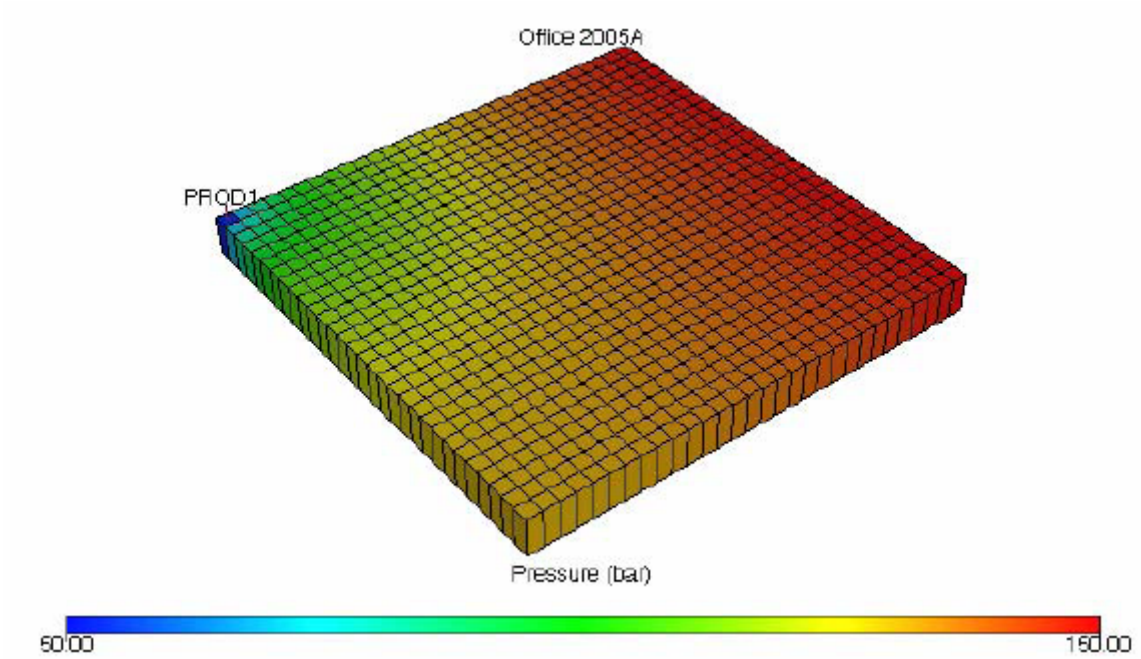


Eclipse

Figure 5.28: Pressure distribution after 75 time steps (1.5 days)



OpenFOAM



Eclipse

Figure 5.29: Pressure distribution after 100 time steps (2 days)

Performance plot from OpenFOAM

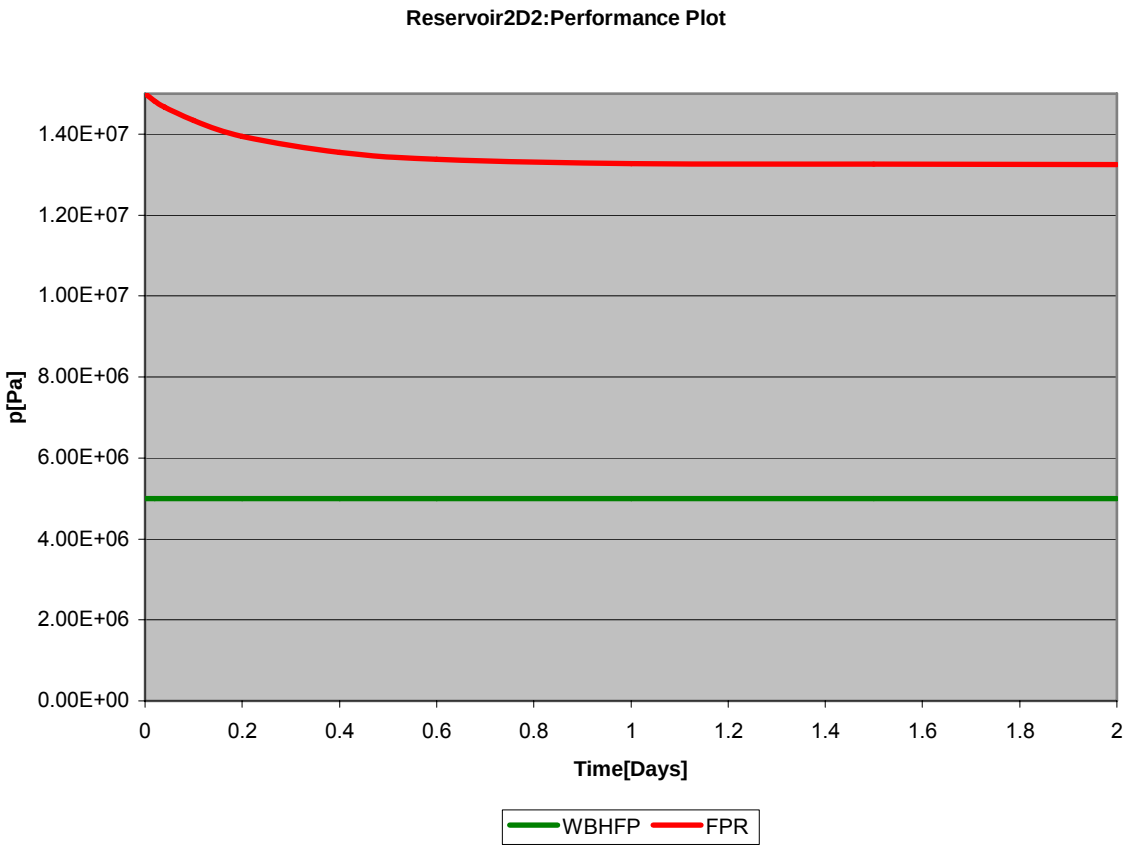


Figure 5.30: *Reservoir2D2* performance plot (red-average field pressure, green-bottom hole flowing pressure) after 2 days of production from OpenFOAM

Performance plot from Eclipse

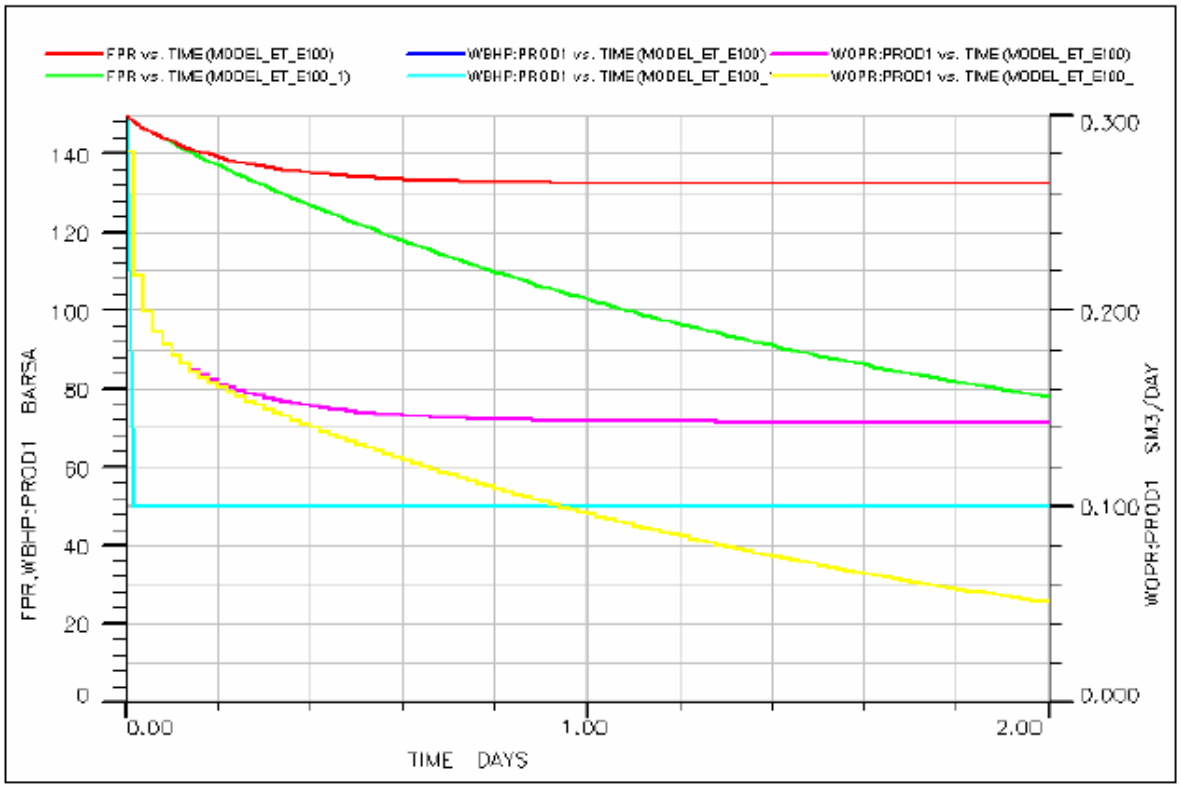


Figure 5.31 Performance plot showing differences between closed boundaries model and constant boundary Model from Eclipse

5.2.2.2 Comparison of results and analysis

As in case 1 , we will first compare the pore volume and the original oil in place at initial pressure of both results.

Pore volume [m^3]

From Eclipse the initial pore volume is automatically generated as PV = 5.48106[m^3].

In OpenFOAM, we have the following:

$$PV = V.\phi = V\phi^0\left[1 + c_\phi \Delta p\right] = V\phi^0$$

Because the reference and the top depth are equals.
which leads to

PV = 5.4[m^3].

This corresponds to 98.53% of Eclipse results. This corresponds on a difference of around 1.47%.

Original oil in place (OOIP) [sm^3]

From Eclipse the original oil in place is automatically generated as
OOIP = 5.769536[Sm^3]

In OpenFOAM, we have the following:

$$OOIP = \frac{PV}{B^0}$$

which leads to OOIP = 5.68 [Sm^3].
This difference is realistics because it is due to the difference occuring in pores volume.

Comparison of field pressure [Pa]

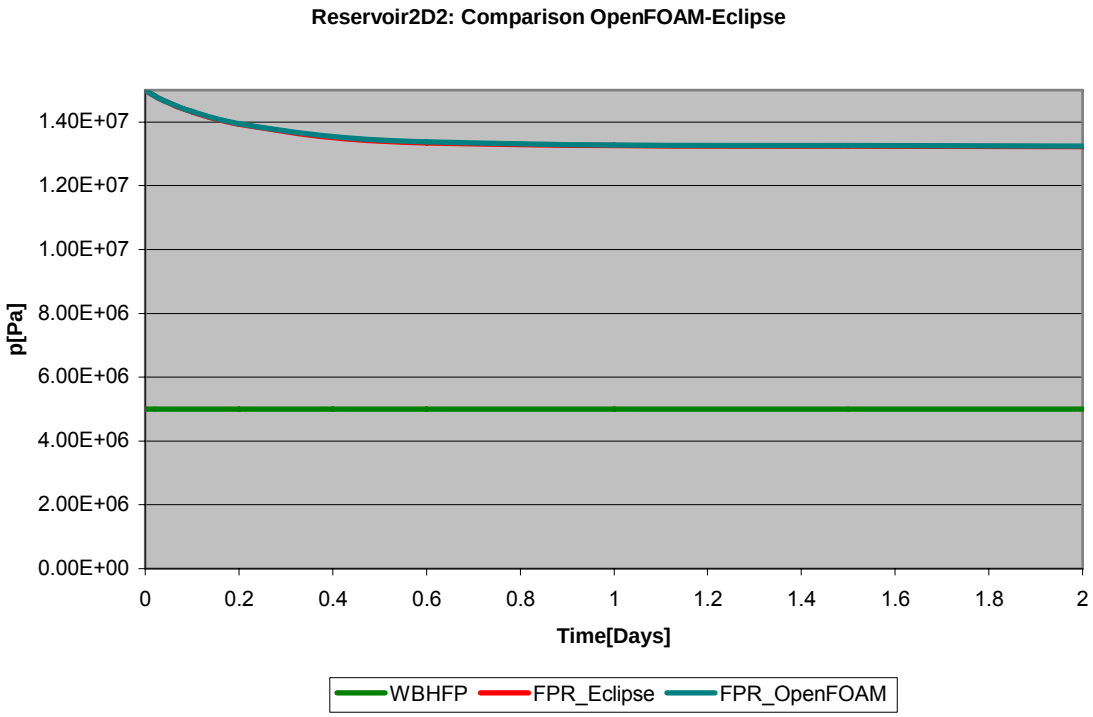


Figure 5.32: Final field pressure comparison between Eclipse and OpenFOAM for case 2

At the end we get the following results (See comparison table on the Appendix C).

