Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

Fadhil Berylian

Calculation and Visualization of Energy Dissipation and Energy Balance in Reservoir Models

Master's thesis in Petroleum Engineering
Supervisor: Carl Fredrik Berg
June 2020

NTNU
Norwegian University of
Science and Technology

Fadhil Berylian

Calculation and Visualization of Energy Dissipation and Energy Balance in Reservoir Models

Master's thesis in Petroleum Engineering
Supervisor: Carl Fredrik Berg
June 2020

Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

NTNU
Norwegian University of
Science and Technology

”Nothing is lost, nothing is created, everything is transformed.”
- Antoine Laurent de Lavoisier

Summary

During the production lifetime of a hydrocarbon reservoir, various forms of energy are involved: external energy through wells, internal energy stored in the reservoir itself, and dissipated energy as heat released to the surrounding environment. Calculating and visualizing these energy forms could present new insights regarding recovery processes of a reservoir. The calculations are tested by an energy balance equation that makes sure that the results adhere to the basic laws of thermodynamics. This study aims to develop calculations for each component in the energy balance, test the calculations using various synthetic cases to ensure their robustness, and finally apply them to a full-field reservoir simulation model.

The test cases are built and simulated using OPM Flow and Schlumberger ECLIPSE. Both are reservoir simulators that use a similar format and allow the same set of input files to be used interchangeably. OPM ResInsight is used for model visualization, and in addition to that, the software's capability to connect with Python allows it to be used for energy calculations. First, a simplistic, one-dimensional model is used to test the energy calculations. Then, both the test model and the calculations are gradually enhanced, introducing more complexities to resemble an actual reservoir better. In the end, the energy calculation is tested against the Norne reservoir model to prove whether it is practical for real-world applications.

After testing the calculations with simple test cases, it is observed that the developed energy calculations provide valid results and achieve the expected energy balance. The calculations have also demonstrated itself of being able to handle additions of non-neighbor connections (NNC), faults, and inactive simulation cells in a reservoir model. Finally, the calculations are applied to Norne, and the results showed a successful energy balance calculation for the real reservoir model. Energy changes in the Norne reservoir are visualized, and interesting insights are found on how the different layers and formations in Norne communicate.

In conclusion, this study has identified the different energy forms present during reservoir recovery and has developed an energy calculation that adheres to energy balance within the reservoir system. The robustness and validity of the calculations have been tested against both simple reservoir cases and a full-field reservoir model. Potential applications of energy calculations and visualizations are also discussed that could help expand upon the subject. Examples of said applications include energy efficiency calculations between well pairs, optimization of well placement and well control based on energy calculations in wells, and implementation of the developed calculations in reservoir engineering software.

Preface

I am grateful to the Almighty God, Allah SWT, for it is only with His guidance that I am able to finish this Master's thesis and complete my study at NTNU.

Thank you from all my heart to my parents, Bambang Irawan and Tiktik Kartika, and my brothers, Faza Fahleraz and Fathan Naufal Trias, for the love and support throughout the last two years. Being far from home and all of you have taught me to always be grateful, and to never take our time together as family for granted.

For my supervisor, Carl Fredrik Berg, I thank you for all the time, supports, and advice. You always encouraged me that I am able to do what I thought I could not do, and in the end it turns out that I could. Special thanks also for I Gusti Agung Gede a.k.a Angga for all the tips and tricks that helped me get satisfactory results in this thesis.

To Nadhira Khairunnisa, the special one: They say distance tears relationships down, yet here we are still standing strong. Thank you for never stopped believing in me, and for brightening up even my darkest days. Thank you for all the laughs and love we shared despite being continents apart. Really looking forward to spend my future with you.

Thank you to the Indonesian Student Association (PPI) in Trondheim for all the precious time since my first day coming here, and to other Indonesian families and communities in Trondheim for always making this place feel like home to me. I will always remember all the fun events, awesome gatherings, delicious foods, and valuable lessons.

Thanks to the dwellers of Moholt Alle 16, 3rd floor: Muhammad Gibran Alfarizi, Mikael Yuan Estuariwinarno, and Harbi Qodri. You have kept me sane throughout my last two semesters at NTNU. Probably I could finish my thesis faster if not for you guys, but I have most likely gone insane. To Salma Alkindira, thanks for the motivations and lessons to improve my cooking skills. To Rinaldi Anwar, thanks for taking nice photographs that I could add to my gallery.

Special shoutout to the group "Gamer Indo Trondheim" for reigniting my love of video games. When I first arrived here, I only play games on my laptop. Now, I have PlayStation, XBOX, and Nintendo consoles. Remember to never stop playing, as a wise man once said: "You don't stop playing because you get old, you get old because you stop playing."

Finally, all my thanks to everyone that I have worked with during my two years in Norway. Each of you have made an impact to me so I could be better. I really appreciate this chapter of my life as a student in NTNU. As a token of gratitude, I present this Master's thesis in hope that it could be any help for those that need it.

Table of Contents

List of Tables

List of Figures

Abbreviations

B	=	formation volume factor (FVF)
$\mathbf{c}$	=	specific heat
c	=	compressibility
E	=	energy
$\hat{e}^p$	=	mass transfer rate of phase p
F	=	Helmholtz free energy
g (subscript)	=	gas phase
i (subscript)	=	grid block index
k	=	permeability
k_r	=	relative permeability
o (subscript)	=	oil phase
P_c	=	capillary pressure
p	=	pressure
Q	=	heat
q	=	fluid flow rate
$\vec{\phi}_q$	=	heat flux
R_s	=	solution gas oil ratio
S	=	entropy
s	=	fluid saturation
T	=	transmissibility
$\hat{T}^p$	=	momentum transfer rate of phase p
t_p	=	stress tensor of phase p
U	=	internal energy
V	=	volume
v	=	specific volume
v_p	=	Darcy velocity of phase p
W	=	work
w (subscript)	=	water phase
Z	=	distance in vertical direction
z	=	gas deviation factor
γ	=	fluid specific gravity
Λ	=	rate of net entropy increase
μ	=	fluid viscosity
ρ	=	density
Φ	=	fluid potential
Φ^p	=	entropy transfer rate of phase p
ϕ	=	porosity
$\vec{\varphi}$	=	entropy flux

Chapter 1

Introduction

This chapter will discuss some background to this study and formulation of problems this study aims to answer, ultimately resulting in the objectives of the thesis.

1.1 Problem Description

A hydrocarbon reservoir undergoes a number of different recovery mechanisms during its lifetime, which would affect its performance on production. In each mechanism, various forms of energy are involved. These energy forms include (Muskat, 1981): energy of compression of different fluid phases within the reservoir (oil, free gas, water); energy within dissolved gas in the oil phase; energy of a body of compressed water surrounding and in connection with the reservoir (also known as an aquifer). All of the energy forms described above play their part during the first stage of a reservoir recovery, and the process is called "primary drive mechanisms." After production wells are opened for the first time, the difference in pressure between compressed reservoir fluids and the production wells results in energy being spent on expanding and flowing the fluids into the wells. At the same time, energy from dissolved gas is expended as it expands within the oil and eventually liberates from solution. Then, the two phases flow separately toward the wells. As time goes on, the internal reservoir energy available to drive production decreases due to lower reservoir-well pressure difference. In some cases, however, the reservoir pressure is maintained by the help of surrounding aquifers. Reservoirs in connection to large aquifers could keep production stable for a much longer time compared to those that are not surrounded by aquifers. Figure 1.1 illustrates the whole process of the primary stage of reservoir recovery. In the figure, dV_o is volume of oil resulted from expansion of the compressed oil, dV_w is water from aquifers that encroached into the reservoir, and dV_g is expanded free gas. All of these volumes push an amount of dV_{tot} to be produced by the well.

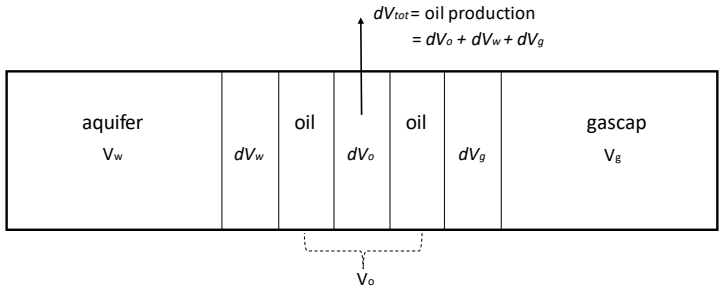


Figure 1.1: Primary reservoir recovery. Modified from Dake (1978).

For a reservoir that depends primarily on energy from fluid expansion, production will continually slow down until its rate is deemed to be too small to be economical. At this point, there is usually still a large portion of hydrocarbons in the reservoir yet to be produced, but they lack energy to push themselves into the wells and up to surface. External sources of energy are then introduced to the reservoir, most commonly in the form of injection wells flowing water or gas into the formation. The fluids are injected with high pressure to push the remaining hydrocarbons to flow into the production wells. This addition of external energy sources is the second phase of reservoir recovery, dubbed ”secondary drive mechanism” or ”secondary recovery.” To reach high efficiency during secondary recovery, the placement of production and injection wells in the reservoir becomes important. An optimal arrangement of wells could flood more areas with injected fluid, resulting in higher production. One example of a common well arrangement used widely is the ”five-spot pattern,” pictured in Figure 1.2. The black dots represent production wells, while the white dots represent injection wells. Each production well is associated with four injection wells around it, and each injection well is also associated with four production wells.

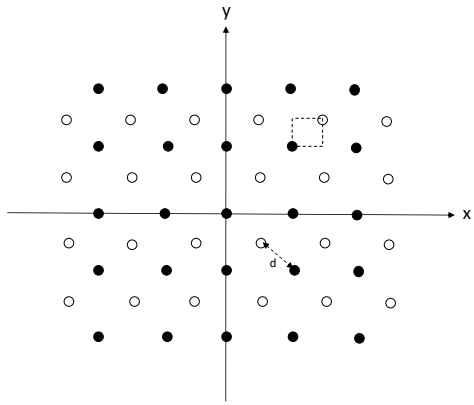


Figure 1.2: Five-spot pattern used in secondary reservoir recovery. The distance between a production and an injection well is d, and a volume element associated with the wells is represented by the dashed segment. Modified from Green and Wilhite (2018).

In the final stage of reservoir recovery, known as "tertiary recovery," other methods are used to extract more hydrocarbons from the reservoir when even the secondary recovery becomes uneconomical. Nowadays, tertiary recovery is more commonly known by the term "enhanced oil recovery" or "EOR," because the methods used are not simply adding energy to the reservoir, but rather enhances or improves the properties of the reservoir so that more hydrocarbon could be produced. Examples of EOR processes include polymer injection, surfactant injection, miscible CO2 injection, and steam injection. Practical considerations have to be made to determine which EOR process to be used in a specific reservoir, specifically for technical and economic aspects (Green and Wilhite, 2018). Technical aspects to be considered include properties of the reservoir that describe the nature of flow in porous media, rock/fluid interactions, and chemical compatibility between injectants and reservoir fluid. Economic aspects to be considered include the availability of injectants and other logistical affairs.

As stated before, various energy forms exist throughout the different stages of reservoir recovery, especially the primary and secondary recovery. Flowing fluids from the pore space of a reservoir to an opened production well requires energy, and that energy will be converted to heat and dissipated during the process. The reservoir could be considered as a system that itself has an amount of energy stored internally, and it could give off or receive energy. Dissipation of energy (E_{dis}) as heat means that the system has given off some energy, and as a result, the system loses some of its stored internal energy (E_{int}). On another side, fluids are brought out of the reservoir system by production wells. This means the reservoir has lost an amount of external energy (E_{ext}), named after the fact that this energy is related to an external system (in this case, the surface). If there is an external source available connected to the system, such as injected water or gas, external energy is gained by the system to be stored and replace the lost energy due to dissipation and production. Using this simplified narrative, an instance of *energy balance* can then be defined for the reservoir, in which the total changes in internal energy equals the total dissipated energy subtracted by the total added energy by an external source. The basic equation for energy balance in a reservoir is defined by Equation 1.1, and a simple illustration is depicted in Figure 1.3.

$$dE_{int} = dE_{ext} - dE_{dis} \tag{1.1}$$

Visualizing the components of this energy balance could provide valuable insights into the inner workings of a reservoir. For example, one could observe areas of the field in which energy is dissipated or stored the most. The main driving force in the reservoir system is the energy used to flow the fluids and is dissipated afterward. Calculating this energy dissipation is an important step in seeing energy balance in a reservoir system.

On the practical side, calculating energy dissipation could lead to some beneficial applications. In a complex field like Norne, where there are multiple production and injection wells, placement and control of those wells need to be taken into consideration. A combination of optimal well placement and well control parameters is key to optimal production.

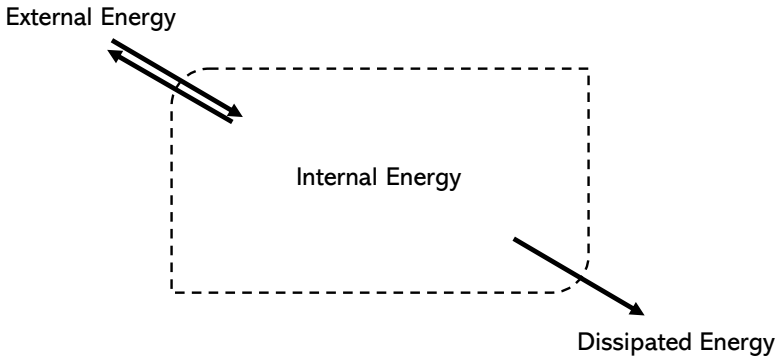


Figure 1.3: Energy balance within a reservoir system.

However, the energy used to drive this production is often overlooked. If one has information on how much energy is spent to flow between a producer to an injector over time, then the efficiency of that producer-injector well pair could be interpreted. Calculating the energy dissipation could also reveal parts of the reservoir in which more heat is released, and then that specific area could be analyzed for anything unusual. This study tries to explore several applications that could prove useful in developing production scenarios of a field. Future works on this topic could expand the subject and open doors to opportunities not explored and thought before.

1.2 Objective

Based on the background and problem description, the objectives of this study can be formulated as:

1. Identify components required to calculate energy dissipation and energy balance, and develop the calculations to be used on reservoir models.

2. Apply the developed calculations for a full-field, real reservoir simulation model.

1.3 Outline

This thesis is segmented into five different chapters, each covering important elements required to develop this study.