This scenario provides us the same results for OpenFOAM and Eclipse simulators. After 100 time steps simulation we remain above the bottom hole flowing pressure with both simulators.

5.2.2.3 Field pressure comparison between *Reservoir2D1* and *Reservoir2D2*

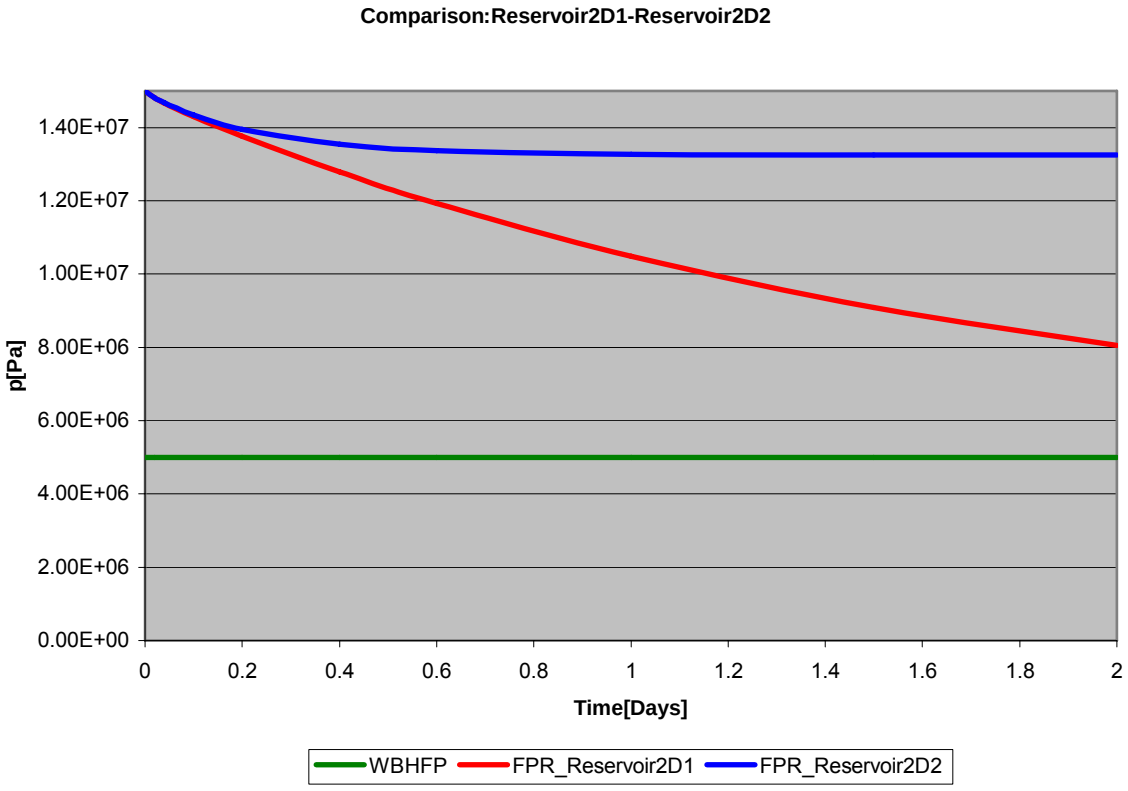


Figure 5.33: Field pressure comparison between *Reservoir2D1* and *Reservoir2D2*

5.2.3 3rd Scenario: *Reservoir2D2Incompressible for incompressible flow and one flow boundary with constant pressure*

The last scenario created in 2D, called *Reservoir2D2Incompressible,* investigated incompressible flow ($\rho = const$) and one flow boundary with constant pressure of 150 E+05[Pa]. We still produce at the left cartesian corner of the reservoir with 0.1m length in x and y direction.

According to the defined parameters and to equation (50), we have

$$Dp = 2.92\text{E-}03[m^2 s^{-1}]$$

5.2.3.1 Results from OpenFOAM version 1.4 simulator

After running the simulation, we obtain the following results:

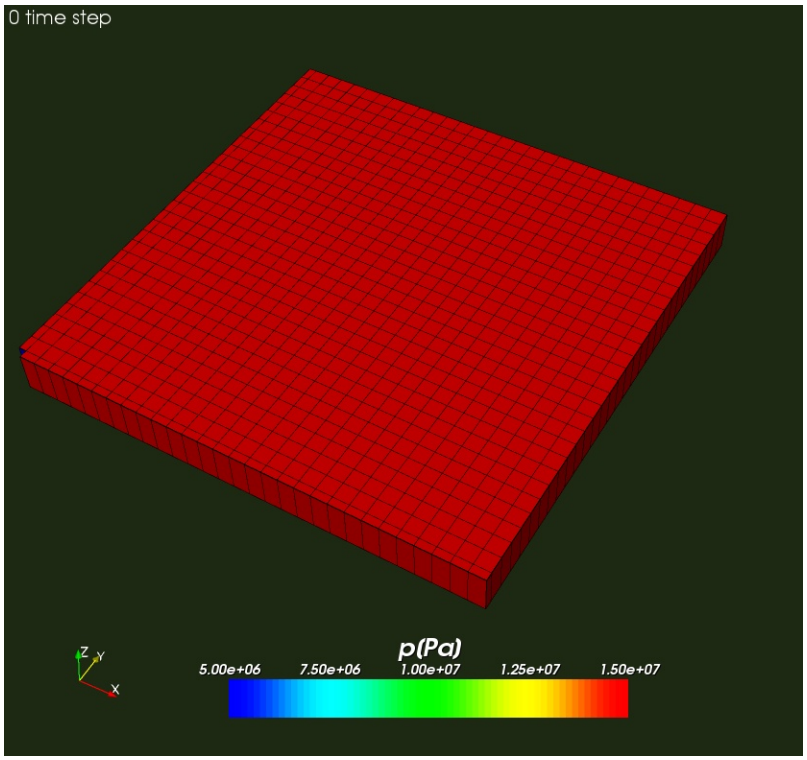


Figure 5.34: Initial pressure distribution

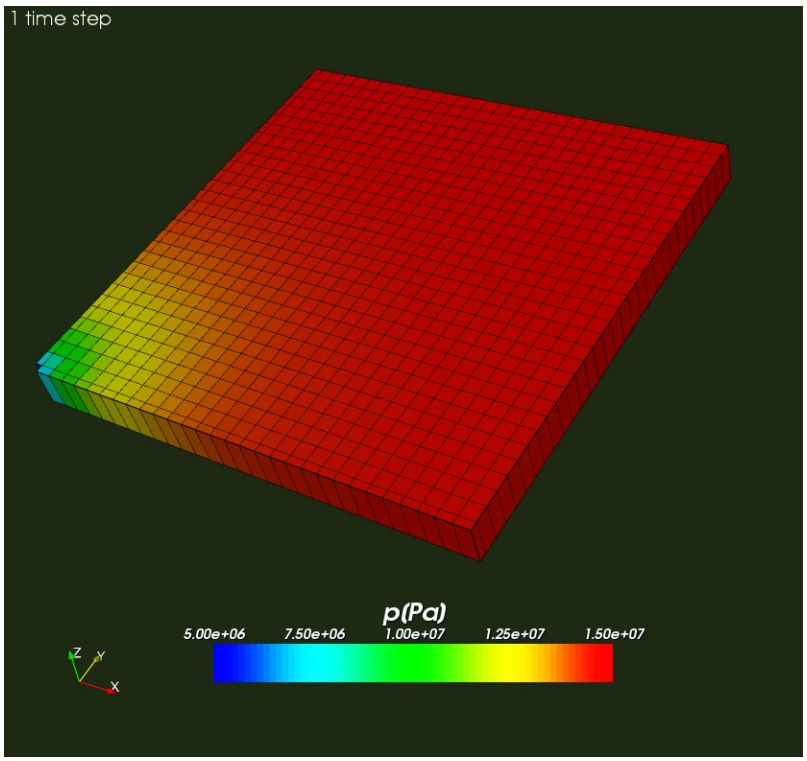


Figure 5.35: Pressure distribution after 1 time step (0.02 day).

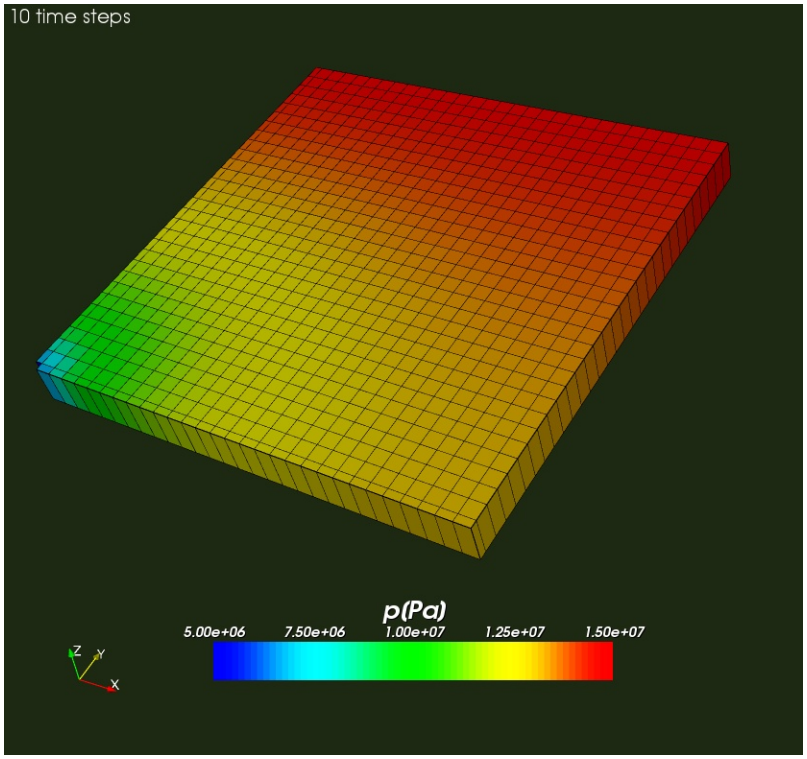


Figure 5.36: Pressure distribution after 10 time steps (0.2 day)