1. **Introduction**. This chapter defines problem description as the background of conducting this study, and research objectives this study intents to achieve.

2. **Background and Basic Theory**. The second chapter covers thermodynamics of flow in reservoir and reservoir simulation, as fundamental theories relevant to this thesis. They are described here as a result of extensive literature reviews. Software

used in this study, OPM Flow and OPM ResInsight, are also described in this chapter. Section 2.2 regarding reservoir simulation and Subsection 2.3.1 regarding OPM Flow have been previously written in the author's report for TPG4560 Specialization Project in Fall 2019.

3. **Methodology**. Procedures to develop and implement this study is presented in this chapter. First, energy calculations are developed in Python and connecting it to ResInsight. Then, various cases are built to test the calculations. Finally, a full-field reservoir model, Norne, is defined, and the developed calculations are applied to determine energy changes and dissipation.

4. **Results and Discussion**. Results from test cases and Norne are presented. From the results, analysis and discussion are then conducted to gain valuable insights and findings from this study.

5. **Conclusions and Recommendations**. This chapter provides conclusions of this thesis, fulfilling all objectives described in the first chapter. In addition, recommendations for future works are given to provide suggestions for further research on the subject.

Chapter 2

Background and Basic Theory

This chapter presents underlying theories and principles used in this study, including literature review and software, and how they are relevant to and used in this study.

2.1 Thermodynamics of Flow in Reservoir

This section will cover the fundamental concepts for different forms of energy involved during fluid flow, energy dissipation and the energy balance calculation, starting from general equations that govern thermodynamics properties of fluids, to their applications in multiphase flow, and finally wrapping up with definition of energy forms in reservoir and how to compute them based on the discussed concepts.

2.1.1 General thermodynamics equations for multiphase flow

The discussion regarding energy in multiphase flow begins at classical thermodynamics, which concerns equilibrium states of uniform matter, such as fluid (Batchelor and Batchelor, 2000). The state of a given fluid in equilibrium could be defined at the most simple form by only two parameters: specific volume v, which is volume (V) per unit mass and equals the inverse of density ($v = \frac{1}{\rho}$) and pressure p. Every other parameters that could define the state of the fluid is in fact a function of these two parameters, including temperature T. As an example, a fluid in equilibrium with specific volume v_1 and pressure p_1 can only have one possible temperature, let us call it T_1. Meanwhile, a fluid in equilibrium that has a temperature of T_1 could have a lot of possible states depending on both its specific volume and pressure.

One important parameter that describes a fluid state is the internal energy of the fluid, U. By the first law of thermodynamics that defines conservation of energy, if the state of the fluid is changed by subjecting it to an amount of work W and giving it an amount of

heat Q, then the internal energy is changed by:

$$\Delta U = Q + W \tag{2.1}$$

The sign for Q and W depends on the direction of energy relative to the system. If heat enters the system then Q is positive, but if it escapes the system then Q is negative. If work is done to the system then W is positive, but if the system is put to work then W is negative.

The most common way in fluid mechanics of subjecting a fluid mass to work is by compressing or expanding it. The work done on the fluid by compressing the volume by δV is $p\delta V$. By convention, compressing the fluid *increases* its internal energy, so attributing work to Equation 2.1 we get:

$$\delta U = \delta Q - p\delta V \tag{2.2}$$

The amount of heat δQ required during a compression of a fluid at rest is given by:

$$\begin{aligned} \delta Q &= \delta U + p\delta V \\ &= \Big(\frac{\partial U}{\partial p}\Big)_V \delta p + \Big(\frac{\partial U}{\partial V}\Big)_p \delta V + p\delta V \end{aligned} \tag{2.3}$$

Adding heat δQ to the fluid will increase its temperature. The amount of δT increase is governed by the fluid's specific heat c, which is defined as:

$$\mathbf{c} = \frac{\delta Q}{\delta T} \tag{2.4}$$

For specific conditions where either pressure or volume is constant throughout the change of state, the specific heat could have two forms: constant pressure specific heat c_p or constant volume specific heat c_v as described by Equations 2.5 and 2.6.

$$\mathbf{c}_p = \Big(\frac{\delta Q}{\delta T}\Big)_p = \Big(\frac{\partial U}{\partial T}\Big)_p + p\Big(\frac{\partial v}{\partial T}\Big) \tag{2.5}$$

$$\mathbf{c}_v = \Big(\frac{\delta Q}{\delta T}\Big)_V = \Big(\frac{\partial U}{\partial T}\Big)_V \tag{2.6}$$

The second law of thermodynamics in classical thermodynamics concerns about another property of fluid in equilibrium, the entropy S. This parameter represents the number of different microscopic configurations a system with macroscopic variables could acquire. During a reversible process from a state of equilibrium to another, the second law states that the total entropy of the system can never decrease. The consequence of this is the total entropy in the universe will always keep rising over time. An increase in entropy is proportional to amount of heat given to a fluid in equilibrium, and depends on the temperature of the system:

$$T\delta S = \delta Q = \delta U + p\delta V \tag{2.7}$$

One other property worth discussing is the *Helmholtz free energy*, F. It describes the amount of energy required to recreate a system in an environment with a defined temperature. If the temperature of the environment is T, and the final entropy of the system is S, the the Helmholtz free energy of the system is:

$$F = U - TS \tag{2.8}$$

In a multiphase flow, especially on porous media such as in hydrocarbon reservoirs, the thermodynamic processes could be described by using equations of conservation (Hassanizadeh and Gray, 1990). For multiphase flow with phases o (for oil) and w (for water), there are conservation of mass, conservation of momentum, and conservation of energy that need to be considered. The conservation of mass is defined as the following:

$$\frac{\partial(\phi s_o \rho_o)}{\partial t} + \phi s_o \rho_o (\nabla \cdot \vec{v}_o) = \sum_{w \neq o} \hat{e}^o_{ow} \tag{2.9}$$

Here we encounter some important properties of multiphase flow in porous media: the porosity ϕ which denotes amount of pore space in the bulk media in which the fluid flow occurs, phase fluid saturation s_o which represents the volume fraction of phase o within the pore space, $\hat{v}_o$ that represents vector of Darcy velocity for phase o, and $\hat{e}^o_{ow}$ is the mass transfer rate into phase o through the interface between o and w. Conservation of momentum is described in the following equation:

$$\phi s_o \rho_o \frac{\partial \vec{v}_o}{\partial t} - \nabla \cdot (\phi s_o t_o) - \phi s_o \rho_o g_o = \sum_{w \neq o} \hat{T}^o_{ow} \tag{2.10}$$

In Equation 2.10, t_o is the stress tensor of phase o which represents pressure exerted by phase o in all directions, and $\hat{T}^o_{ow}$ is momentum transfer rate into phase o through the interface between o and w. Finally, the equation for conservation of energy is:

$$\phi s_o \rho_o \frac{\partial E_o}{\partial t} - \phi s_o t_o \colon \nabla v_o - \nabla \cdot (\phi s_o \vec{\phi}_{q_o}) - \phi s_o h_o = \sum_{w \neq o} \hat{Q}^o_{ow} \tag{2.11}$$

In Equation 2.11, note that there is a double dot notation between two tensors t_o and v_o, which multiplies the two tensors to produce a scalar result, much like the dot notation in vector multiplication. Also, $\vec{q}_o$ is heat flux (flow of energy per unit area per unit of time) of the phase, h_o is external energy supplied into the phase, and $\hat{Q}^o_{ow}$ is energy transfer rate into phase oa through the interface between o and w.

The second law of thermodynamics can also be applied to the multiphase flow system such that a balance of entropy for a specified phase, for example oil, could be described as the following:

$$\phi s_o \rho_o \frac{\partial S_o}{\partial t} - \nabla \cdot (\phi s_o \vec{\varphi}_o) - \phi s_o b_o = \sum_{w \neq o} \hat{\Phi}^o_{ow} + \Lambda_o \tag{2.12}$$

Notations regarding entropy in Equation 2.12 include the entropy of the phase itself, S_o, the entropy flux $\vec{\varphi}_o$, external entropy supply b_o, entropy transfer rate into phase o through $o-w$ interface $\hat{\Phi}^o_{ow}$, and rate of net entropy increase Λ_o.

Regarding internal energy of the system in multiphase flow, a thermodynamic theory has been developed (Hassanizadeh and Gray, 1990) that employs Helmholtz free energy F to represents the amount of internal energy possessed by each phase. F for each phase p in the system are defined as a function of their density, temperature, and saturation:

$$\left(\frac{\partial F_p}{\partial \rho_p}\right)_{T_p,s_p} = \frac{p_p}{(\rho_p)^2} \quad ; \quad p = \{o,w\} \tag{2.13}$$

$$\left(\frac{\partial F_p}{\partial T_p}\right)_{\rho_p,s_p} = -S_p \quad ; \quad p = \{o,w\} \tag{2.14}$$

$$\left(\frac{\partial F_p}{\partial s_p}\right)_{\rho_p,T_p} = -\frac{\Omega_p}{s_p \rho_p} \quad ; \quad p = \{o,w\} \tag{2.15}$$

The notation Ω_p in Equation 2.15 denotes a parameter called "wettability potential", which is the Helmholtz free energy dependence on saturation change. The three equations above provide a basis for internal energy and energy balance calculations in multiphase flow in porous media, for example groundwater aquifers and hydrocarbon reservoirs.

2.1.2 Energy forms involved in reservoir recovery

As depicted in Figure 1.3, if we consider a reservoir as a closed system then there are generally three energy forms involved: external energy, internal energy, and dissipated energy. External energy could be thought of supply (or discharge) of energy that comes from (or heading to) other systems in connection with the reservoir, which is usually production and injection wells, with addition of aquifers in some cases. Internal energy of the reservoir is related to its contents, mainly the various phases of fluids that resides within. Each of these phases has compressibility that causes it to expand or contract as reservoir pressure changes, and these volume alterations indicate changes in the internal energy of the system. Dissipated energy is heat released to surrounding environment due to work performed by the system to flow fluids throughout the reservoir. Flow is driven by difference in pressure from one point to another, and in turn this pressure drop dissipates some amount of energy stored in the flowing fluid.

Let there be a simple block reservoir containing two phases, water (w) and oil (o). One end of the reservoir is directly connected to a large aquifer of w, and the other end to a large aquifer of o. If a pressure difference exists between the two ends, then flow will occur, replacing one phase with another. Figure 2.1 gives such example where phase w (colored blue) gradually flows into the reservoir (colored gray), replacing o (colored orange). The surrounding aquifers then give the following external work on the reservoir, based on Equation 2.2:

$$dW_{external} = p_w dV_w - p_o dV_o \tag{2.16}$$

Furthermore, the bottom part of Figure 2.1 implies that the amount of displaced fluid equals to those of displacing fluid. Thus, $dV_w = -dV_o$. The reservoir itself does not

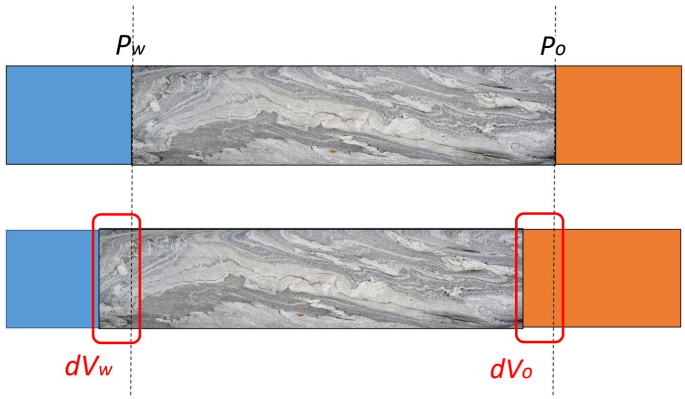


Figure 2.1: Illustration of external work to a reservoir system. Pressure difference exists a moment before displacement occurs (top). After displacement for some time (bottom), the displacing and displaced volumes are given by the red circles. Modified from Khanamiri et al. (2018).

actually move to the left as the figure might suggest, but rather the point of view for the bottom part has been moved to the right in order to better observe the volumes that flow in and out of the system. Equation 2.16 then could be formulated further:

$$dW_{external} = (p_w - p_o)dV_w = -\phi V(p_w - p_o)ds_w \tag{2.17}$$

where V represents bulk volume of the reservoir and s_w denotes saturation of water in the reservoir. Within a real reservoir in a field, these external works are done by sink/source spots such as production and injection wells, and/or external aquifers that regularly supply energy into the reservoir. Here one form of energy involved in reservoir recovery is defined, the **external energy** that carries flow between the reservoir system and other external system(s). For a well in connection with the reservoir, the amount of external energy change (dE_{ext}) it does to the system depends on its bottom hole pressure (p_{bh}) and volume of fluid exchanged between the well and the reservoir (V), based on Equation 2.16. The following equations describe external energy change for injection and production wells, respectively:

$$dE_{ext,inj} = p_{bh,inj}V_{inj} \tag{2.18}$$

$$dE_{ext,prod} = -p_{bh,prod}V_{prod} \tag{2.19}$$

Next, there is one energy change that is substantial in reservoir which is the **dissipated energy**. This is not actually an energy form in itself, but rather loss of energy as heat to the environment due to thermodynamics processes that happen within a system. Energy dissipation affirms the second law of thermodynamics that concerns how in a reversible process (such as fluid flow) the net entropy change can never decrease, and in consequence some amount of energy will be converted to heat (dissipated) during the process. As fluid moves

across a pressure gradient, it gradually loses energy. The bigger the pressure drop is, the more energy the fluid loses as defined by the following equation (Kundu et al., 2015):

$$dE_{dis} = q\, dp \tag{2.20}$$

where q is volumetric fluid flow rate. There are two instances where dissipation is important during reservoir recovery. First, as discussed before the flow of fluid passes through various pressure difference throughout the reservoir, and as a result energy dissipation is present anywhere flow occurs. Dissipation is also considered at the bottomhole part of a well. Equations 2.18 and 2.19 state that external energy change depends on bottom hole pressure p_{bh}. However, there exists a usually considerable pressure difference between the well bottomhole and the formation in order to drive flow in or out of the well. Then, for cases of injecting well inj and producing well $prod$ where formation pressure near the well opening is p_f, the dissipation in the well opening is:

$$dE_{dis,inj} = q_{inj}(p_{bh,inj} - p_f) \tag{2.21}$$

$$dE_{dis,prod} = q_{prod}(p_f - p_{bh,prod}) \tag{2.22}$$

Finally, the **internal energy** of the system is defined, which is the stored form of energy within the system, and could also be thought as "potential energy" of the reservoir. Reservoir fluid needs energy in order for it to flow throughout the formation, but the fluid itself could also lose or receive energy even when at rest, and this is what the definition of internal energy shall focus on. It is quite difficult to determine the value of internal energy of a system, like reservoir, but the internal energy *change* during a process could be calculated. Recall Equations 2.1 and 2.2 that state internal energy change as total change due to heat and work attributed to the system. Based on these equations, then internal energy change (dE_{int}) for a reservoir system that is subjected to work from wells and has lost some energy as dissipation is:

$$\begin{aligned} dE_{int} &= -dE_{dis} + dE_{ext} \\ &= -q_{res}\, dp + p_{bh,inj}\, q_{inj} - p_{bh,prod}\, q_{prod} \end{aligned} \tag{2.23}$$

The fluid flow rates in reservoir and in each wells could be different, as seen in Equation 2.23, due to the fact that each reservoir fluid has a degree of compressibility, which determine how much its volume will change when pressure is changed. For a constant temperature (isotherm), fluid compressibility is defined as (Whitson and Brulé, 2000):

$$c_f = -\Big(\frac{1}{V_f}\frac{dV_f}{dp}\Big) \tag{2.24}$$

Among the three phases commonly found in hydrocarbon reservoirs, gas generally exhibits the highest compressibility, often a few order magnitude higher than oil and water (Whitson and Brulé, 2000). Meanwhile, between oil and water, usually oil has a higher compressibility. Figure 2.2 presents isothermal compressibility values against pressure for a selected number of reservoir fluids. Total compressibility of a reservoir block could be determined by taking the compressibility and saturation of each phase. For example, in a water-oil system:

$$c_t = c_w\, s_w + c_o\, s_o \tag{2.25}$$

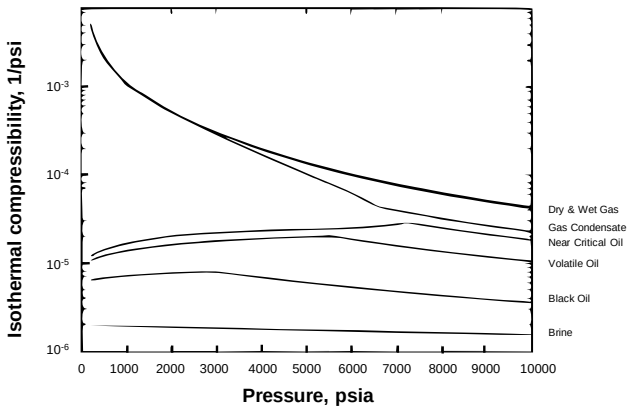


Figure 2.2: Isothermal compressibility for various reservoir fluids at different pressures. Modified from Whitson and Brulé (2000).

Reservoir rock constitutes most of total reservoir volume, and even though it is solid, for some formation such as unconsolidated sandstone the compressibility is actually at a same order of magnitude as water and oil and thus could not be ignored. Including rock compressibility into Equation 2.25, the result will be as the following:

$$c_t = c_\phi \left(1 - \phi\right) + \left(c_w \, s_w + c_o \, s_o\right) \phi \tag{2.26}$$

where ϕ is porosity of the system and c_ϕ is rock compressibility. c_ϕ has the same definition as fluid compressibility (Equation 2.24), but uses rock volume instead of fluid volume.

To summarize all the energy forms described in this subsection, an example is built using a two-block reservoir model connected to wells in each end of the model. One well injects fluid in and the other produces fluid out of the system. Imagine that the process has taken place for enough time until flow has reached steady-state condition, meaning that there are no more pressure change within the system. Figure 2.3 illustrates the example in detail. The system is divided to five parts: (1) is the bottomhole of the injection well, (2) is the pore space within the first block, (3) is fluid flow that occurs between the two blocks, (4) is the pore space within the second block, and (5) is the bottomhole of the production well. As the condition is steady-state flow, the bottomhole pressures (p_{inj} and p_{prod}) and the reservoir pressure are constant. The two blocks have different pressures p_{b1} and p_{b2} to allow interblock flow. During a specific amount of time dt, the volume of fluid injected in (1) is V_{inj}, the volume transferred between first and second block (3) is V_f, and the volume produced in (5) is V_{prod}. Volumes in (1), (3), and (5) are given at in-situ conditions, not at standard condition. This is to avoid converting the flow volume at each point using fluid compressibility, thus making the calculations much simpler. Using equations described in this subsection for external, internal, and dissipated energy, the following Table 2.1 shows energy change rate (dE/dt) in each part of the system.

Using Table 2.1, it can be proven that energy balance is achieved in the model. The sum

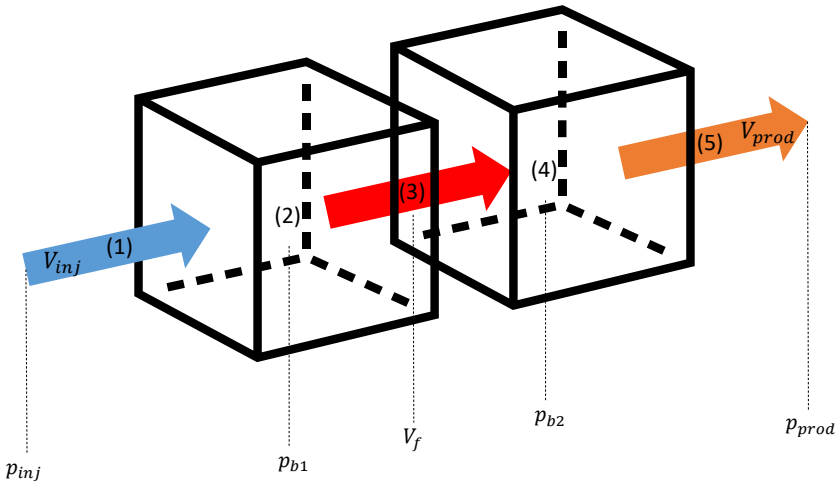


Figure 2.3: An example of a simple two-block reservoir connected to two wells. All three energy forms are present in this figure.

Energy change rate	External	Internal	Dissipation
(1)	$p_{inj}\frac{V_{inj}}{dt}$	0	$(p_{inj}-p_{b1})\frac{V_{inj}}{dt}$
(2)	0	$p_{b1}\frac{(V_{inj}-V_f)}{dt}$	0
(3)	0	0	$(p_{b1}-p_{b2})\frac{V_{inj}}{dt}$
(4)	0	$p_{b2}\frac{(V_f-V_{prod})}{dt}$	0
(5)	$-p_{prod}\frac{V_{prod}}{dt}$	0	$(p_{b2}-p_{prod})\frac{V_{inj}}{dt}$

Table 2.1: Energy change rate for the process occurring in the example depicted by Figure 2.3.

of each column for external, internal, and dissipation energy change rate is:

$$\frac{dE_{ext,total}}{dt} = p_{inj}\,V_{inj} - p_{prod}\,V_{prod} \tag{2.27}$$

$$\frac{dE_{int,total}}{dt} = p_{b1}\,(V_{inj} - V_f) + p_{b2}\,(V_f - V_{prod}) \tag{2.28}$$

$$\frac{dE_{dis,total}}{dt} = p_{inj}\,V_{inj} + p_{b1}\,(V_f - V_{inj}) + p_{b2}\,(V_{prod} - V_f) - p_{prod}\,V_{prod} \tag{2.29}$$

Substracting the total external and total internal change will yield a similar result to total dissipation, showing that energy balance as defined in Equation 1.1 is achieved for the system. The reservoir model could be further complicated by dividing it to multiple blocks, adding more wells, or even expanding the dimensionality of the reservoir to a 2D or 3D model. On a model with multiple blocks, dissipation term inside the reservoir will be calculated for each interblock flow, meaning that if a block flows fluid to three other blocks in three different directions (common case for a 3D reservoir simulation), then there are three dissipation terms that need to be calculated. Energy balance should still be achieved

for more complicated models based on this example, provided there are no other systems or processes that are not previously defined by this illustration.

2.2 Reservoir Simulation

Here the fundamental principles of reservoir simulation will be elaborated, starting from the reasoning behind reservoir simulation, to basic technical concepts such as properties and Darcy's law, and finally description of equations that dictate multiphase flow process within a reservoir, using finite-difference method for numerical approach in order to be able to simulate multiphase flow using a computer program.

2.2.1 The need for reservoir simulation

Reservoir simulation is an act of utilizing computers to simulate fluid flow in a hydrocarbon reservoir model (Berg and Slotte, 2020), by solving complex equations through the use of physical, mathematical, computer programming, and reservoir engineering knowledge (Ertekin et al., 2001). Reservoir simulation is necessary because petroleum engineers, especially reservoir engineers, have to obtain an accurate depiction of reservoir performance under various operating conditions. Since hydrocarbon recovery projects pose high risks in costs and safety, these risks have to be assessed thoroughly and minimized as far as possible. By taking into consideration the results from simulating a hydrocarbon reservoir, the engineers could get a proper picture of how the reservoir would undergone recovery, predicting if any unwanted occurrence could happen, and how to mitigate them. Most of the risks would exist simply because the reservoir has specific characteristic that could hinder the recovery process, for example heterogeneous and anisotropic rock properties could lead to greatly varying permeability values across the reservoir that might give recovery much less than what was initially predicted. One should consider using reservoir simulation to acknowledge risks before venturing further into the development phase of hydrocarbon recovery.

Advancements in computing technology lead to the widespread use of reservoir simulation tools in the petroleum industry. Only a few decades ago, only a few high-end computers would be able to run a computationally-heavy reservoir simulator. Today, nearly every personal computer could run a reservoir simulation tool with great speed and accuracy. At its core, reservoir simulator solves partial differential equations (PDE's) that would later be developed into a set of algebraic mathematical equations. These equations should have appropriate boundary and initial conditions in order to be able to approximate the behavior of the reservoir. The flow of different fluid phases (oil, water, gas) and mass transfers are also represented by the same equations. Another important equation, Darcy's law, depicts effects of different acting forces in the reservoir (gravity, viscous, capillary).

Reservoir simulation is generally performed in the following steps, according to Ertekin et al. (2001):

1. **Set the objectives**. The objectives have to be clear, realistic, and compatible with

available data. The objectives are used to set strategy, identify resources, and determine lessons to be learned.

2. **Acquire and validate all reservoir data**. After the objectives are set, reservoir data are gathered and incorporated into the reservoir model.
3. **Construct the reservoir model**. The reservoir model is built in the form of stacked grid blocks or cells. Figure 2.4 shows one example of representing real physical feature in terms of gridblocks. Properties of the reservoir are assigned to each of the cells. Even though in reality reservoir properties differ from one point to another, in the reservoir model the properties are assumed to be constant within a grid block.
4. **History match the reservoir model**. The simulation model could be improved with the help of production data, if available. The tuning process using historical production data is called ”history matching”.
5. **Run prediction cases.** Finally, various production schemes are evaluated and sensitivity analyses are conducted on different production and reservoir parameters.

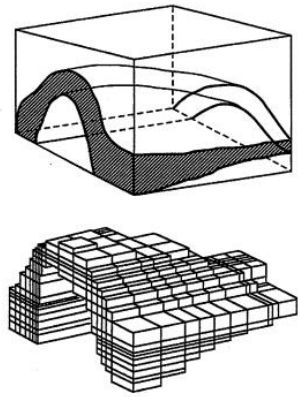


Figure 2.4: Three-dimensional gridblocks representation of an anticline. Taken from Ertekin et al. (2001)

2.2.2 Basic reservoir engineering concepts

In order to be able to model flow problems in reservoir, understanding basic concepts of reservoir engineering is important. This subsection will discuss some of these concepts: rock and fluid properties, fluid potential, and finally Darcy’s law for fluid flow in porous media.

Rock properties
Basic rock properties discussed here are assumed to be independent of fluid flowing through it, and no chemical reaction occurs between the rock and the fluid.

1. **Porosity**, symbolized as ϕ, is defined as the ratio of pore space in a rock sample to the total volume of the rock sample. In reservoir simulation calculations, only interconnected pore spaces comes into concern.

2. **Permeability**, symbolized as k, is the capacity of a porous medium to transmit fluids through its interconnected pores. Generally, the permeability used in reservoir simulation calculation is the horizontal permeability, k_H, since flow in reservoirs are usually horizontal. However, for many cases vertical permeability k_V would also become important. In general, $k_H > k_V$.

Fluid properties
The properties described here are independent of the rock that it flows through. Fluid properties generally are highly dependent on reservoir pressure and temperature. The three types of fluid used in reservoir simulation calculations are oil, water, and gas (denoted as o, w, and g respectively).

1. **Fluid density**, symbolized as ρ, is defined as mass per unit volume. The density in reservoir conditions can be found from density in standard conditions (sc) and FVF

$$\rho_l = \frac{\rho_{l,sc}}{B_l} \tag{2.30}$$

   For gas, density can be found from real-gas law

$$\rho_g = \frac{pM}{zRT} \tag{2.31}$$

   where M is molar mass of the gas, z is gas deviation factor, R is gas constant with an approximate value of 8.314 J/(mol $\cdot$ K), and T is the gas temperature.

2. **Solution-gas/liquid ratio**, symbolized as R_s, is the volume of gas that must dissolve into a unit volume of liquid (both volumes measured at standard condition), for the liquid and gas system to reach equilibrium at reservoir conditions.

3. **Formation volume factor (FVF)**, symbolized as B_l, where $l = o, w, g$, is the ratio of volume of phase l at reservoir conditions to its volume at standard conditions.

4. **Fluid compressibility**, symbolized as c_l for liquid and c_g for gas, is defined as change in relative volume of a unit mass as a result of change in pressure at constant temperature. Liquid compressibility of liquid l is defined as

$$c_l = \frac{1}{\rho_l}\left(\frac{\partial \rho_l}{\partial p}\right)_T \tag{2.32}$$

   While for gas, compressibility is

$$c_g = \frac{1}{p} - \frac{1}{z}\left(\frac{\partial z}{\partial p}\right)_T \tag{2.33}$$

   ρ in Equations 2.32 and 2.33 is density, while z in Equation 2.33 is gas-compressibility factor that defines deviation of a real gas from an ideal gas.

5. **Fluid viscosity**, symbolized as μ, is the measure of how easy the fluid would flow under applied pressure. A more viscous fluid would be more difficult to move under the same amount of force. In general, when comparing the three phases, the order from least viscous to most viscous would be: gas, water, oil.