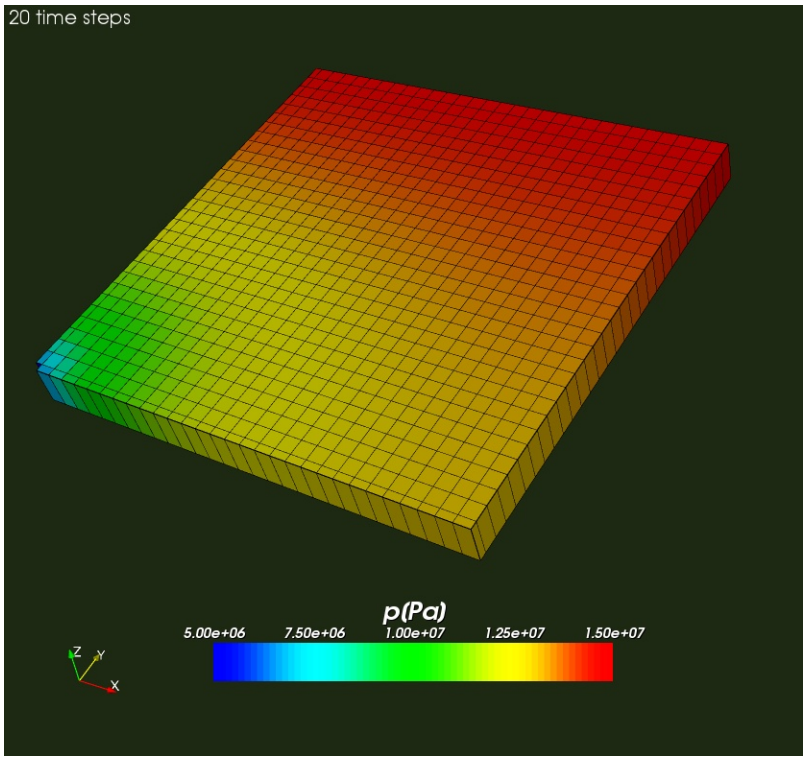


Figure 5.37: Pressure distribution after 20 time steps (0.4 day)

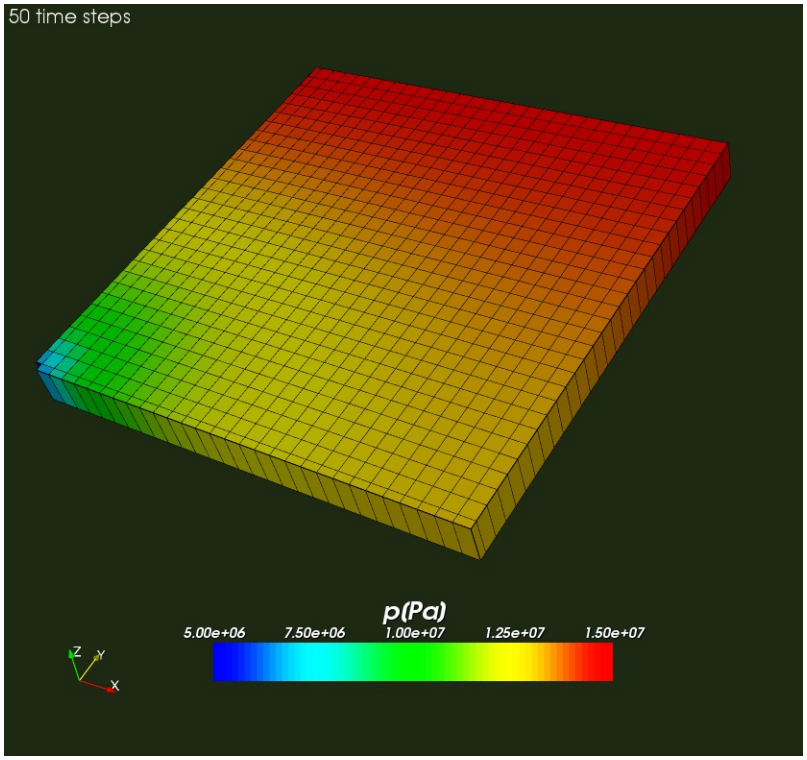


Figure 5.38: Pressure distribution after 50 time steps (1 day)

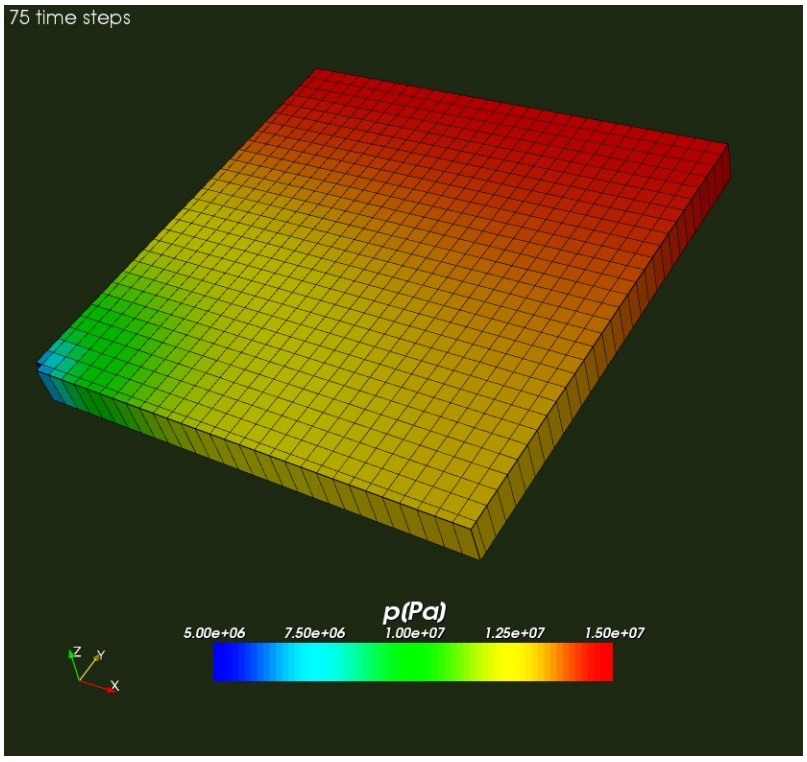


Figure 5.39: Pressure distribution after 75 time steps (1.5 day)

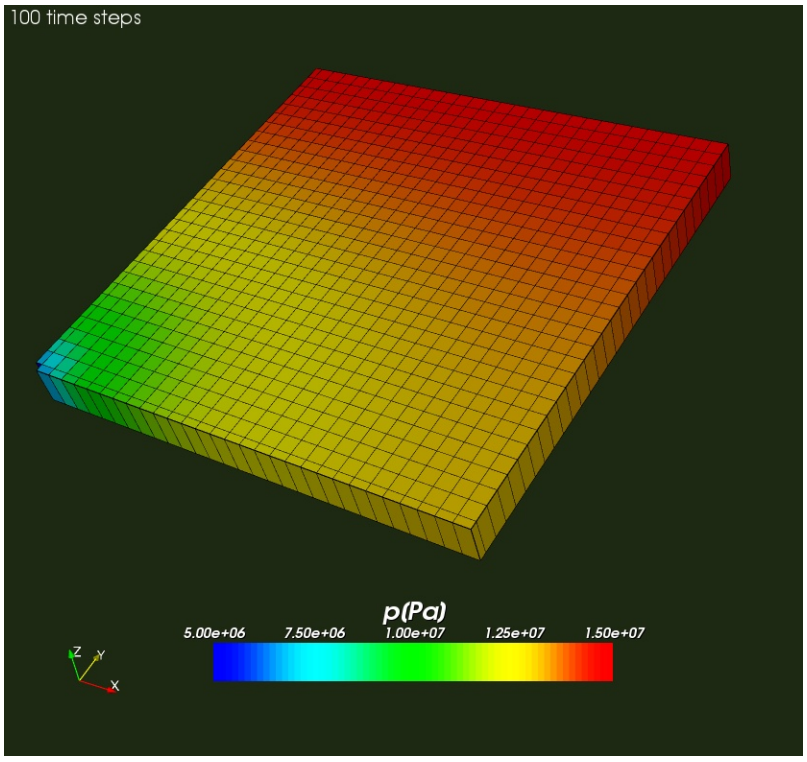


Figure 5.40: Pressure distribution after 100 time steps (2 days)

Performance plot.

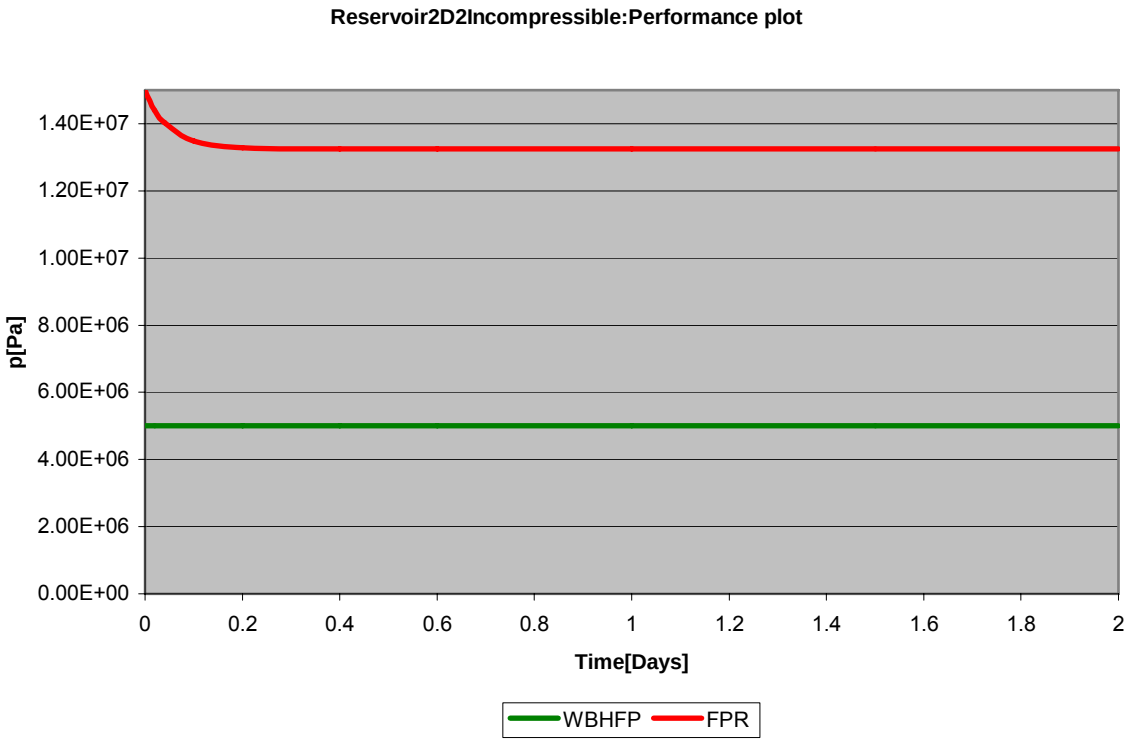


Figure 5.41:*Reservoir2D2Icompressible* performance plot (red-average field pressure, green- bottom hole flowing pressure) after 2 days of production

5.2.3.2 Field pressure comparison between *Reservoir2D2* and *Reservoir2D2Incompressible.*

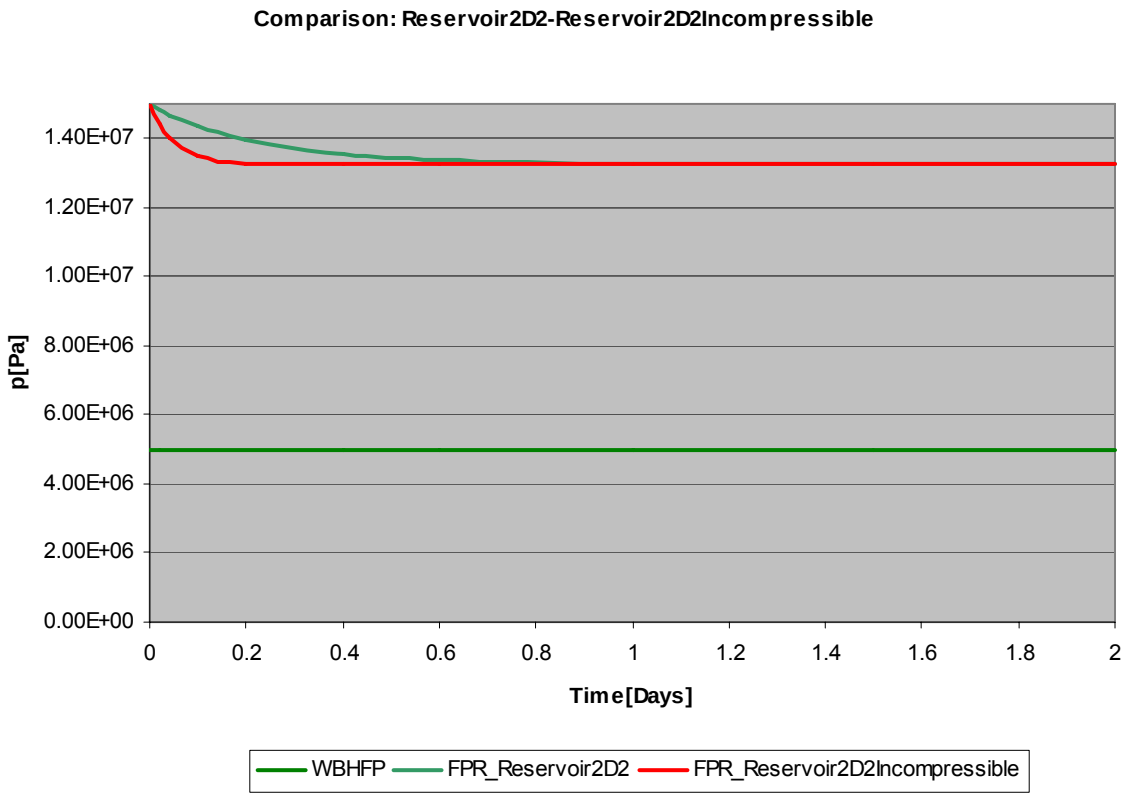


Figure 5.42: Field pressure comparison between *Reservoir2D2* and *Reservoir2D2Incompressible*

This pressure difference at the beginning of the production is realsitic because when we start production the pressure will drop more quickly for the incompressible case when it is assisted by fluid compressibility in the low compressible one.

6. Description and application of the 3D model

6.1 Description of the model called Reservoir3D

The model is a one layer reservoir that has a simple 3D geometry with no dipping or faults.

The grid model used is a three dimensional cartesian one. The size of the model is 50 times 52 times 14 meters. A cartesian grid with a resolution of 1 meter is used to discretize the model.This results in 50 blocks in x ,52 blocks in y-direction and 14 blocks in z-direction leading to a total model size of 36400 blocks.

To keep issues simple the model is set up with a homogeneous permeability and porosity distribution.

The density of oil is 800[kgm^{-3}] and its viscosity is 1.14 [cp].

The compressibility of the oil is modelled by the oil formation volume factor. Its numerical values are found on the appendix B.

The whole model is initialized with a pressure of 206.32E+05[Pa] valid at a reference depth of 4200[m].

Since only one phase (oil) is present the oil saturation equals unity for all cells.

Production takes place through six cells at different positions perforated at 13m from the top. The production cells are operated in a way that they are bottom hole flowing pressure controlled.

The bottom hole pressure target is 40E+05[Pa].

The properties of the fine grid model are as follows:

Initial Porosity $\phi = 0.3$ at reference pressure

Viscosity (dynamic) μ = 1.14cp = 1.14E-03 [$kgs^{-1}m^{-1}$] (constant throughout the run)

Oil density ρ = 800 [kgm^{-3}]

Permeability k = 1[mD] = E-15[m^2] (constant throughout the run)

Rock compressibility c_ϕ = E-09[1/Pa]

Oil compressiblity Because our oil is low compressible the compressibility is constant during pressure change.

6.1.1 Geometry of the model

Because the blockMesh directory is not flexible enough when building complex geometries, we have used the pre-processing tool Gambit to build the following geometry:

Gambit

Gambit is a geometry and mesh generation software. Its has several geometry and meshing tools in a powerful, flexible, tighly-integrated, and easy-to use interface.
Gambit reduces dramatically preprocessing times for many applications.
The Gambit software package helps to build mesh model for computational flow dynamics and other applications. Its main tasks are geometry creation, meshing, mesh examination, boundary and continuum zone assignment.

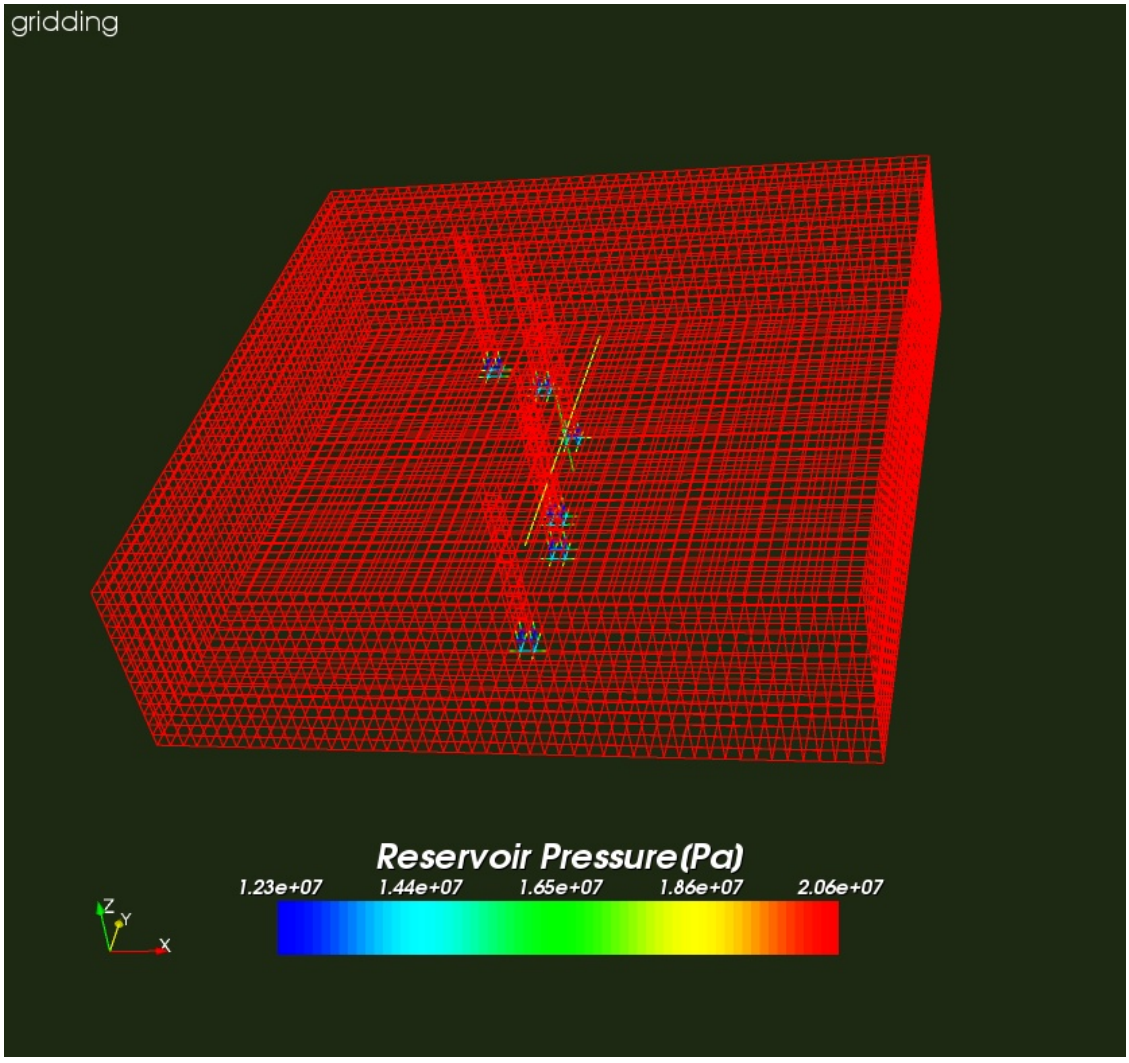


Figure 6.1 3D-Reservoir geometry

6.2. Case *Reservoir3D* for low compressible flow with closed boundaries

This scenario investigate pressure drop through the reservoir when the initial reservoir pressure is 206.32E+05[Pa]. All boundaries are no flow boundaries

6.2.1 Results from OpenFOAM version 1.4 simulator and analysis

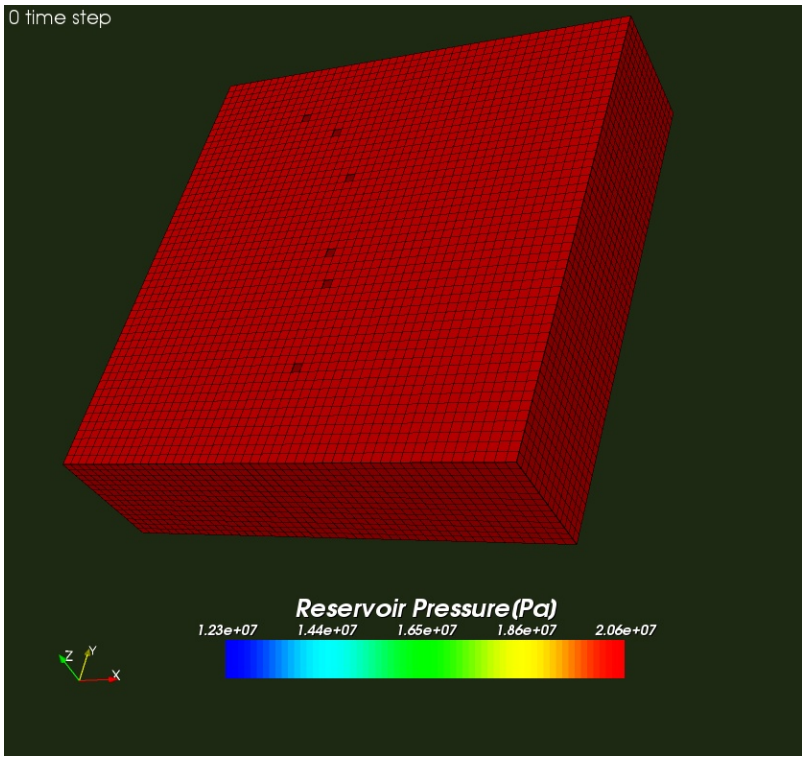


Figure 6.2: Initial pressure distribution

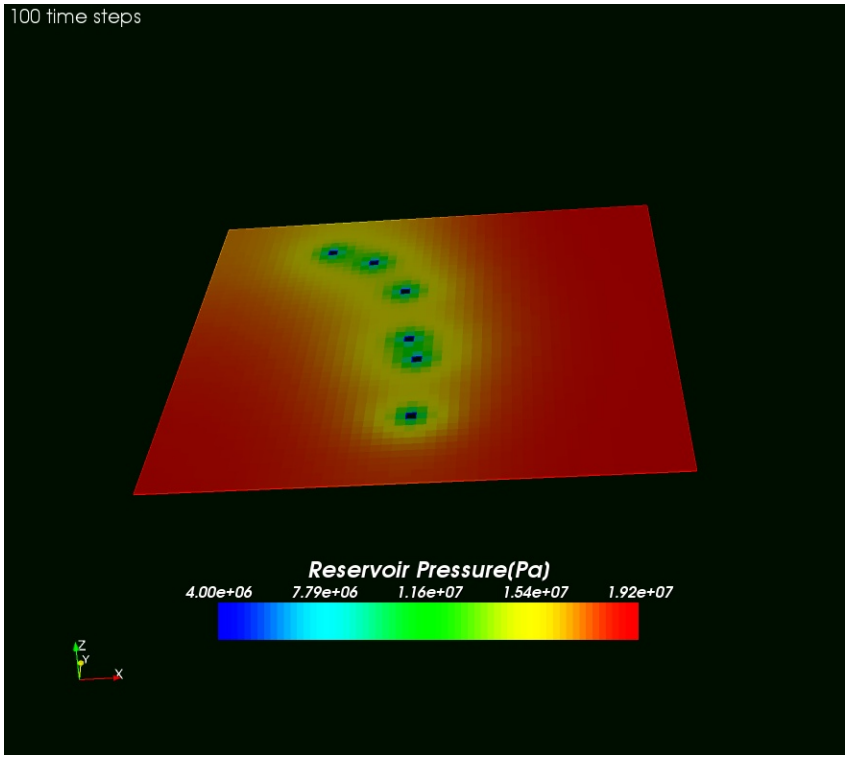


Figure 6.3: Pressure distribution after 100 time steps (2 days) (plane z=0.5)