Fluid-Rock properties
This segment discusses two important properties that depends on both fluid and rock in the reservoir.

1. **Fluid saturation**, symbolized as S_l, is the fraction of porosity occupied by phase l, where $l = o, w, g$. For all available phases in pore space, then the total saturation would be 1. In multiphase flow with oil, gas, and water this would be

$$s_w + s_o + s_g = 1 \tag{2.34}$$

2. **Capillary pressure**, symbolized as P_c, is the result of capillary forces acting in small openings in pore systems between two or more phases. Capillary pressure can be defined as the pressure of the non-wetting phase minus the pressure of the wetting phase. For oil/water systems, commonly oil would be the non-wetting phase while water is the wetting phase. The capillary pressure between oil and water is then

$$P_{cow} = p_o - p_w \tag{2.35}$$

   Capillary pressure is a function of saturation and also history of saturation (due to hysterisis effects), and differs for various reservoir rocks and fluids.

3. **Relative permeability**, symbolized as k_r, is the ratio of effective permeability of a phase flowing in multiphase flow to the absolute permeability of the medium. Relative permeability is generally modeled as a function of saturation between two phases (for example, oil/water and oil/gas relative permeability).

Fluid Potential
Fluid potential, Φ, is defined as the work required to transfer a mass of fluid from a state of atmospheric pressure and reference elevation/datum to the point in question. When comparing fluid potential in different points in space, the fluid would flow from a higher potential point to a lower potential one. Fluid potential in the form of an equation is

$$\Phi = p - \gamma Z \tag{2.36}$$

where p is pressure at that datum, γ is fluid gravity, and Z is the distance from reference elevation (positive value for vertical downward direction). For any arbitrary point with a new datum, fluid potential could be written as

$$\Phi - \Phi^o = (p - p^o) - \gamma Z \tag{2.37}$$

where values with superscript o denotes those at datum, and values with no superscripts define values at the arbitrary point. The potential gradient, obtained by differentiating Equation 2.37, is

$$\vec{\nabla}\Phi = \vec{\nabla}p - \gamma\vec{\nabla}Z \tag{2.38}$$

While for multiphase flow, the potential gradient for each phase (oil, water, and gas) is

$$\vec{\nabla}\Phi_l = \vec{\nabla}p_l - \gamma_l\vec{\nabla}Z \tag{2.39}$$

where l can be o for oil, w for water, or g for gas.
It is imperative to remember that Z is positive in vertical downward direction, since in the realm of reservoir engineering (and petroleum engineering in general), the term for distance increases with increasing depth.

Darcy's Law
Henry Darcy formulated this law based on his experimental results in 1856. Darcy's law has been widely used to describe the movement of other fluids, including two or more phases, in consolidated rocks and other porous media (Craft et al., 1991). Darcy's law states that the apparent velocity (flow rate over a cross-area perpendicular to flow direction) of a single-phase fluid in a porous medium, u, is proportional to the permeability in the direction of flow, k, and the fluid potential gradient, $\vec{\nabla}\Phi$, while inversely proportional to the fluid viscosity, μ:

$$\vec{v} = -\frac{k}{\mu}\vec{\nabla}\Phi \tag{2.40}$$

Using the definition of velocity as flow rate, q, over a cross-sectional area perpendicular to the flow, A, and the definition of fluid potential gradient from Equation 2.38, Darcy's law can be reformulated to

$$\frac{\vec{q}}{A} = -\frac{k}{\mu}\left(\vec{\nabla}p - \gamma\vec{\nabla}Z\right) \tag{2.41}$$

However, when using Equation 2.41, it is necessary to remember there are assumptions and limitations imposed:

1. The fluid is single-phase, homogeneous, and Newtonian.

2. No chemical reaction occurs between the fluid and the porous medium.

3. Permeability is property of the medium, it is independent of the type of fluid, and the system's pressure and temperature.

4. The flow condition is laminar.

For multiphase flow, another term is added, which is relative permeability (k_r). Relative permeability of a phase can be found by dividing the effective permeability of the phase to the absolute permeability of the medium. Effective permeability is the permeability of a phase in a multiphase flow. The sum of effective permeabilities of phases present is always less than the absolute permeability (Dake, 1978). Equation 2.41 for phase $l = o, w, g$ in multiphase flow would become

$$\frac{\vec{q_l}}{A} = -\frac{kk_{rl}}{\mu_l}\left(\vec{\nabla}p_l - \gamma_l\vec{\nabla}Z\right) \tag{2.42}$$

2.2.3 Development of Multiphase Flow Equation

Starting from single-phase flow equation, for single-phase flow in porous media it is obtained by combining Darcy's law and the law of mass conservation. Darcy's law have been explained previously in Subsection 2.2.2. The law of mass conservation describes the material balance in a control volume. For any component c, the material balance is expressed as

$$(m_i - m_o)_c + (m_s)_c = (m_a)_c \tag{2.43}$$

In Equation 2.43, m_i denotes mass entering the control volume (mass in), m_o denotes mass leaving the control volume (mass out), m_s denotes mass entering/leaving the control volume via external pathways, in petroleum reservoir systems usually by wells (sink/source), and m_a is accumulated mass in the control volume. For single-phase three-dimensional flow, using a rectangular coordinate systems (x, y, z), the mass-conservation equation is

$$\begin{aligned} &-\frac{\partial}{\partial x}(\dot{m}_x A_x)\Delta x - \frac{\partial}{\partial y}(\dot{m}_y A_y)\Delta y - \frac{\partial}{\partial z}(\dot{m}_z A_z)\Delta z \\ &= V_b \frac{\partial}{\partial t}(m_v) - q_m \end{aligned} \tag{2.44}$$

In Equation 2.44, $\dot{m}$ is mass flux, A is the cross-section perpendicular to the dimension direction, V_b is the volume of the system, m_v is the total mass in the system, and q_m is mass entering/exiting via sink/source.
Combining Equation 2.41 (Darcy's law) and Equation 2.44 (Mass conservation law), the basic equation for single-phase flow can be developed

$$\begin{aligned} &\frac{\partial}{\partial x}\left[\frac{k_x A_x}{\mu_l B_l}\left(\frac{\partial p}{\partial x} - \gamma_l \frac{\partial Z}{\partial x}\right)\right]\Delta x + \frac{\partial}{\partial y}\left[\frac{k_y A_y}{\mu_l B_l}\left(\frac{\partial p}{\partial y} - \gamma_l \frac{\partial Z}{\partial y}\right)\right]\Delta y \\ &+ \frac{\partial}{\partial z}\left[\frac{k_z A_z}{\mu_l B_l}\left(\frac{\partial p}{\partial z} - \gamma_l \frac{\partial Z}{\partial z}\right)\right]\Delta z = V_b \frac{\partial}{\partial t}\left(\frac{\phi}{B_l}\right) - q_{l,sc} \end{aligned} \tag{2.45}$$

where $l = o, w, g$. Equation 2.45 is the fundamental equation in the development of reservoir simulation. For a slightly compressible system usually found in reservoir systems, the accumulated mass term becomes

$$V_b \frac{\partial}{\partial t}\left(\frac{\phi}{B_l}\right) = \frac{V_b \phi c_l}{B_l^o}\frac{\partial p}{\partial t} \tag{2.46}$$

where B^o denotes FVF at a reference pressure.

Finite-difference is used in reservoir simulation as for approaching analytical equations in a numerical manner. The basis of finite-difference concept is a sequence of mathematical operations that occurs at discrete points, whether in space or in time. Based on the location of the reference point in the calculation, finite-difference operations can be forward-difference, backward-difference, or central-difference. The most widely used application of finite-difference is in derivative analysis. The forward-difference definition of first derivative of $f(x)$ at point x_i is

$$\frac{df}{dx} \approx \frac{f(x_{i+1}) - f(x_i)}{h} \tag{2.47}$$

The approximation on Equation 2.47 will be better as h approaches zero.

Industry-standard reservoir simulation tools in general use a rectangular coordinate system that discretizes the system into block-shaped grids, called gridblocks/cells. Figure 2.5 illustrates the rectangular coordinate system for three-dimensional flow. Taking for example

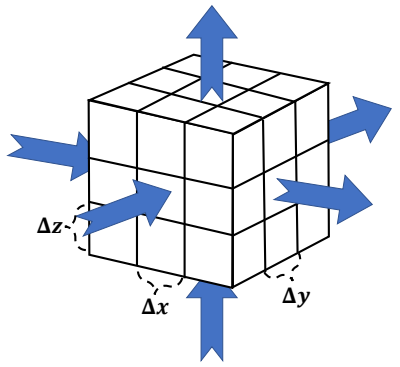


Figure 2.5: Rectangular coordinate system for 3D flow using (x,y,z) coordinate. The blue arrows indicate flow directions in all three dimensions. Modified from Ertekin et al. (2001)

one-dimensional flow in x direction from Equation 2.45, and knowing that $\partial Z/\partial x = 0$, then the basic flow equation can be discretized in space using central-difference into

$$\begin{aligned}\left(\frac{A_x k_x}{\mu_l B_l \Delta x}\right)_{i+1/2}(p_{i+1}-p_i) - \left(\frac{A_x k_x}{\mu_l B_l \Delta x}\right)_{i-1/2}(p_i - p_{i-1}) \\ + (q_{l,sc})_i = \left(\frac{V_b \phi c_l}{B_l^o}\right)_i \frac{\partial p_i}{\partial t}\end{aligned} \tag{2.48}$$

This can then be simplified into

$$T_{lx,i+1/2}(p_{i+1}-p_i) - T_{lx,i-1/2}(p_i - p_{i-1}) + (q_{l,sc})_i = \left(\frac{V_b \phi c_l}{B_l^o}\right)_i \frac{\partial p_i}{\partial t} \tag{2.49}$$

The coefficient T is called "transmissibility", and is a property of the porous medium and the flowing fluid.
For the time derivative, let us discretize at different time levels $n+1$ for next time step and n for the current time. Continuing from Equation 2.48, by applying time discretization the equation becomes

$$\begin{aligned}T_{lx,i+1/2}(p_{i+1}^n - p_i^n) - T_{lx,i-1/2}(p_i^n - p_{i-1}^n) \\ + (q_{l,sc})_i = \left(\frac{V_b \phi c_l}{B_l^o \Delta t}\right)_i (p_i^{n+1} - p_i^n)\end{aligned} \tag{2.50}$$

So far, the discussion has only concerned itself with single-phase flow. From here, the

flow condition in which multiple phase flow together will be discussed. There are a lot of differences that arise from adding more phases to the flow system, but the underlying equations and concepts used are similar, and they only need to be extended accordingly. The goal now is to find the flow equation for each of the phases (oil,water,gas) based on the basic single-phase flow equation, which is Equation 2.45. For oil and water, simply add relative permeability and saturation terms for each phase.

$$\begin{aligned}\frac{\partial}{\partial x}\left[\frac{k_x k_{ro} A_x}{\mu_o B_o}\left(\frac{\partial p_o}{\partial x}-\gamma_o\frac{\partial Z}{\partial x}\right)\right]\Delta x+\frac{\partial}{\partial y}\left[\frac{k_y k_{ro} A_y}{\mu_o B_o}\left(\frac{\partial p_o}{\partial y}-\gamma_o\frac{\partial Z}{\partial y}\right)\right]\Delta y\\ +\frac{\partial}{\partial z}\left[\frac{k_z k_{ro} A_z}{\mu_o B_o}\left(\frac{\partial p_o}{\partial z}-\gamma_o\frac{\partial Z}{\partial z}\right)\right]\Delta z=V_b\frac{\partial}{\partial t}\left(\frac{\phi S_o}{B_o}\right)-q_{o,sc}\end{aligned} \tag{2.51}$$

$$\begin{aligned}\frac{\partial}{\partial x}\left[\frac{k_x k_{rw} A_x}{\mu_w B_w}\left(\frac{\partial p_w}{\partial x}-\gamma_w\frac{\partial Z}{\partial x}\right)\right]\Delta x+\frac{\partial}{\partial y}\left[\frac{k_y k_{rw} A_y}{\mu_w B_w}\left(\frac{\partial p_w}{\partial y}-\gamma_w\frac{\partial Z}{\partial y}\right)\right]\Delta y\\ +\frac{\partial}{\partial z}\left[\frac{k_z k_{rw} A_z}{\mu_w B_w}\left(\frac{\partial p_w}{\partial z}-\gamma_w\frac{\partial Z}{\partial z}\right)\right]\Delta z=V_b\frac{\partial}{\partial t}\left(\frac{\phi S_w}{B_w}\right)-q_{w,sc}\end{aligned} \tag{2.52}$$

For gas, however, another thing have to be considered. Recall the discussion regarding solution gas-oil ratio (R_s) in Subsection 2.2.2. According to the black-oil model, mass transfer is possible between the oil and gas phases. Gas components can dissolve in oil to some extent, depending on reservoir conditions. Therefore, the flow equation should represent not only gas components in the gas phase, but also gas components in the oil phase.

$$\begin{aligned}\frac{\partial}{\partial x}\left[\frac{k_x k_{rg} A_x}{\mu_g B_g}\left(\frac{\partial p_g}{\partial x}-\gamma_g\frac{\partial Z}{\partial x}\right)+R_s\frac{k_x k_{ro} A_x}{\mu_o B_o}\left(\frac{\partial p_o}{\partial x}-\gamma_o\frac{\partial Z}{\partial x}\right)\right]\Delta x\\ +\frac{\partial}{\partial y}\left[\frac{k_y k_{rg} A_y}{\mu_g B_g}\left(\frac{\partial p_g}{\partial y}-\gamma_g\frac{\partial Z}{\partial y}\right)+R_s\frac{k_y k_{ro} A_y}{\mu_o B_o}\left(\frac{\partial p_o}{\partial y}-\gamma_o\frac{\partial Z}{\partial y}\right)\right]\Delta y\\ +\frac{\partial}{\partial z}\left[\frac{k_z k_{rg} A_z}{\mu_g B_g}\left(\frac{\partial p_g}{\partial z}-\gamma_g\frac{\partial Z}{\partial z}\right)+R_s\frac{k_z k_{ro} A_z}{\mu_o B_o}\left(\frac{\partial p_o}{\partial z}-\gamma_o\frac{\partial Z}{\partial z}\right)\right]\Delta z\\ =V_b\frac{\partial}{\partial t}\left(\frac{\phi S_g}{B_g}+\frac{\phi R_s S_o}{B_o}\right)-q_{g,sc}\end{aligned} \tag{2.53}$$

Equations 2.51, 2.52, and 2.53 are the fundamental equations for each phase in multiphase flow. These equations can be further extended if the reservoir simulation concern more situations that could happen in real world application, such as capillary pressure between phases, reaction between rock and fluid, etc.