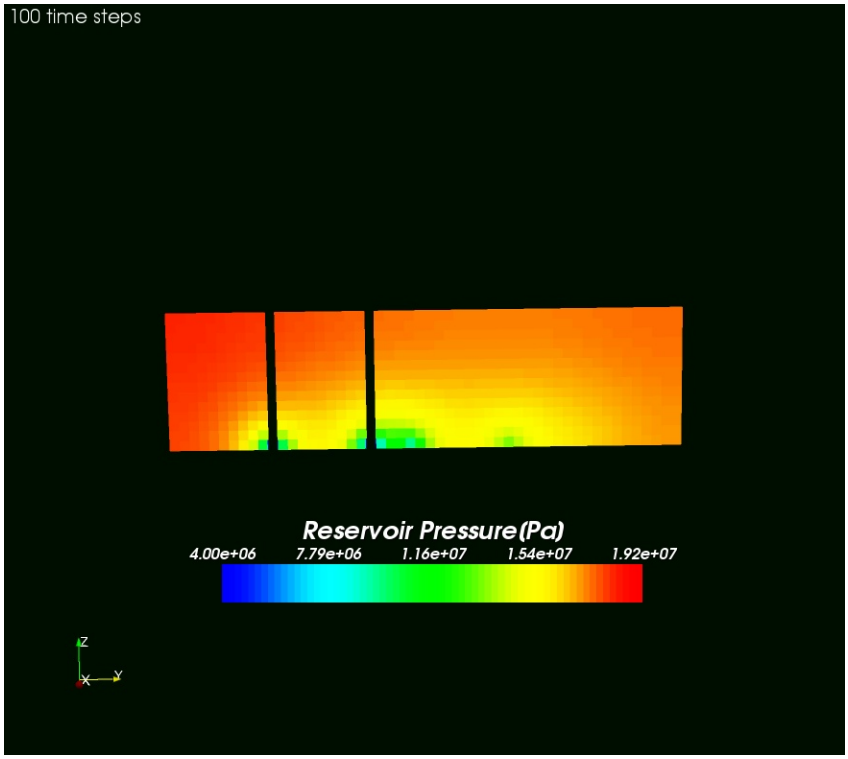


Figure 6.4: Pressure distribution after 100 time steps (2 days) (plane y-z)

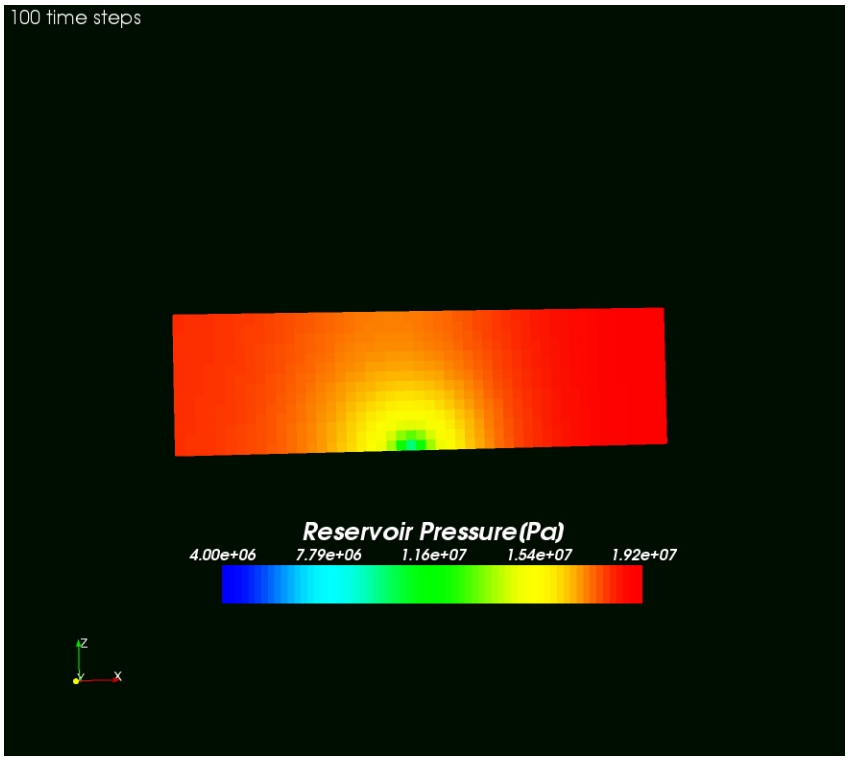


Figure 6.5: Pressure distribution after 100 time steps (2 days) (plane x-z)

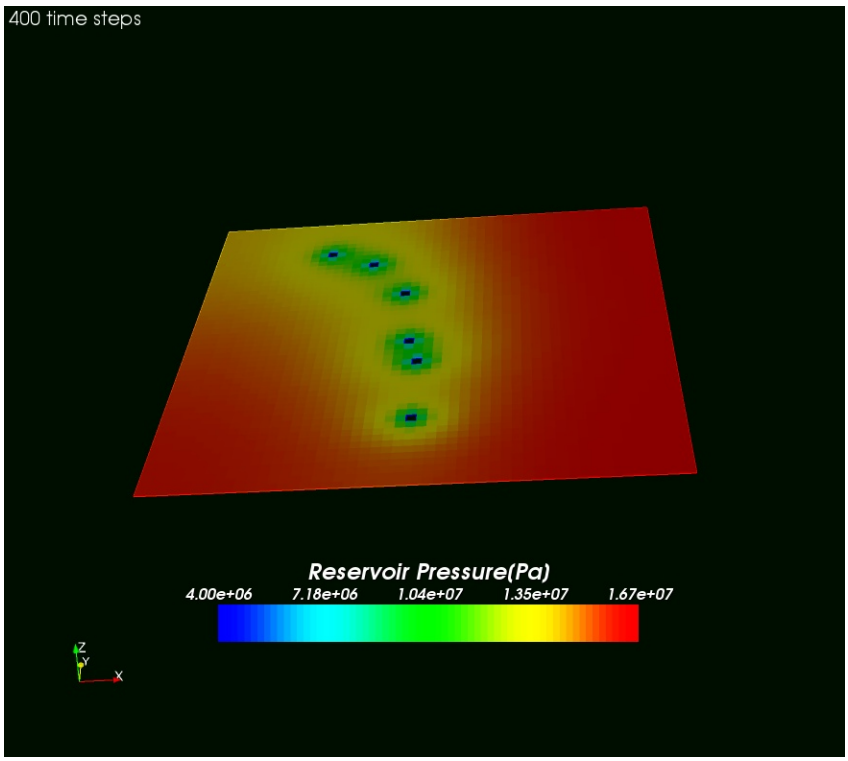


Figure 6.6: Pressure distribution after 400 time steps (8days) (plane z=0.5)

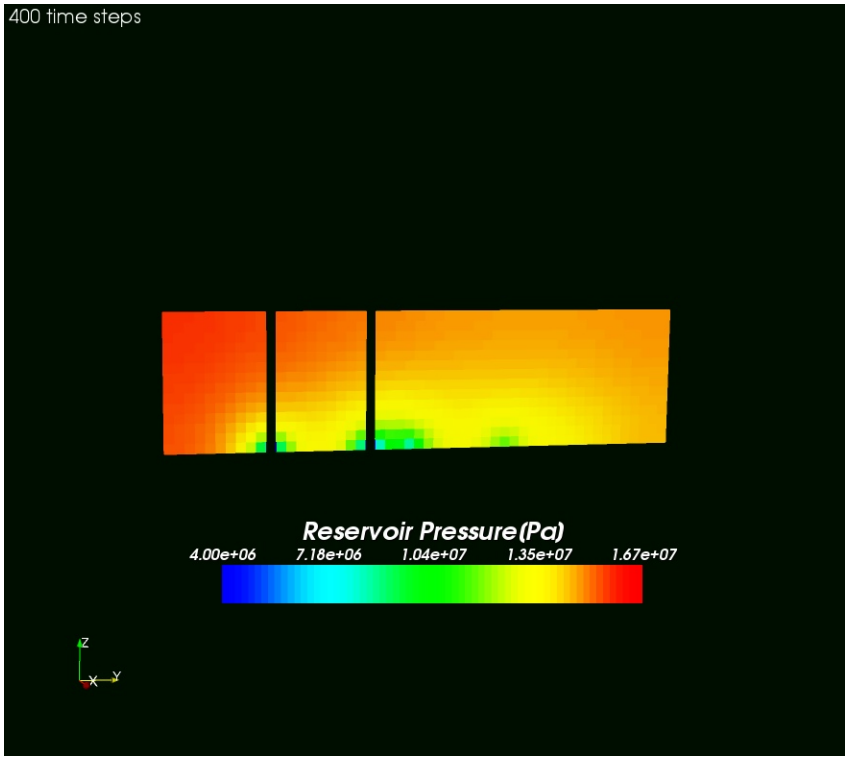


Figure 6.7: Pressure distribution after 400 time steps (8 days) (plane y-z)

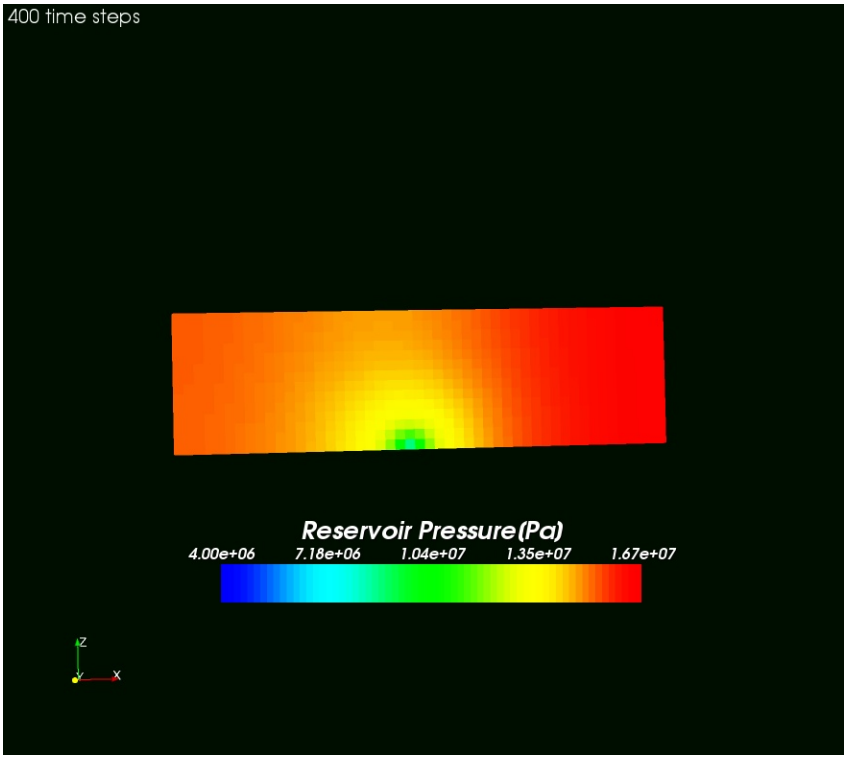


Figure 6.8: Pressure distribution after 400 time steps (8 days) (plane x-z)

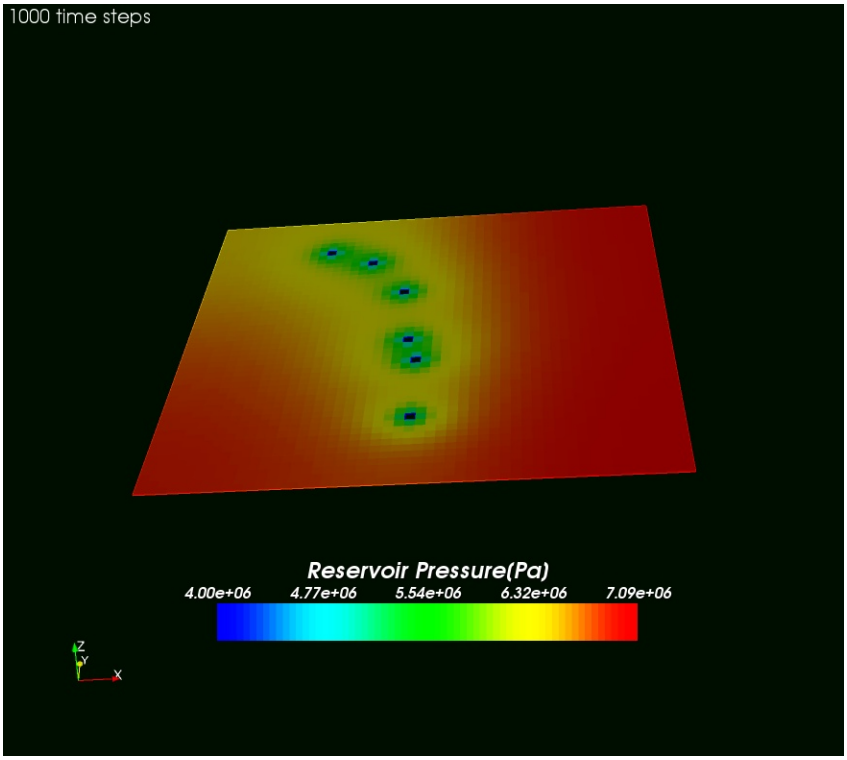


Figure 6.9: Pressure distribution after 1000 time steps (20 days) (plane z=0.5)

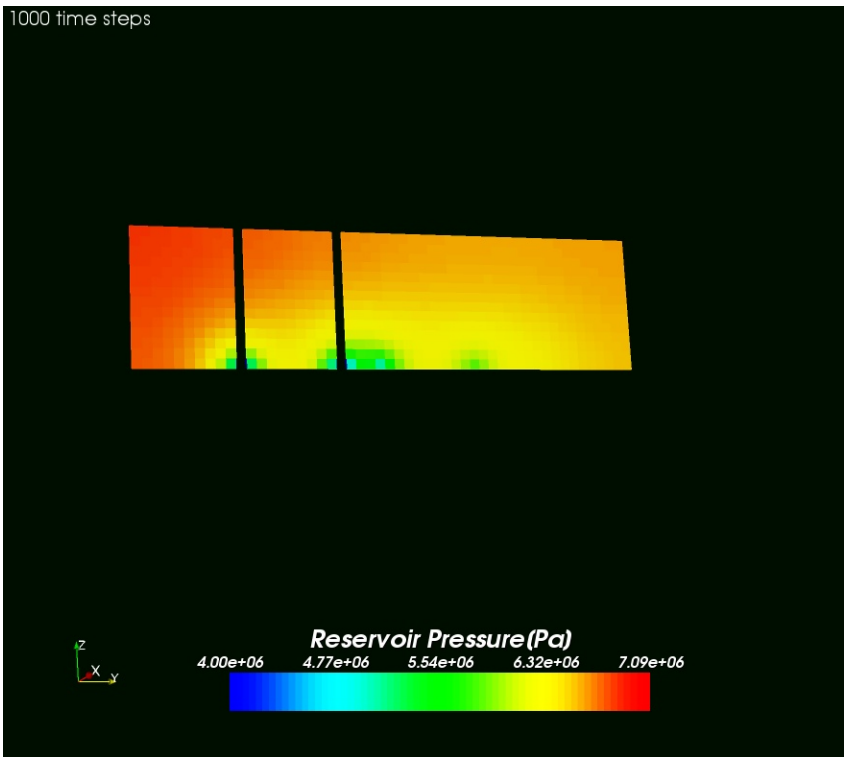


Figure 6.10: Pressure distribution after 1000 time steps (20 days) (plane y-z)

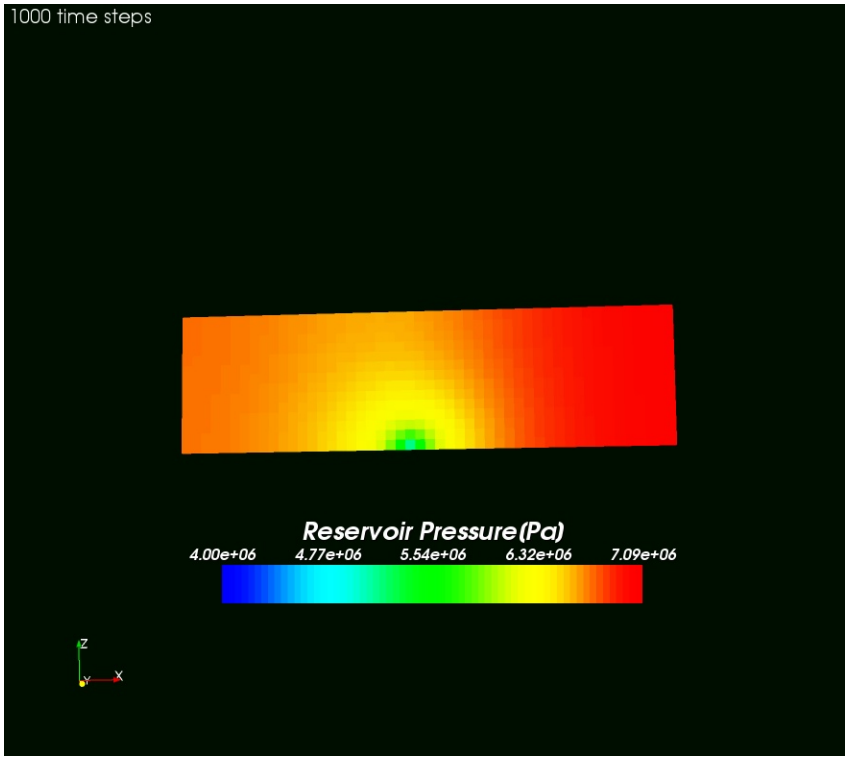


Figure 6.11: Pressure distribution after 1000 time steps (20 days) (plane x-z)

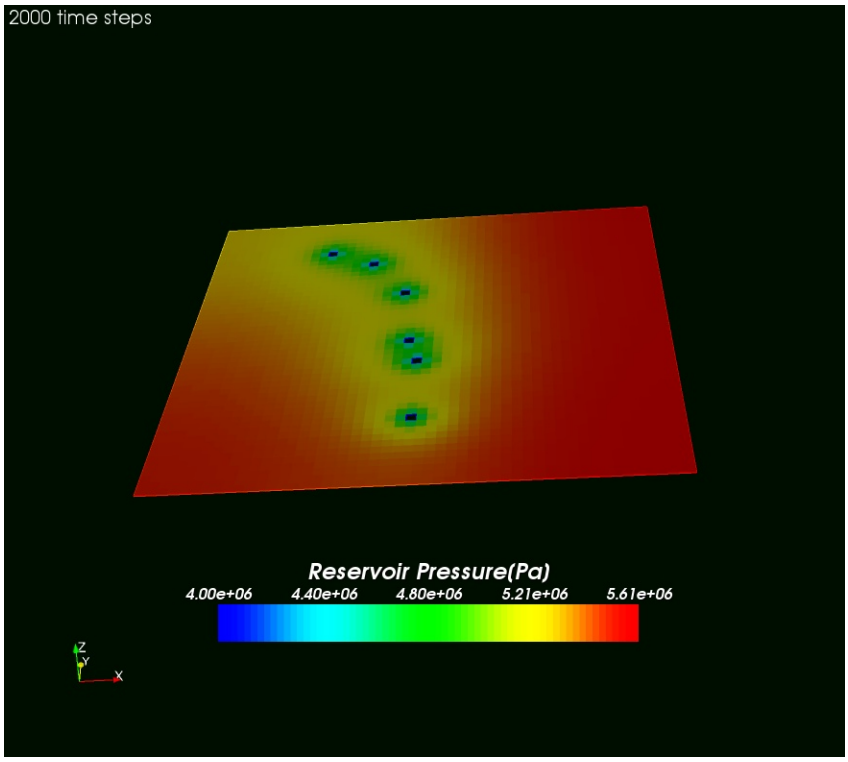


Figure 6.12: Pressure distribution after 2000 time steps (40 days) (plane z=0.5)

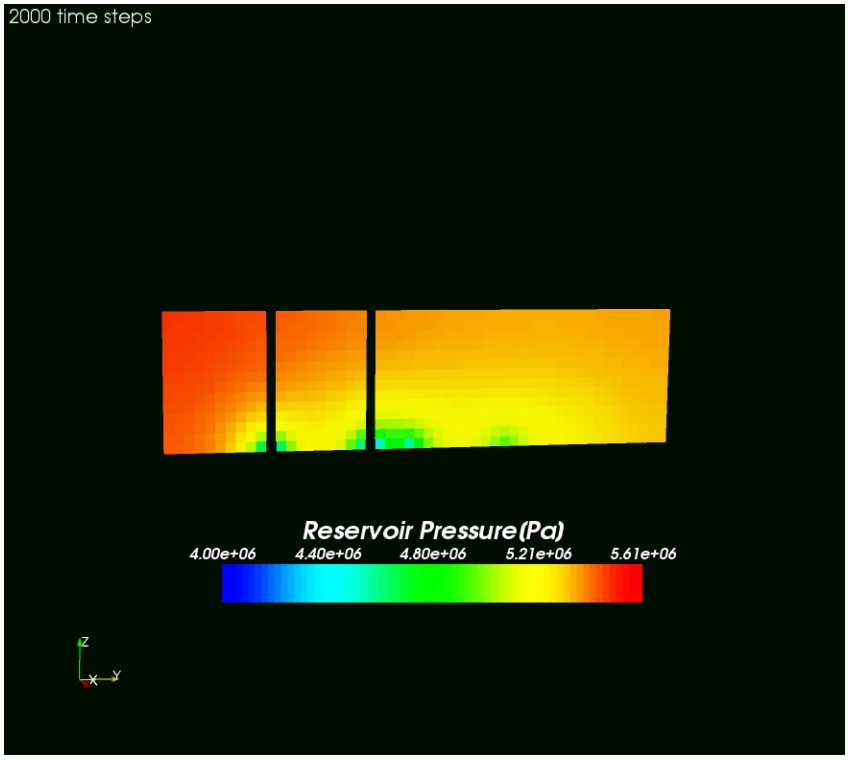


Figure 6.13: Pressure distribution after 2000 time steps (40 days) (plane y-z)

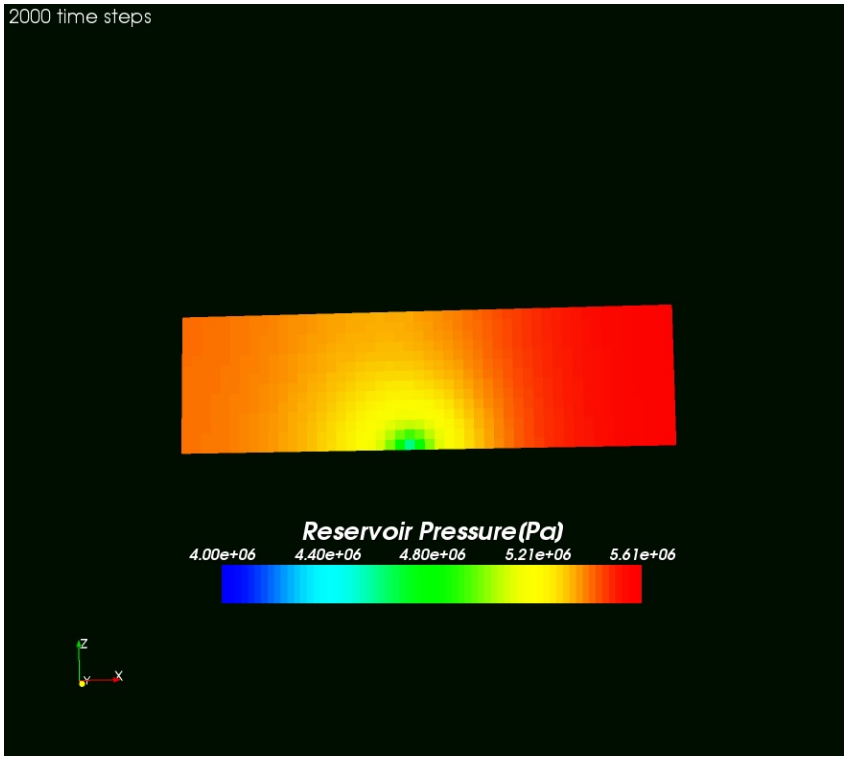


Figure 6.14: Pressure distribution after 2000 time steps (40 days) (plane x-z)