2.3 Software

Different tools used for this study are briefly detailed in this section. The various software detailed here are open source, meaning that anyone can freely access and distribute

them for any purpose. The Open Porous Media (OPM) initiative started in 2009 to encourage innovation and research on porous media simulation that is open and free for anyone (Rasmussen et al., 2019). Throughout the years, they have developed Flow for reservoir simulation and ResInsight for 3D visualization, and both will be discussed within this section.

2.3.1 OPM Flow as Reservoir Simulator

OPM Flow uses the black-oil model, in which the reservoir fluid consists of three phases (oleic, aqueous, gaseous) and three components (oil, water, gas). Oil and gas components can be found on oleic and gaseous phases, and component transfer between the two phases are allowed. For the objective variables, OPM Flow chooses oil pressure (p_o), water saturation (s_w), and a variable third variable. The third variable could be gas saturation (s_g), solution gas-oil ratio (r_{go}), or solution oil-gas ratio (r_{og}), depending on the presence of oleic and/or gaseous phase. The choice of the third variable is made for each grid cell. For initialization, OPM Flow uses initial values of pressure and saturation specified by the user to define initial conditions, while boundary conditions use no-flow boundaries (Neumann boundary condition) as default. More recent versions have added some aquifer models as constant-pressure boundaries. Rock and fluid properties could be defined by including tables in the input file. From the tables, OPM Flow would interpolate or extrapolate to some degree the dependence of the properties to pressure or saturation.

OPM Flow has two different models for wells: standard well and multi-segment well. The standard well model describes flow in each well with one set of objective variables. The variables for one well in a three-phase system includes total flow rate Q_t, water F_w and gas F_g fractions, and bottom-hole pressure p_{bh}. The well and the reservoir are coupled to keep the system closed, by coupling mass conservation equations for each component in the reservoir and in the well. More limitations can also be imposed via well control, in which wells are only allowed to operate above a certain Q_t and/or p_{bh}. The multi-segment well model could be used to model more advanced wells such as multilateral wells or horizontal wells. In this model, the well is split into different segments, each having inlet and outlet nodes. The pressure of the nodes replace p_{bh} as a part of the objective variables. Each node can connect to two or more nodes from different segments, but each node can only be a part of one segment. For a node with multiple connections, it is important to make sure pressure of that node would result in the same value when calculating well equations for each connected segment.

The reservoir and well equations described in the reservoir and well model are subject to a fully-implicit solution method that can be simplified into

$$J(y_n)(y_{n+1} - y_n) = -R(y_n) \tag{2.54}$$

Each term in Equation 2.54 includes reservoir and well equations, as well as the coupling equations.

Developing the Jacobian matrix J requires the simulator to compute various partial derivatives in each iteration. Traditionally, the partial derivatives would be computed analytically

and would sometimes produce unreliable results as a result of manual calculations of partial derivatives, while also being time-consuming because hard-coding each partial derivative is inefficient. OPM Flow has employed a way to mitigate this problem, by the use of Automatic Differentiation (AD). The basic idea of AD is for a complicated derivation that needs to utilize the chain rule, AD would compute all sequence of values in the chain rule and the derivations of the sequences automatically (Neidinger, 2010). A simple example would be calculating the derivative of $y = \cos(x^2 + 1)$ at $x = 4$. Using AD, the software would not only calculate the left column in Table 2.2, but also the right column automatically.

Variables	Derivatives
$x = 4$	
$y_1 = x^2 + 1 = 17$	$y_1^{'} = 2x = 8$
$y = \cos(y_1) = -0.275$	$y^{'} = -\sin(y_1)y_1^{'} = 7.691$

Table 2.2: Example of Automatic Differentiation (AD) in action.

In order to be relevant and easily recognizable in the industry, OPM Flow uses a widely-used format when reading and writing files. The ECLIPSE simulator from Schlumberger is the dominant simulator in the industry, and OPM Flow has decided to support the I/O formats used by ECLIPSE. For input, OPM Flow reads from a `.DATA` file that contains various keywords that have different commands and/or specifications for the simulation. The details of each keyword and how to use them have been listed in the OPM Flow manual (Baxendale, 2019). The `.DATA` file is organized into several sections, each dealing with a specific part of the simulation. The sections are listed in Table 2.3.

Section	Function
`RUNSPEC`	Overall simulation settings (grid dimensions, phases present, etc.)
`GRID`	Grid geometry and petrophysical properties.
`EDIT`	Modifications and multipliers to the `GRID` section.
`PROPS`	Flow parameters, fluid properties (PVT), and rock properties.
`REGIONS`	Define different regions in the model that have different properties.
`SOLUTION`	Model initialization (Water saturation, pressure, fluid contacts, etc.)
`SUMMARY`	Choose variables to save as output.
`SCHEDULE`	Well control and parameters, time step definition.

Table 2.3: The sections in a `.DATA` file and their functions.

OPM Flow has a range of different output files. Summary files (`.UNSMRY` and `.SMSPEC`) contain well and reservoir data in time-series format that then can be viewed using visualization tools such as *ResInsight* or *ECLIPSE Office*. There is also a restart file (`.UNRST`) that can be used for restarting the simulation from an arbitrary checkpoint, alongside `.INIT` and `.EGRID` files that provide the initialization and grid model. There are also documentation files such as `.PRT` and `.DBG` that let users see the step-by-step process of

the simulation and lists of errors or warnings that occured throughout the run. All of these files are formatted to be similar to the ones used in ECLIPSE, so users already familiar with ECLIPSE would have no problem adjusting when they start working with OPM Flow.

The version of OPM Flow used in this study is version 2019.10, since it is the latest Flow version released at the start of this study. However, near the end of writing this thesis, a newer version (ver 2020.04) was released and added some new features and enhancements to the reservoir simulator (Baxendale, 2020).

2.3.2 OPM ResInsight as 3D Visualization Tool

ResInsight is another part of the Open Porous Media initiative, developed as a tool for cross-platform 3D visualization, curve plotting, and post processing for reservoir models and simulations (Dale et al., 2012). Specifically, ResInsight is compatible with models and simulations run using ECL format, for example Schlumberger Eclipse and OPM Flow, and can also visualize geomechanical simulations from ABAQUS with some configurations. ResInsight has four underlying submodules to support its functionality. There are Custom Visualization Core (CVC) that uses OpenGL for handling custom 3D visualization, Ensemble Reservoir Tool (ERT) for reading ECL simulation output files described at the previous section, Qt to facilitate cross-platform, and GNU Octave for complicated numerical computations. Figure 2.6 shows the system architecture of ResInsight, complete with its auxiliary modules. GNU Octave is colored differently to show that it is a separate program, but is used for aiding ResInsight functionality.

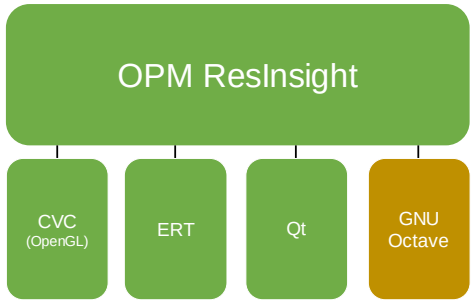


Figure 2.6: Complete system architecture of ResInsight, including underlying submodules. Modified from Dale et al. (2012).

ResInsight has two main windows: 3D main window and Plot main window. The 3D main window includes the following:

- **3D Main View** of the simulation, inputted to ResInsight from `.EGRID` file
- **Project Tree** that consists of all simulation objects (e.g. faults, wells) in a tree structure.

- **Property Editor** for displaying properties to the main view.
- **Result Info** that displays property result of an object selected in the main view.
- **Result Plot** that displays a plot of property result from Result Info for all timesteps.
- **Messages** for displaying other information and warnings related to the software operations.

Figure 2.7 presents an instance of 3D main window of ResInsight, displaying a simulated model of a real reservoir, Norne. The selected property displayed here is oil saturation.

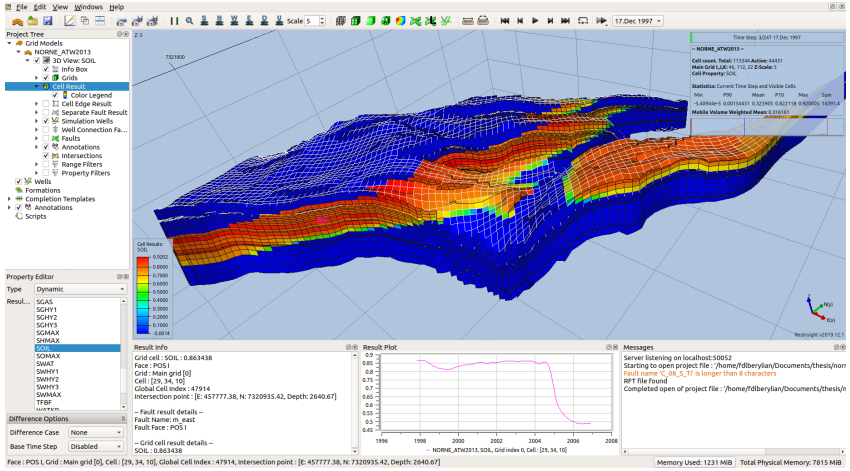


Figure 2.7: The 3D main window of ResInsight, presenting Norne simulation in the main view.

The Plot main window is used for 2D graphing and plotting, and has the following components:

- **Plot Main View** displays the selected plot or graph.
- **Plot Project Tree** contains all possible choice of plotting in a tree structure.
- **Property Editor** for viewing and editing plot properties.

Figure 2.8 presents an instance of Plot main window of ResInsight, displaying a plot of field oil production rate (FOPR) of Norne reservoir over time.
As can be seen in the Plot Project Tree, there are a few selections of plot in ResInsight, from summary plots that comes from **.SMSPEC** file, RFT plots, PLT plots, well log plots, and flow diagnostics plots that has results calculated by ResInsight.

One important advantage of using ResInsight comes from its capability for results post-processing and automation by scripting. The scripting interfaces supported by ResInsight include Python, Octave, and command line interface.

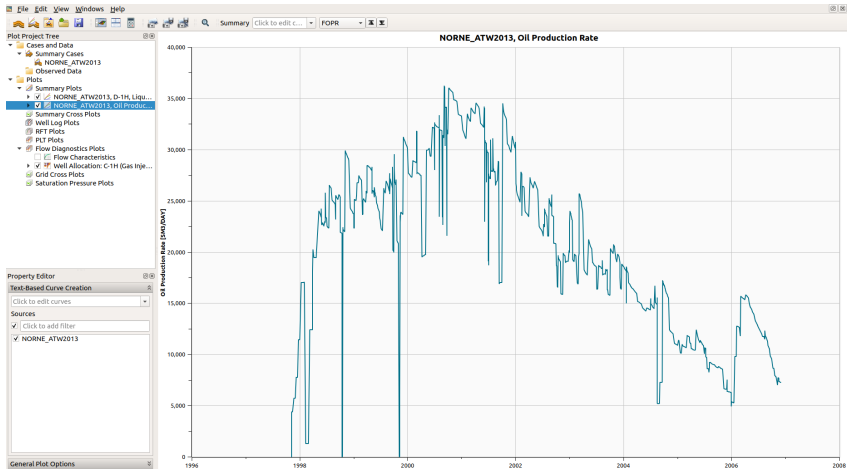


Figure 2.8: The Plot main window of ResInsight, displaying FOPR plot of Norne over time in the main view.

- **Python**: The version of Python supported is Python 3. Using this interface enables the user to extract results data to Python for further processing, and return them to be displayed in ResInsight. Python could also be used to directly communicate with ResInsight, giving the ability to control views, take snapshots, load data files, etc.
- **Command line**: Several command line parameters are supported that could help with case handling and task automation, simply by calling *resinsight –[parameter name]* in the command line/terminal interface. Plotting summary plots and taking snapshots are a few examples of things that could be done using command line scripting.
- **Octave**: Octave could be used for similar functionalities as Python, such as direct communication to an active ResInsight session, or execute scripts to extract results data. In addition to these, ResInsight provides a feature to manage and edit Octave scripts in a simple manner, directly accessible in the Project Tree on the 3D main window. However, the Octave interface dose not support Flow diagnostics, injection flooding results, and geomechanical cases.

Detailed references for using each of the scripting interfaces could be accessed in https://resinsight.org/scripting/.

The version of ResInsight used in this study is version 2019.12.1. Near the end of this study, a newer version 2020.04 was released but is not used for this thesis in order to avoid any inconsistencies and other incompatibilities that could arise due to software update.

Chapter 3

Methodology

This chapter presents in detail the steps taken to conduct this study: beginning from the description of steps taken to develop calculations for energy dissipation and energy balance in a reservoir model, then continuing with the implementation of the calculations in simple simulation cases, slowly adding complexities in the model to test the validity of the calculations, and finally the presentation of the calculation results from the test cases.

The energy calculations done in this study, both for dissipation and energy balance, are developed using Python scripting in ResInsight, as described in subsection 2.3.2. ResInsight has created detailed documentation for its Application Programming Interface (API) that allows it to interact with Python at https://api.resinsight.org/.

There are some prerequisites in order to be able to do Python scripting on ResInsight. First, as discussed in subsection 2.3.2, only Python 3 is compatible with ResInsight. This is because the necessary Python client package developed by ResInsight, called *rips*, only supports Python 3. Next, install the *rips* package. It is continually updated as newer versions of ResInsight are being rolled out, but for this study, both the ResInsight and the *rips* package are on version 2019.12.1. Finally, the user needs to configure the script server for connecting Python and ResInsight, in the *Preference* dialog tab within ResInsight. Figure 3.1 shows the dialog tab.