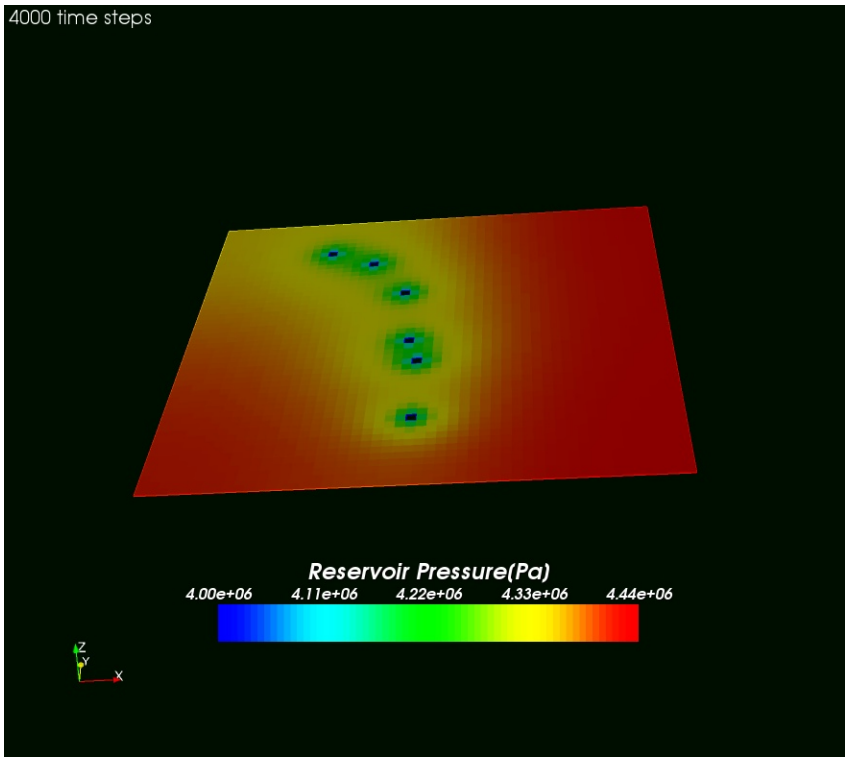


Figure 6.15: Pressure distribution after 4000 time steps (80 days) (plane z=0.5)

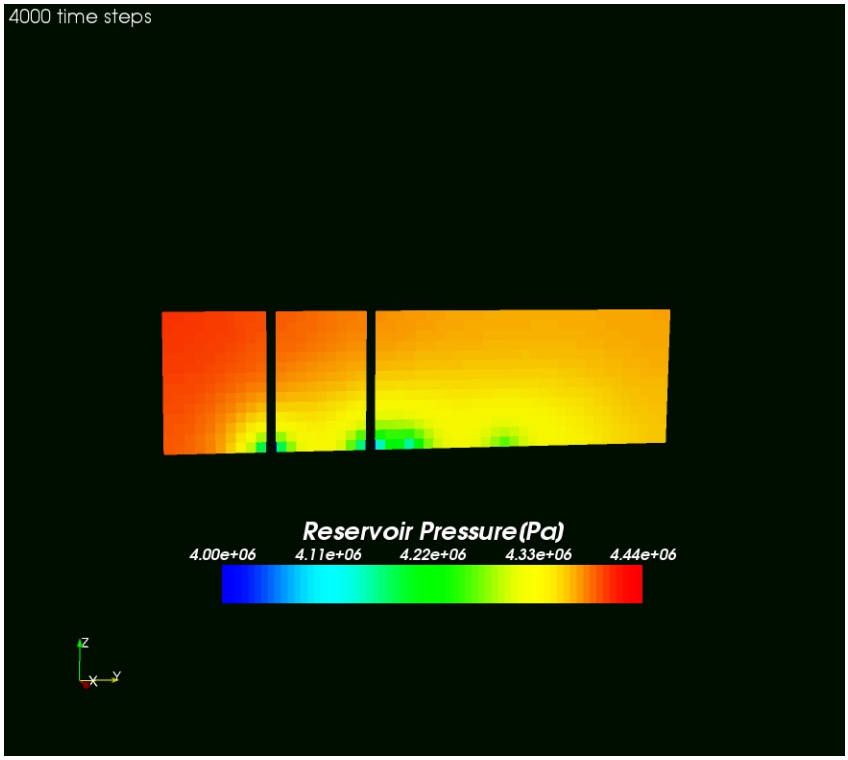


Figure 6.16: Pressure distribution after 4000 time steps (80 days) (plane y-z)

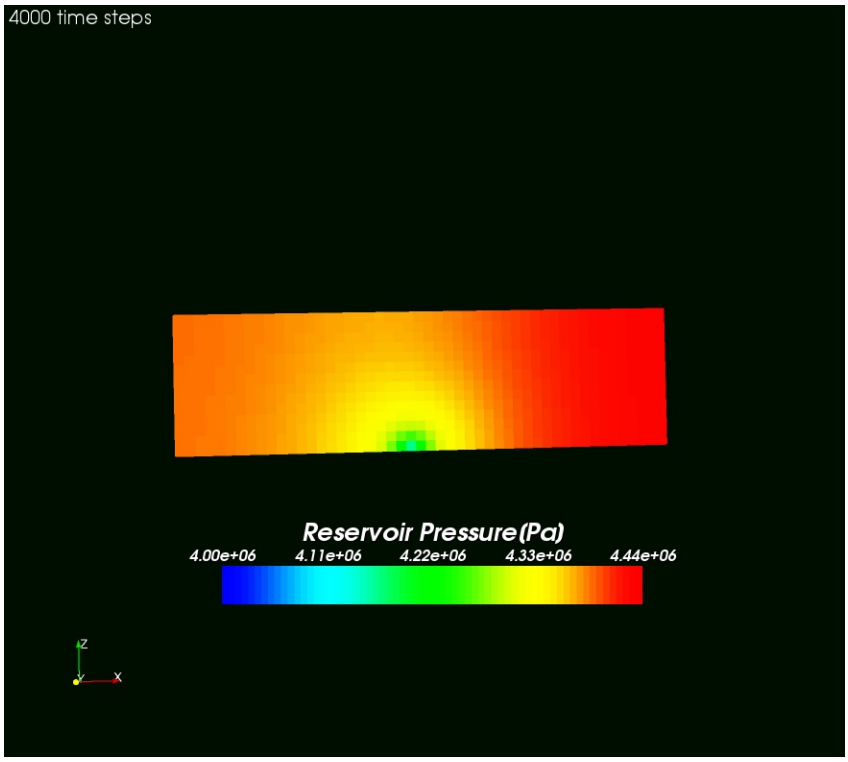


Figure 6.17: Pressure distribution after 4000 time steps (80 days) (plane x-z)

Performance plot

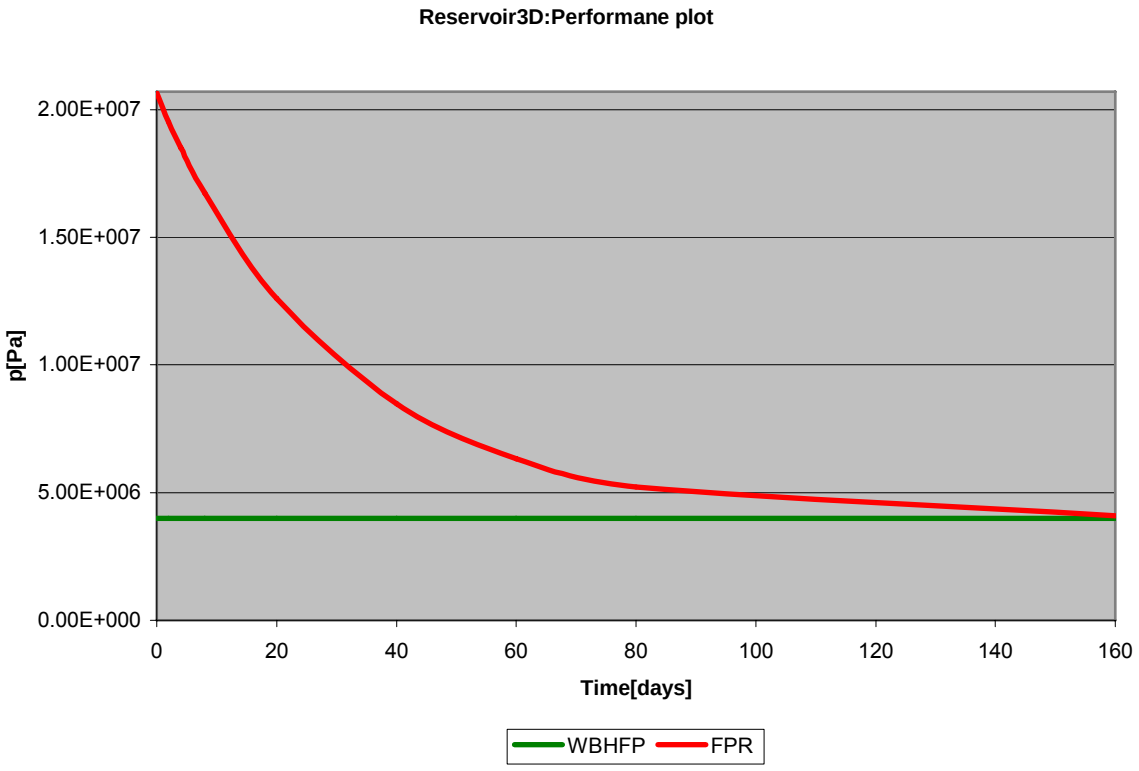


Figure 6.18:*Reservoir3D* performance plot (red-average field pressure, green- bottom hole flowing pressure) after 160 days of production

The results obtained show us that after 160 days of production we will reach more or less the bottom hole flowing pressure.

7. Conclusions, recommendations and future research

7.1 Conclusions

The open Source software OpenFOAM has the potential to simulate oil field reservoirs and can give a lot of information about the fluid flow behaviour inside porous media.

A comparison between two different Software packages (Eclipse and OpenFOAM) has been performed in simulating oil reservoirs on the basis of 2D and 3D models.

Three different 2D scenarios have been studied using OpenFOAM, namely low compressible flow with no flow boundary, low compressible flow with one flow boundary at constant pressure, and incompressible flow with one flow boundary at constant pressure.The pressure drop in the reservoir over production time was investigated .The final results were compared with the ones obtained from the reservoir simulation software Eclipse, and were in good agreement.

Furthermore, a 3D model with 6 production wells was set up. Again low compressible one phase flow was assumed. During the simulation no numerical instabilities were detected. However due to the simplified assumptions made for the 3D-model in OpenFOAM, no comparison with Eclipse was possible.There need to be done further research in order to be able to get comparable results.

The comparison of the results obtained with OpenFOAM and Eclipse , which without doubt is the most widely used tool in the area of hydrocarbon reservoir simulation, allows us to classify OpenFOAM as a potential software for reservoir simulation.

7.2 Recommendations and future research

One of the main advantages of OpenFOAM is the fact that it is an open source software and every one can download it and change the code to his specific problems. However, a weakness of the blockMesh utility is the fact that it is not flexible when building complex geometries. It will be important for complex geometries to use other pre-processing tools like Gambit, to generate the geometry and to export it to OpenFOAM format.

Hence it is very important to continue the development of the software by implementing OpenFOAM interface e.g. as Eclipse to apply OpenFOAM in the future to all kind of hydrocarbon reservoirs.

[1] **Anton, H.**, “Lectures notes of Reservoir Characterization and Management”, Leoben, Wintersemester 2006/2007.

[2] **Aziz, K., and Settari, A.**, “Petroleum Reservoir Simulation”, Elsevier Applied Sciences Publishers, London and Newyork, 1985.

[3] **Chick, V.,F.**, “Model Study for a Partly Miscible Gas Injection Project for a Heterogeneous Multi-Layer Reservoir”, Master Thesis, Leoben, 2003.

[4] **Eclipse Black Oil Reservoir Simulation**,Training and Exercise Guide, Schlumberger Information Solutions, Training 25-Aug-05.

[5] **Ferziger, J.,H., and Peric, M.,** “Computational Methods for Fluid Dynamics”, Second Edition, Springer.

[6] **Heinemann, Z.**, “Lectures Notes of Fluid Flow in Porous Media”, Volume1, Leoben January 2003.

[7] **Ganzer, L.**,”Lectures Notes of Reservoir Simulation”, Leoben, Wintersemester 2006/2007.

[8] **Lang, C.B., and Pucker,N.**, “ Mathematische Methoden in der Physik“, Spektrum Akademischer Verlag, Heidelberg Berlin, 1998.

[9] **Munka, M., and Pápay, J.**, “4D Numerical Modeling of Petroleum Reservoir Recovery”, Published by Akadémiai Kiadó, Budapest, 2001.

[10] **OpenFOAM**, User Guide, Version 1.4, March 2006.

[11] **OpenFOAM**, Programmer’s Guide, Version 1.0, December 2004.

[12] **Sauer, M.**, “Lectures Notes of Lab in Petroleum Production Engineering II”, Leoben 2004.

[13] **Tarek, A.**, “Reservoir Engineering Handbook“, Gulf Publishing Company, Houston Texas, 2000.

[14] **Schmaranz, K.**, “Software entwicklung in C++” Xpert-press. Springer-Verlag Berlin Heidelberg 2003,Germany.

[15] http://www.openfoam.org

[16] http://simple.wikipedia.org/wiki/Open-source

[17] http://mathworld.wolfram.com/PartialDifferentialEquation.html

APPENDIX A

Discretization of PDE terms in OpenFOAM.

Term description	Implicit/Explicit	Text expression	fvm::/fvc:: functions
Laplacian	Imp/Exp	$\nabla^2 p$ $\nabla . Dp \nabla p$	laplacian(p) laplacian(Dp , p)
Time derivative	Imp/Exp	$\frac{\partial p}{\partial t}$	ddt(p)
Divergence	Exp	$\nabla . p$	div(p)
Gradient	Exp	∇p	grad(p)

Dp is a scalar.

APPENDIX B

p[Pa]xE+05	B[rm^3/Sm^3]	Visc[Pa.s]xE-03
25	1	1.14
50	0.99	1.14
75	0.98	1.14
100	0.97	1.14
125	0.96	1.14
150	0.95	1.14
175	0.94	1.14
200	0.93	1.14

Table B-1: Formation volume factor and viscosity as a function of pressure

Time[Days]	WBHP[Pa]	Res2D1_FPR[Pa]	Res2D2_FPR[Pa]
0	5.00E+06	1.50E+07	1.50E+07
0.02	5.00E+06	1.48E+07	1.48E+07
0.04	5.00E+06	1.47E+07	1.47E+07
0.1	5.00E+06	1.43E+07	1.43E+07
0.2	5.00E+06	1.37E+07	1.39E+07
0.4	5.00E+06	1.27E+07	1.35E+07
0.6	5.00E+06	1.18E+07	1.34E+07
1	5.00E+06	1.03E+07	1.33E+07
1.5	5.00E+06	8.85E+06	1.32E+07
2	5.00E+06	7.80E+06	1.32E+07

Table B-2: Post-processing data from Eclipse for 2D

APPENDIX C

Some post processing data from OpenFOAM

Case *Reservoir2D1*

Time[Days]	WBHFP[Pa]	Res2D1_FPR[Pa]
0	5.00E+06	1.50E+07
0.02	5.00E+06	1.48E+07
0.04	5.00E+06	1.47E+07
0.1	5.00E+06	1.43E+07
0.2	5.00E+06	1.38E+07
0.4	5.00E+06	1.28E+07
0.6	5.00E+06	1.19E+07
1	5.00E+06	1.05E+07
1.5	5.00E+06	9.09E+06
2	5.00E+06	8.05E+06

TableC-1: WBHFP, FPR data of the case *Reservoir2D1.*

Case *Reservoir2D2*

Time[Days]	WBHFP[Pa]	Res2D2_FPR[Pa]
0	5.00E+06	1.50E+07
0.02	5.00E+06	1.48E+07
0.04	5.00E+06	1.47E+07
0.1	5.00E+06	1.43E+07
0.2	5.00E+06	1.40E+07
0.4	5.00E+06	1.35E+07
0.6	5.00E+06	1.34E+07
1	5.00E+06	1.33E+07
1.5	5.00E+06	1.33E+07
2	5.00E+06	1.33E+07

TableC-2: WBHFP, FPR data of the case *Reservoir2D2*

Case *Reservoir2D2Incompressible*

Time[Days]	WBHFP[Pa]	Res2D2Inc._FPR[Pa]
0	5.00E+06	1.50E+07
0.02	5.00E+06	1.44E+07
0.04	5.00E+06	1.40E+07
0.1	5.00E+06	1.35E+07
0.2	5.00E+06	1.33E+07
0.4	5.00E+06	1.33E+07
0.6	5.00E+06	1.33E+07
1	5.00E+06	1.33E+07
1.5	5.00E+06	1.33E+07
2	5.00E+06	1.33E+07

TableC-3: WBHFP, FPR data the case *Reservoir2D2Inc*

Data Comparison:

Eclipse	OpenFOAM			
Reservoir2D1	Reservoir2D1			
Res2D1_FPR[Pa]	Res2D1_FPR[Pa]	Eclipse/OpenF.	Res2D1_FPR[Pa]	Eclipse/OpenF.
1.50E+07	1.50E+07	1.00	1.50E+07	1.00
1.48E+07	1.48E+07	1.00	1.48E+07	1.00
1.47E+07	1.47E+07	1.00	1.47E+07	1.00
1.43E+07	1.44E+07	0.99	1.43E+07	1.00
1.37E+07	1.40E+07	0.98	1.38E+07	0.99
1.27E+07	1.33E+07	0.95	1.28E+07	0.99
1.18E+07	1.26E+07	0.94	1.19E+07	0.99
1.03E+07	1.14E+07	0.90	1.05E+07	0.98
8.85E+06	1.02E+07	0.87	9.09E+06	0.97
7.80E+06	9.20E+06	0.85	8.05E+06	0.97

Table C-4: Pressure-data comparison between Eclipse and OpenFOAM for *Reservoir2D1*

Eclipse	OpenFOAM			
Reservoir2D2	Reservoir2D2			
Res2D2_FPR[Pa]	Res2D2_FPR[Pa]	Eclipse/OpenF.	Res2D2_FPR[Pa]	Eclipse/OpenF.
1.50E+07	1.50E+07	1.00	1.50E+07	1.00
1.48E+07	1.48E+07	1.00	1.48E+07	1.00
1.47E+07	1.47E+07	1.00	1.47E+07	1.00
1.43E+07	1.45E+07	0.99	1.43E+07	1.00
1.39E+07	1.42E+07	0.98	1.40E+07	0.99
1.35E+07	1.38E+07	0.98	1.35E+07	1.00
1.34E+07	1.37E+07	0.97	1.34E+07	1.00
1.33E+07	1.36E+07	0.97	1.33E+07	1.00
1.32E+07	1.36E+07	0.97	1.33E+07	1.00
1.32E+07	1.36E+07	0.97	1.33E+07	1.00

Table C-5: Pressure-data comparison between Eclipse and OpenFOAM for *Reservoir2D2*

Case *Reservoir3D*

Time[days]	WBHFP[Pa]	FPR[Pa]
0	4.00E+006	2.07E+007
2	4.00E+006	1.95E+007
4	4.00E+006	1.85E+007
8	4.00E+006	1.67E+007
20	4.00E+006	1.26E+007
40	4.00E+006	8.48E+006
60	4.00E+006	6.33E+006
80	4.00E+006	5.22E+006
160	4.00E+006	4.09E+006

TableC-6: WBHFP, FPR data of the case *Reservoir3D*

APPENDIX D

Dimensional units

In continuum mechanics, properties are represented in some chosen units, e.g. mass in kilograms [kg], volume in cubic metres [m^3], pressure in Pascals [$kg.m^{-1}s^{-2}$]. Algebraic operations must be performed on these properties using consistent units of measurement; in particular, addition, subtraction and equality are only physically meaningful for properties of the same dimensional units. As a safeguard against implementing a meaningless operation, OpenFOAM encourages the user to attach dimensional units to any tensor and will then perform dimension checking of any tensor operation.

Units are defined using the *dimensionSet* class, e.g.

dimensionSet pressureDims(1, -1, -2, 0, 0, 0, 0);

No.	Property	Unit	Symbol
1	Mass	kilogram	kg
2	Length	metre	m
3	Time	second	s
4	Temperature	Kelvin	K
5	Quantity	moles	mol
6	Current	ampere	A
7	Luminous	intensity candela	cd

TableD-1: S.I. base units of measurement

Where each of the values corresponds to the power of each of the S.I. base units of measurement listed in table above. The line of code declares *pressureDims* to be the *dimensionSet* for pressure [$kg.m^{-1}s^{-2}$] since the first entry in the *pressureDims* array, 1, corresponds to kg, the second entry, -1, corresponds to m^{-1} etc... A tensor with

units is defined using the dimensioned<Type> template class, the <Type> being scalar, vector, tensor, etc... [10]

APPENDIX E

Some important standard solvers

*Basic CFD solvers

filterFoam: Transient solver for compressible flow in a porous medium.

lapacianFoam: Solves a simple laplacian equation.

porousIcoFoam: Transient solver for incompressible, laminar flow of Newtonian fluids with porous regions.

porousSimpleFoam: Steady-state solver for incompressible, turbulent flow of non-Newtonian fluids.

steadyFilterFoam: Steady –state solver for compressible flow in a porous medium.

*Some important standard utilities

FoamX: *GUI*

Mesh generation

blockMesh: Mesh generator: blockOffsets_(createBlockOffsets()), mergeList_(createMergeList()), points_(createPoints()), cells_(createCells()), patches_(createPatches()).

extrudeMesh: Extrude mesh from existing patch or from patch read from file.

Mesh manipulation

refineMesh: Utility to refine cells in multiple directions. Either supply -all option to refine all cells (3D refinement for 3D cases; 2D for 2D cases) or reads a refineMeshDict with - cellSet to refine - directions to refine.

cellSet: Selects a cell set through a dictionary.

checkMesh: Checks validity of a mesh.

refineMesh: Utility to refine cells in multiple directions. Either supply -all option to refine all cells (3D refinement for 3D cases; 2D for 2D cases) or reads a refineMeshDict with - cellSet to refine - directions to refine.[10]

APPENDIX F (CD)

Solver: *laplacianFoamSimulator*

Cases:

***Reservoir2D1*: Low compressible flow with no flow boundaries**

***Reservoir2D2*: Low compressible flow with one flow boundary at constant pressure**

***Reservoir2D2Incompressible*: Incompressible flow with one flow boundary at constant pressure**

***Reservoir3D1*: Low compressible flow with closed boundary**