3.1 Development of Energy Calculation

Equations for each energy component in the reservoir are derived from the discussion surrounding Figure 2.3. For the current discussion, the model will be expanded from a two-block reservoir to a one-dimensional reservoir with multiple blocks in the x-direction. Starting from external energy change, the equations in Table 2.1 hold for wells connected to the reservoir system. In ECL-based reservoir simulation, volumes flowing through wells are usually defined in standard conditions. These need to be converted to subsurface conditions by using FVF for each specific phase. Since the fluids are compressible, the FVF

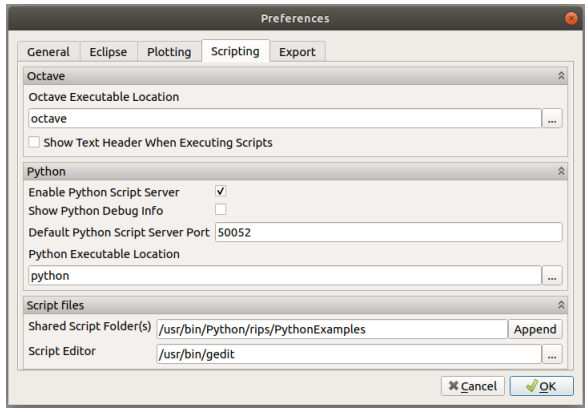


Figure 3.1: Scripting panel in the *Preferences* dialog tab within ResInsight. The Python scripting server here is defaulted to Port 50052.

will be different for wells with different bottom hole pressures.

Next is the energy dissipation calculation. For well bottom holes, once again, FVF is included in the equations if well fluid volumes are provided in standard condition. Dissipation calculation is based on Equation 2.20. From the equation, it can be seen that a nonzero pressure difference between two grid blocks yields some dissipation. The dissipation will always be positive since the flow rate will always be opposite of the pressure gradient direction. This adheres to the second law of thermodynamics: the entropy change of the system (in this case, total dissipated heat) will never be negative.

The last component to be calculated is internal energy change. For internal energy change at the reservoir blocks, the simulation gives pore space volume at reservoir conditions. This is in line with the example given in Figure 2.3, and there is no need to concern different compressibilities/FVF of each fluid. Internal energy change for each block depends on how much fluid it releases to and how much it receives from other connected blocks. Based on Equation 2.23, calculation of internal energy change for each individual reservoir block can be calculated as:

$$dE_{int} = p_b \left(q_{in} - q_{out} \right) \tag{3.1}$$

where p_b is the block pressure, q_{in} is the total fluid flow rate that goes into the block, and q_{out} is the total fluid flow rate that goes out to other blocks. The difference between q_{in} and q_{out} depends on the interblock pressure difference and the fluid itself. The volume of a more compressible fluid, such as gas, is highly dependent on pressure. A slight pressure difference could change the fluid volume significantly, resulting in a large difference between q_{in} and q_{out}. Thus, a more compressible fluid will experience higher internal energy change. For internal energy change due to wells, the well inflow and outflow rate will be attributed to q_{in} and q_{out}, respectively. As previously stated, well rates usually defined at standard conditions and need to be converted to reservoir conditions using fluid FVF.

Based on the discussion above, equations for energy calculations can be developed in Python scripting for ResInsight. The development adheres to OPM Flow simulation results, so some modifications would have to be done to equations defined previously, depending on data availability from OPM Flow output files. Starting simple with a one-dimensional reservoir that contains multiple grid blocks connected to one injection well and one production well, the calculations are done as follows:

1. **Connect Python and ResInsight**. In the Python script, connect to ResInsight first:

```
import rips

# Connect to ResInsight
resinsight = rips.Instance.find()
case = resinsight.project.cases()[0]
name = case.name
```

2. **Fetch data required for calculation in reservoir**. Energy dissipation calculation needs interblock flow rate and pressure in each block, as defined in Equation 2.20. However, simulation result using OPM Flow does not provide flow rate. Thus, it has to be computed manually using the transmissibility concept from Equations 2.48 and 2.49:

```
1 tranx_results = case.active_cell_property('STATIC_NATIVE', 'TRANX',
      0)
2 pres_results = case.active_cell_property('DYNAMIC_NATIVE', 'PRESSURE'
      , tstep)
3 krw_results = case.active_cell_property('DYNAMIC_NATIVE', 'WATKR',
      tstep)
4 kro_results = case.active_cell_property('DYNAMIC_NATIVE', 'OILKR',
      tstep)
5 muw_results = case.active_cell_property('DYNAMIC_NATIVE', 'WAT_VISC',
       tstep)
6 muo_results = case.active_cell_property('DYNAMIC_NATIVE', 'OIL_VISC',
       tstep)
```

   The `TRANX` parameter fetched in Line 1 is not the transmissibility as defined in Equation 2.49, but rather a normalization, transmissibility divided by k_r/μ. Hence relative permeabilities (`WATKR`, `OILKR`) and viscosities (`WAT_VISC`, `OIL_VISC`) are needed to calculate transmissibility. The same set of parameters will also be used to calculate internal energy change. Transmissibilities in ECL-based simulators are measured at reservoir conditions, so Equation 3.1 could be directly used for calculating internal energy.

3. **List wells in simulation**. Well data are not available to fetch from ResInsight, but it could be extracted directly from the `.UNRST` file. Thus, the script needs to read the restart file first and then list the wells:

```
rst_file = EclFile("%s.UNRST" % name)

# List wells in model
nwells = len(summary_file.wells())
well_list = []
for wel in range(nwells):
```

```
    welname = rst_file["ZWEL"][0][wel*3]
    welname = welname.rstrip()
    niwelz = int(len(rst_file["IWEL"][0]) / nwells)
    weli = rst_file["IWEL"][0][wel*niwelz]
    welj = rst_file["IWEL"][0][wel*niwelz + 1]
    welidx = int(weli-1)
    if rst_file["IWEL"][0][wel*niwelz + 6] == 1:
        weltype = "PROD"
    else:
        weltype = "INJE"

    well_list.append(well(welname,welidx,weltype))
```

The algorithm to extract well data such as name and location (idx) defined in the code listing above follows OPM Flow documentation that lists all well data in the `.UNRST` file (Baxendale, 2019).

4. **Fetch data required for calculation in well bottomhole**. Well bottomhole pressure (BHP) and inflow/outflow rate need to be fetched for every well. The injection well only injects water (WWIR), while the production well produces oil and water (WOPR and WWPR):

```
for wel in well_list:
        bhp = summary_file["WBHP:%s" % wel.name][tstep].value # Well
    BHP
        if wel.type == "PROD": # If production well
            wpr = summary_file["WWPR:%s" % wel.name][tstep].value
            opr = summary_file["WOPR:%s" % wel.name][tstep].value
        elif wel.type == "INJE": # If injection well
            wir = summary_file["WWIR:%s" % wel.name][tstep].value

```

5. **Calculate energy change in reservoir**. Energy changes in reservoir require calculation of flow rate, which uses parameters imported from ResInsight as defined previously in step 2. The calculation is done for every direction (only in x-direction for one dimensional case). Since the transmissibility is defined in the I+ direction (meaning in the positive direction along the x-axis), then the flow rate is calculated also in I+ direction for each block. Flow rate calculation is based on Equation 2.49, using the fetched parameters in step 2:

$$q_{n,n+1} = T_{x,n} \left(\frac{k_{r,w}}{\mu_w} + \frac{k_{r,o}}{\mu_o} \right) (p_n - p_{n+1}) \tag{3.2}$$

In Equation 3.2, $T_{x,n}$ is transmissibility of block n in positive x-direction (`TRANX`), k_r and μ are relative permeability and viscosity defined for each phase, p_n is grid block pressure, and p_{n+1} is pressure at the neighboring block in positive x-direction. The subscript $n, n+1$ on flow rate denotes the direction of flow, which goes from block n to block $n+1$. After calculating flow rate at each grid block, dissipation and internal energy change are determined using the following equations:

$$dE_{dis,n} = q_{n,n+1} (p_n - p_{n+1}) \tag{3.3}$$

$$dE_{int,n} = p_n \left(q_{n-1,n} - q_{n,n+1}\right) \tag{3.4}$$

In Equation 3.4, $q_{n-1,n}$ is flow rate from block $n-1$ to the observed block n. Pressures and transmissibilities in previous and next grid blocks have to be determined to calculate both $q_{n-1,n}$ and $q_{n,n+1}$.

```
# Determination of upstream/downstream cells in each direction
# If cell is at boundary, the pressure beyond the boundary is set to
    be 0.0
prev_pres_results = pres_results[:-1]
next_pres_results = pres_results[1:]
prev_pres_results.insert(0,0.0)
next_pres_results.append(0.0)

# Calculate interblock flowrate in + direction(s)
flow_x_results = []
zip_results_f = zip(tranx_results, pres_results, prev_pres_results,
    next_pres_results, krw_results, kro_results, muw_results,
    muo_results)
for (tranx, pres, prev_pres, next_pres, krw, kro, muw, muo) in
    zip_results_f:
    if next_pres > 0.0 and tranx > 0.0:
        flow_x = tranx * ((krw/muw)+(kro/muo)) * (pres-next_pres)
    else:
        flow_x = 0.0

    flow_x_results.append(flow_x)

prevx_flow_results = flow_x_results[:-1]
prevx_flow_results.insert(0,0.0)

# Calculate energy dissipation and internal energy change
e_dis = []
e_int = []
zip_results_e = zip(pres_results, prev_pres_results,
    next_pres_results, flow_x_results, prevx_flow_results)
for (pres, prev_pres, next_pres, flow_x, prevx_flow) in zip_results_e
    :
    ed = flow_x * (pres - next_pres)
    ei = pres * (prevx_flow - flow_x)

    e_dis.append(ed*(1e5/86400)) # Conversion factor
    e_int.append(ei*(1e5/86400))
```

6. **Calculate energy changes in well bottomhole**. The energy changes in well bottomhole include three components: external energy change due to inflow/outflow in regards to the reservoir system, internal energy change due to inflow/outflow to well blocks, and energy dissipation due to pressure drop when fluid flows from BHP to reservoir pressure. Formation volume factors for each phase are required here because well production and injection rates are given at standard conditions. OPM Flow version 2019.10 was unable to output FVF for each phase, but it is now possible through the updated version 2020.04 by using several command line options as listed by the OPM manual (Baxendale, 2020).

```
for wel in well_list:
```

```
        pcel = case.active_cell_property('DYNAMIC_NATIVE', 'PRESSURE', tstep)
        pwel = pcel[wel.idx] # Pressure at block connected to well
        bo = case.active_cell_property('DYNAMIC_NATIVE', 'BO', tstep)
        bw = case.active_cell_property('DYNAMIC_NATIVE', 'BW', tstep)
        bo_wel = bo[wel.idx] # Oil FVF
        bw_wel = bw[wel.idx] # Water FVF
        if wel.type == "PROD":
            e_external = -bhp * (wpr*bw_wel+opr*bo_wel) * 1e5/86400 # External energy
            e_internal_well = -pwel * (wpr*bw_wel+opr*bo_wel) * 1e5/86400 # Internal energy in well
            e_dissipation_well = (wpr*bw_wel+opr*bo_wel) * (pwel-bhp) * 1e5/86400 # Dissipated energy in well
        elif wel.type == "INJE":
            e_external = bhp * wir*bw_wel * 1e5/86400
            e_internal_well = pwel * wir*bw_wel * 1e5/86400
            e_dissipation_well = wir*bw_wel * (bhp-pwel) * 1e5/86400
```

7. **Calculate energy balance**. Steps 2 to 6 are repeated for every timestep. Then, based on Equation 1.1, the energy balance of the system can be found after all results are calculated.

As can be inferred from the code listings in steps 5 and 6, all the energy calculation results have the dimension of energy per timeThis means what is calculated for each timestep is energy change rate, which in the metric system has the unit of Watt (W). The steps defined above can be further expanded for more complicated cases, such as two-dimensional or three-dimensional cases, or possibly adding gas as the third phase in the reservoir. The calculations are divided into three different Python scripts: one script (*energy_reservoir.py*) handles flowrate calculation and energy changes inside the reservoir blocks (step 5) and apply the results to ResInsight, another script (*energy_well.py*) calculates energy changes at well bottomholes (step 6), and the last script *energy_balance.py* fetches energy change results from the previous two scripts and process them to calculate and plot energy balance for the simulation model. Figure 3.2 presents the flowchart for energy calculation as discussed in this section.

Even though there are three scripts, only two of them are run. Before running the scripts, the simulation model to be calculated has to be open in ResInsight in the background. The script *energy_reservoir.py* is run first to store energy dissipation and internal energy change results in each grid block in the ResInsight project. Next, while keeping ResInsight open, *energy_balance.py* is run, and it will use functions defined in *energy_well.py* to calculate energy changes in well bottomhole. After the calculation is finished, the script uses the well energy results, fetches reservoir energy results from ResInsight, and calculate energy balance. Plots of energy balance and its components (external energy change, internal energy change, and energy dissipation) are displayed at the very end of the process.

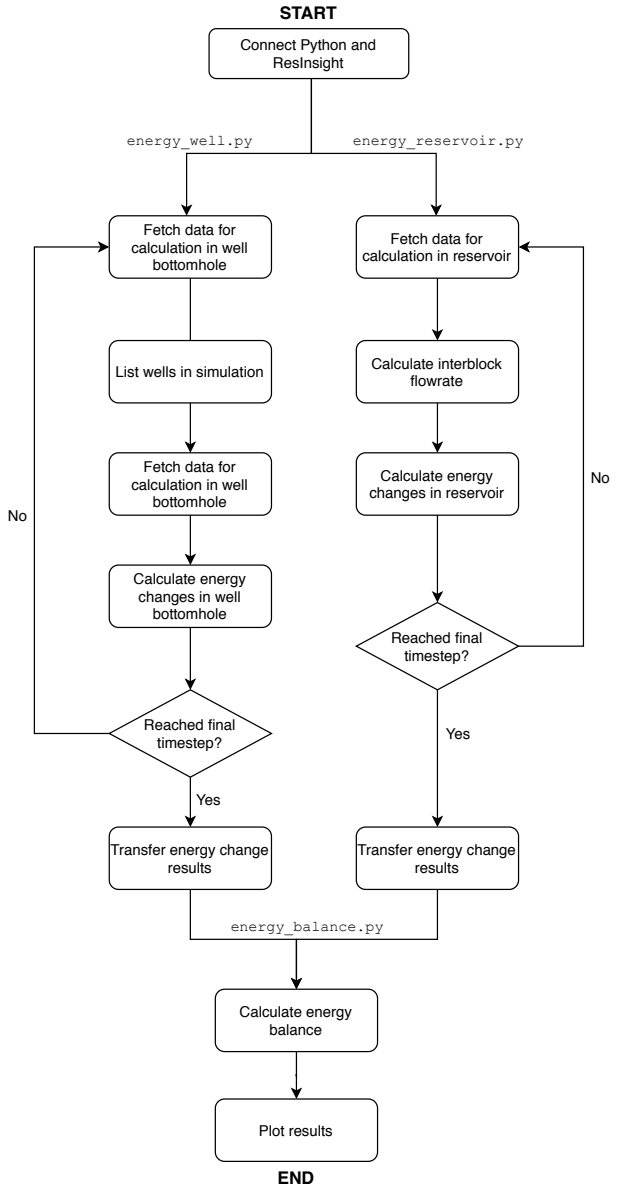


Figure 3.2: Flowchart of energy calculation development, showing how the computation process is divided into two Python scripts.

3.2 Cases for Calculation Testing

In this section, overview of the cases built for testing the calculation are presented. Starting with a very simple simulation, and eventually reach a 3D simulation with a few objects and

complications in the reservoir system. As the reservoir system becomes more complex, modifications to the energy calculation will be discussed. Input properties and commands for all of the test cases are built into a `.DATA` file. Important parameters and properties used in the input file are presented in Table 3.1. Relative permeability curve for the water-oil system in multiphase flow is presented in Figure 3.3.

Gridblock Parameters		
Dimension	2 x 2 x 2	m^3
Rock Properties		
ϕ	0.20	
k	100	mD
Water Properties		
ρ_w	1025	kg/m^3
B_w	1.01	rm^3/Sm^3
c_w	10^{-4}	1/bar
μ_w	1.0	cp
Oil Properties		
ρ_o	849	kg/m^3
B_o @ 100 bar	1.02	rm^3/Sm^3
c_o @ 100 bar	$2x10^{-5}$	1/bar
μ_o @ 100 bar	2.99	cp
Initialization		
S_w	0.2	(in two phase cases)
p_i	100.0	bar

Table 3.1: Grid parameters and reservoir properties for input in `.DATA` file, used to test energy calculation scripts.

3.2.1 One-dimensional cases

The first case consists only of a few blocks in one dimension. There are no wells, but each block has a different pressure to induce flow for some time. There is also just a single phase, water, in the reservoir. The purpose of this is to test energy calculations in the simplest manner possible. After enough time, pressure will eventually equilibrate in all grid blocks, and there are no more energy changes. Figure 3.4 shows the 1D case with no wells at the beginning of simulation. There are 10 cells for this case as shown in Figure 3.4. The timestep increment used is small, only 0.0004 days (approximately 35 seconds). The reservoir model is also small, and the equilibrating process would be too quick to observe if the timestep width is not narrow enough. In less than one day in simulation time, pressures on all cells are expected to be the same.

The next case adds two wells, one injector at the left grid block, and one producer at the right grid block. Well BHP is set to 200 bar for the injector and 100 bar for the producer. The reservoir size is also increased by adding more cells to allow for a longer simulation

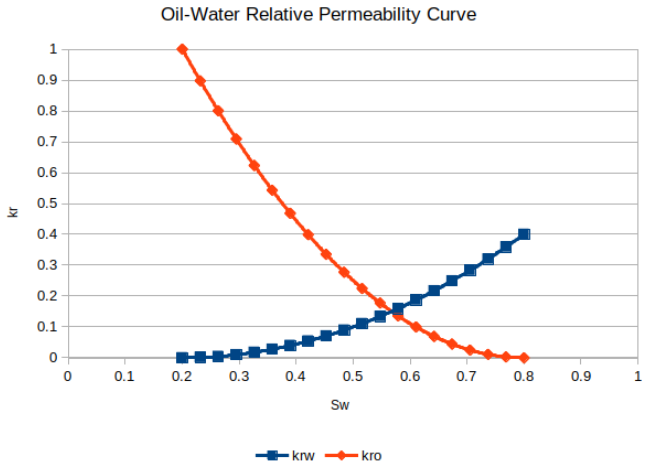


Figure 3.3: Oil-water relative permeability curve for the two-phase cases.

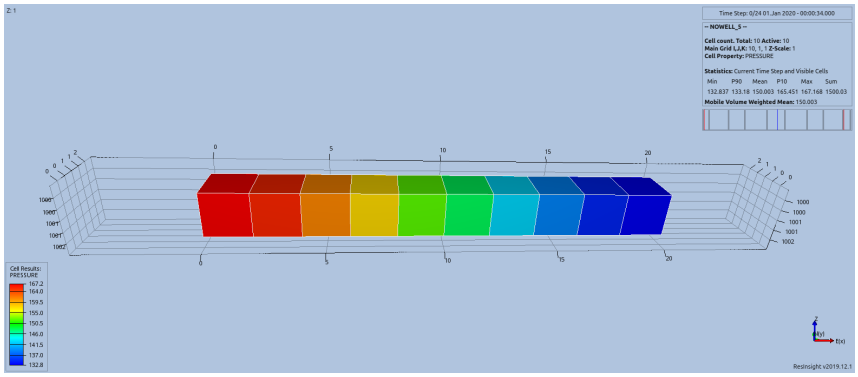


Figure 3.4: Simple 1D single-phase case without well. The property displayed in each cell is its pressure at the start of the simulation.

period. Each cell has the same pressure at simulation start, as defined in *Initialization* part in Table 3.1. There are 52 cells in this case, where two of them are associated with each well. Water is still the only fluid phase in the system. In conclusion, the new addition to this case is only external energy change by fluid in The timestep width is 0.05 days, and simulation is run for 50 timesteps. Figure 3.5 presents the reservoir model of this 1D, single-phase case with wells.

The last one-dimensional case for calculation testing adds oil to the system. Therefore, flow in the reservoir becomes two-phase and relative permeabilities need to be taken into account. Simulation is run using the same timestep size as before, 0.05 days, but now it is run until water reaches the production well to observe how water breakthrough affects energy change in the system. Figure 3.6 presents the reservoir model of this 1D, two-phase case.

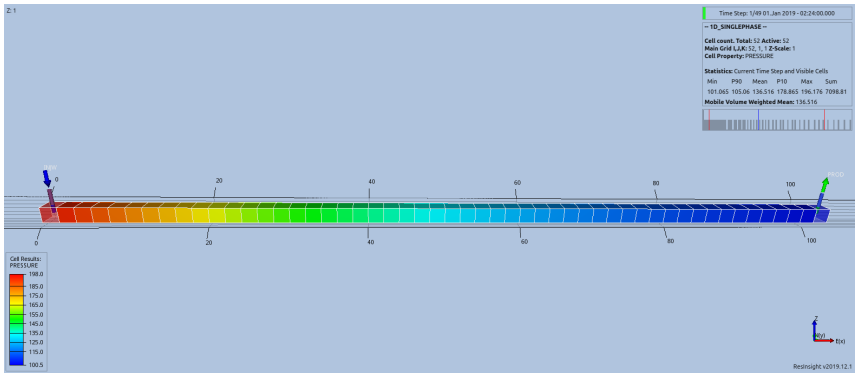


Figure 3.5: Simple 1D single-phase case with wells. The property displayed in each cell is its pressure nine (9) time step after the simulation starts.

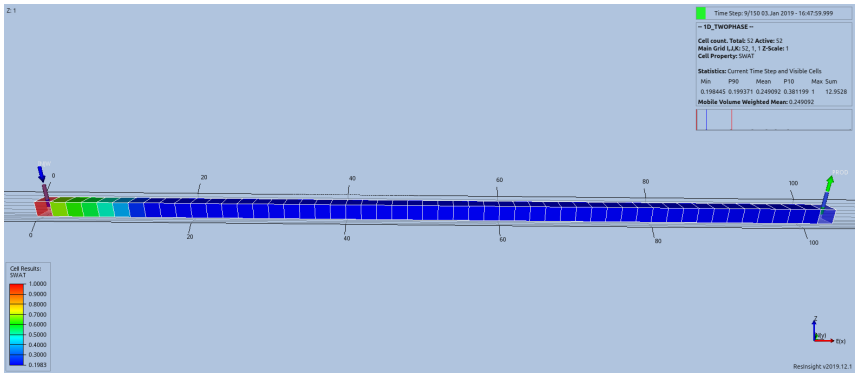


Figure 3.6: Simple 1D two-phase case with wells. The property displayed in each cell is its water saturation a moment after simulation starts.

3.2.2 Two-dimensional cases

Expanding the model to two dimensions requires some modifications in the calculation, especially for interblock dissipation and internal energy calculation. Fluids now flow in two directions, and each flow direction contributes to dissipation separately. Permeability in each direction is set to be the same, which is the value defined in Table 3.1. The number of grid blocks in this case is 2500 (50 in each direction). However, the block dimension and porosity remain unchanged. There are four wells in total: two injectors on the left side and two producers on the right side of the reservoir. The number of wells was added following the expansion of the model. The timestep size is 2.0 days, and the simulation is run until water breakthrough occurs in both production wells. Figure 3.7 shows the simulation model for this 2D case, displaying water that has swept more than halfway through the reservoir.

The next two-dimensional case adds some NNC (non-neighbor connection) pairs. NNCs are specified to set up transmissibility between a pair of non-adjacent grid blocks, usually

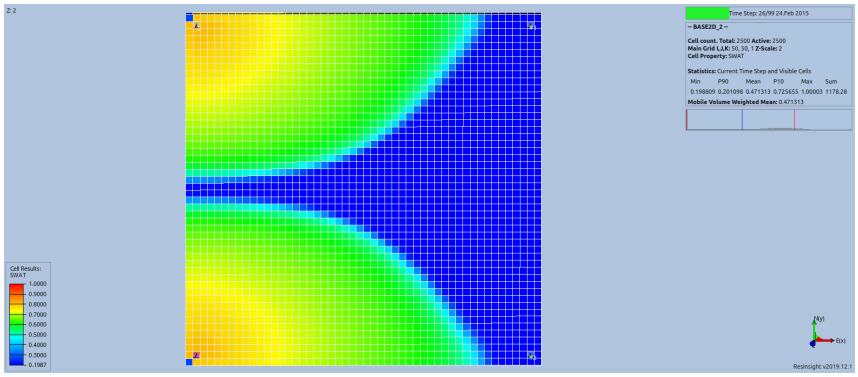


Figure 3.7: Simple 2D test case, showing water saturation in the middle of the simulation run.

used in the presence of faults or pinchouts that separates two grid blocks in simulation. However, in reality there is still flow between them. However, for this case NNCs are set up only to add complexities to the model. The effect of NNC on energy calculation is that there are additional dissipation and internal energy changes due to flow between NNC pairs. Figure 3.8 presents the simulation model for this 2D case, with NNC pairs visible as increased water saturation in a few cells in the middle of the model. The blocks with NNC pairs are located around the center part of the model.

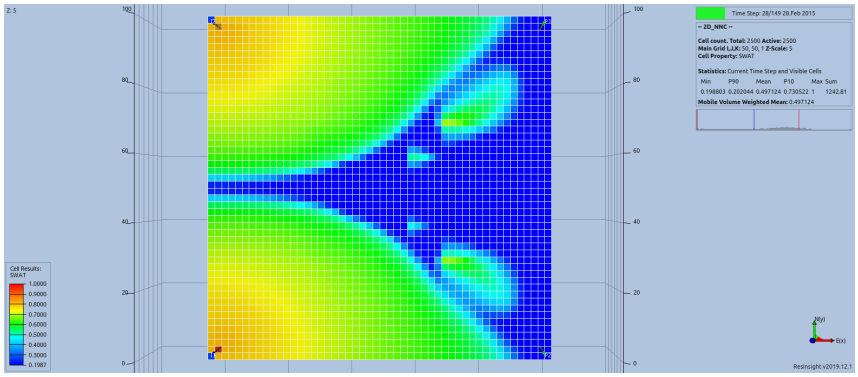


Figure 3.8: Simple 2D test case with NNC pairs, showing water saturation in the middle of simulation run.

3.2.3 Three-dimensional cases

Including the third dimension into the model once again require an extension in energy dissipation calculation, to determine dissipated energy in all three flow directions. There are three vertical layers in this case with each layer containing 50 x 50 cells, thus giving a total of 7500 grid blocks. Since permeability in the vertical direction is typically lower

than in horizontal direction, in this case vertical permeability is set to be 10 percent of horizontal permeability. The number of wells and their locations are similar to the 2D case, and the wells only penetrate the upper layer of the reservoir. There are no additional complexities, such as NNCs in the model. Figure 3.9 presents the simple 3D model displaying water saturation in each cell. The view is tilted to allow observation to each layer of the model.

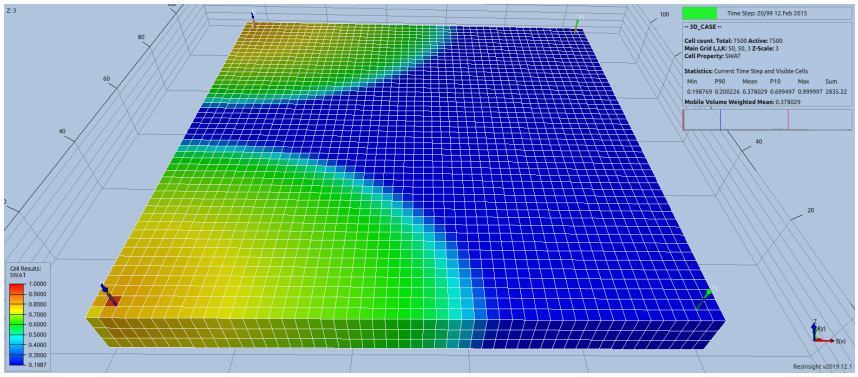


Figure 3.9: Simple 3D test case, showing water saturation in the middle of the simulation run.

Since there are multiple vertical layers in three-dimensional cases, an opportunity arises to complicate the wells with more connections to the model. All wells in the 3D cases will have two grid blocks connected with it, not only one connection like in the 1D and 2D cases. The consequence is that well inflow/outflow rates in Step 4 of Section 3.1 have to be redefined. Instead of directly using `WWPR`, `WOPR`, and `WWIR` that represents total well inflow/outflow rates for each phase, rates from each well connection is used. For water production, oil production, and water injection, the keywords are `CWPR`, `COPR`, and `CWIR` respectively.

The last test case aims to include as many features/complexities that are expected to be encountered in a real field model, but still simple enough to observe and understand. Taking the grid model from the previous 3D case, there are three additions for this case: NNCs, faults, and inactive cells. NNC has been featured and discussed in the previous subsection regarding two-dimensional cases, and there are no changes to its implementation here in a 3D case. Faults act as barriers that restrict some flow between two adjacent grid blocks. They are usually represented as a plane, affecting multiple cells simultaneously. Two faults are defined for this case: one in the north and one in the south. The northern fault is a straight plane going in the north-south direction, while the southern fault is a diagonal plane in the northeast-southwest direction. Figure 3.10 shows the faults' location in regards to the defined wells, while grid visibility is disabled.

Inactive cells are parts of the reservoir model that are not processed when simulating the model. Since for OPM Flow, the dimensionality of the grid has to be defined in a box

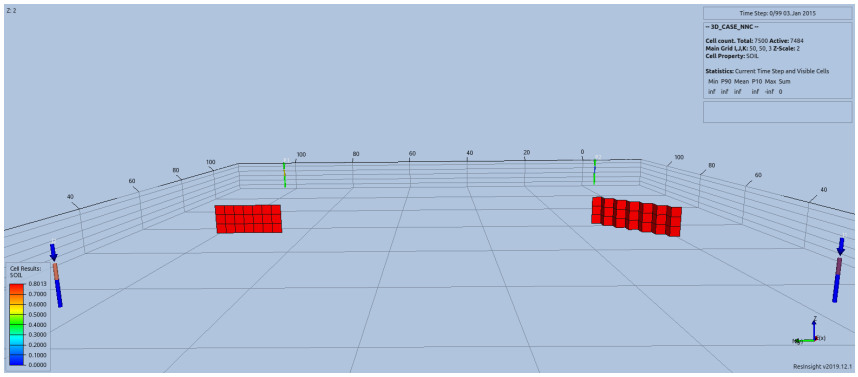


Figure 3.10: Location of defined faults and wells in the 3D test case with complexities. The view is towards the west direction, and grid block visibility is turned off.

shape, if in reality the reservoir is in another shape (which it usually is), then some parts of the box have to be disregarded, or deemed ”inactive.” In this case, a small chunk of the 3D box reservoir is set as inactive. Energy calculations developed earlier should be made sure not to process these grid blocks, as the simulator does not output any data to inactive cells. Figure 3.11 focuses on the inactive cells set at the top layer of the grid model, while Figure 3.12 presents the simple 3D case with additional complexities, showing water saturation.

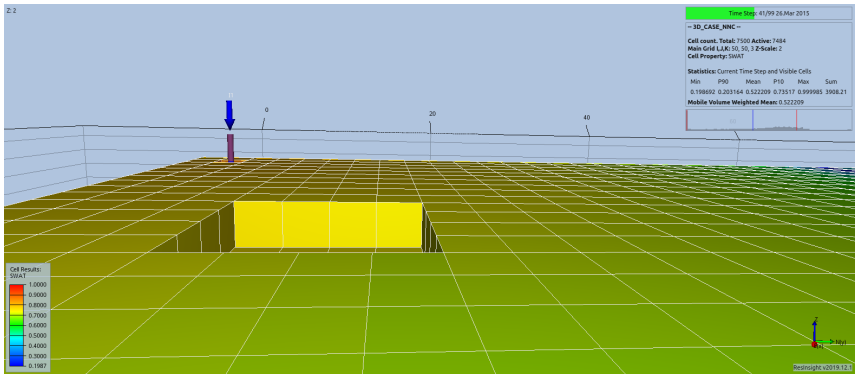


Figure 3.11: Inactive cells in a box reservoir model, presenting itself as a missing chunk of grid-blocks.

Figure 3.12 highlights some effects of these complexities. Flow from one NNC pair in the southeast part of the model almost reaches production well there. This is because flow reaches the NNC pair quicker in the south, as there are some inactive cells there that affect the flow direction. The shape of the faults also influences its ability to hinder flow. In this case, the flow direction is from west to east, so a straight north-south fault will have a higher impact on slowing fluid flow than a diagonal fault.

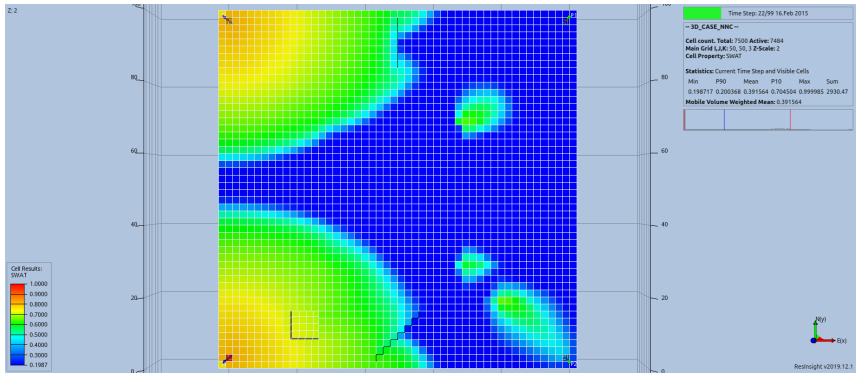


Figure 3.12: Simple 3D case with complexities, showing water saturation in the middle of the simulation run.

3.2.4 Summary of test cases

In total, there are seven (7) cases built to test energy calculations in this study: three 1D cases, two 2D cases, and two 3D cases. Table 3.2 summarizes all cases described in this section, showing progression of the reservoir from basic to complicated through the cases.

Case Name	Dimensions	Wells	Phases	NNCs	Faults	Inactive Cells
1D, single-phase, no wells	1	No	1	No	No	No
1D, single-phase, wells	1	Yes	1	No	No	No
1D, two-phase	1	Yes	2	No	No	No
2D	2	Yes	2	No	No	No
2D with NNCs	2	Yes	2	Yes	No	No
3D	3	Yes	2	No	No	No
3D with complexities	3	Yes	2	Yes	Yes	Yes

Table 3.2: Summary of cases defined to test energy calculations.

3.3 Norne Reservoir Model

After testing energy calculations with cases defined in Table 3.2, the next step is to apply the calculations to a real, full-field reservoir model. The Norne field is chosen as a subject of this study because its data and reservoir model are already made free for the public. The Norne case was taken from OPM's open datasets at https://github.com/OPM/opm-data.

The Norne field is an oil field located in the Norwegian sea, specifically in the blocks 6608/10 and 6508/10 in the Nordland II area, around 200 km west offshore of Sandnessjøen (Gjerstad et al., 1995). Figure 3.13 shows the location of Norne field. The discovery of the field took place in December 1991. Development wells drilling started in August 1996, and the first oil production was in November 1997. Per Norwegian Petroleum

Directorate (NPD) report in December 2019 (https://factpages.npd.no/en/field/pageview/all/43778), 91.17 million Sm^3 of oil and 8.16 billion Sm^3 of gas have been recovered from Norne. There are still 3.85 million Sm^3 of recoverable oil and 8.16 billion Sm^3 of recoverable gas remaining in the reservoir. There are two main structures of the field: the main structure and the northeast segment (Rwechungura et al., 2010). The hydrocarbon column consists of 110 m oil and 25 m gas, for a total length of 135 m.

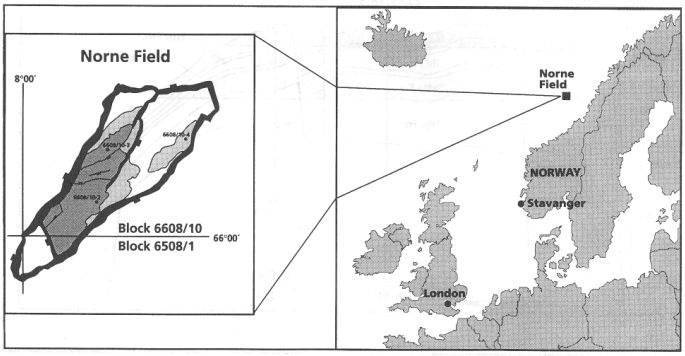


Figure 3.13: Location of the Norne field in the Norwegian sea. Taken from Gjerstad et al. (1995).

Norne is divided into four formations from top to bottom: Garn, Ile, Tofte and Tilje. A shale zone Not is located between Garn and Ile. The reservoir rock mostly consists of fine-grained sandstones and buried at the depth of around 2500-2700 m. Porosity of the rock is around between 25 and 30 percent, while permeability ranges from 20 and up to 2500 mD. The four formations plus Not are segmented further into multiple reservoir zones in the Norne geological model. Geological zonation of Norne field is presented as Table 3.3.

The reservoir simulation model consists of 22 layers based on the geological zonation defined in Table 3.3. Layers #1 to #3 represent Garn, layer #4 represents Not, layers #5 to #11 represents Ile, layers #12 to #18 represents Tofte, and layers #19 to #22 represents Tilje. In total there are 113344 grid blocks in the reservoir model, of which 44431 are active. 36 wells are defined in the model, some of them are fixed (either producer or injector), but some other alternates between producing and injecting fluid throughout the simulation run. Simulation of the reservoir is run from November 1997 to December 2006, divided into 248 time steps. Figure 3.14 shows the entire Norne reservoir model looking northwest, showing oil saturation which dominates the middle layers of the main structure and upper layers of the northeast segment.

Garn 3 **Garn 2** **Garn 1**		
Not 3	Upper Not Shale	
Not 2	Not 2.3 Not 2.2 Not 2.1	Norne Sandstone
Not 1	Lower Not Shale	
Ile 2 **Ile 1**	Ile 2.2 Ile 2.1 Ile 1.3 Ile 1.2 Ile 1.1	Ile 2.2.2 Ile 2.2.1
Tofte 2	Tofte 2.2 Tofte 2.1	
Tofte 1	Tofte 1.2 Tofte 1.1	
Tilje 4 **Tilje 3** **Tilje 2** **Tilje 1**		

Table 3.3: Geological zonation of the Norne field. Modified from Rwechungura et al. (2010).

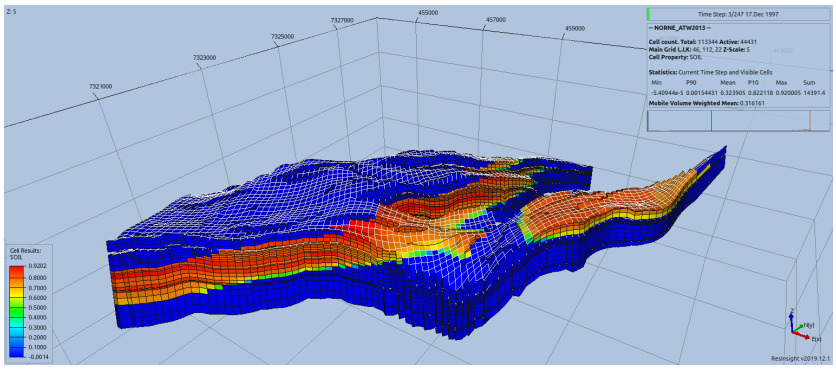


Figure 3.14: The Norne reservoir model, showing oil saturation in all cells.

Chapter 4

Results and Discussion

In this chapter, the results of energy calculations in the test cases and the Norne reservoir model are presented. The calculations are developed using the steps described in Section 3.1. The calculation results are then discussed to provide the insights required to achieve the objectives of this study.

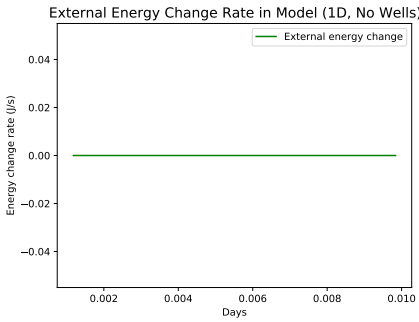
4.1 Energy Calculation Results on the Test Cases

The test cases have been defined at Section 3.2 and summarized at Table 3.2. Starting from one-dimensional cases, two-dimensional cases, and finally three-dimensional cases, the results discussed here will include plots of components in energy balance, and the discussion will concern how the reservoir recovery process in each case affects external energy changes, internal energy changes, and energy dissipation.

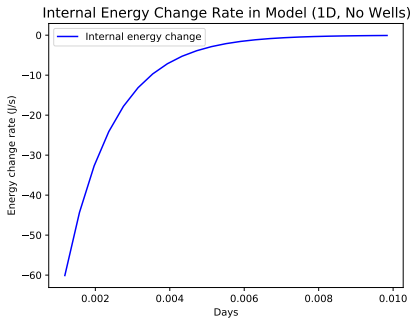
4.1.1 One-dimensional cases

1D case, single-phase, no wells
The first 1D case as pictured in Figure 3.4 has no wells and only a few blocks arranged in a straight line. Initially, a pressure gradient exists from the left block to the right block and simulation is run until the pressures in all blocks equilibriate. Figure 4.1 displays the three energy change rates throughout the simulation of this case. There are no energy from external systems as there are no wells in this case, and as a result external energy change is zero throughout the simulation. The dissipation rate was initially high, however it quickly faded out as pressures in each gridblock converged towards a single value with no more flow in the system. Since there are no external source of energy, then the energy needed to flow fluids in this case had to come from the internal energy of the system. This is evident by seeing how internal energy change rate is negative throughout the simulation, and eventually becomes zero. and in Negative internal energy change mean the system has to release some of its stored energy to drive flow between grid blocks. In absence of external energy, the dissipation rate and internal energy change rate are inverse of each other and

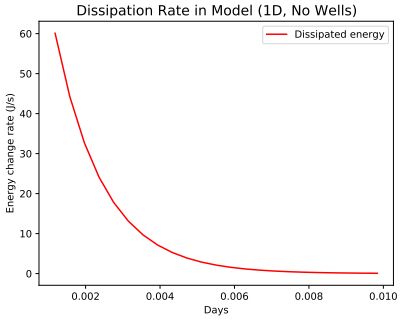
they balance out. Figure 4.2 shows how the energy balance (in black solid line) is achieved for this 1D single-phase case without wells. As can be seen in both figures, the process to reach equilibrium takes around 0.01 days (approximately 14 minutes). This relatively short time is related to the model length which is only 20 meters long.



(a) External energy change rate.



(b) Internal energy change rate.



(c) Energy dissipation rate.

Figure 4.1: External energy, internal energy, and dissipation rate for 1D single-phase case without wells.

1D case, single-phase, wells

The next case to observe is the 1D single-phase case with wells. This case has a same initial pressure gradient as before (2 bar/m), but now there are 50 grid blocks, making the total length of the model 100 m. Additionally, there are wells at each end of the model that provide energy transfer from/to external systems. As can be seen in Figure 4.3, at the beginning of the simulation, all energy change rates are high. This large rate is caused by initial pressure difference between injection well bottomhole and the reservoir which was set at 100 bar. After the simulation carried on for some time, average reservoir pressure increased while well BHP is kept constant. Eventually, average reservoir pressure reached a constant value (Figure 4.4), and this caused fluid flow to become steady-state, meaning there are no more pressure changes throughout the reservoir.

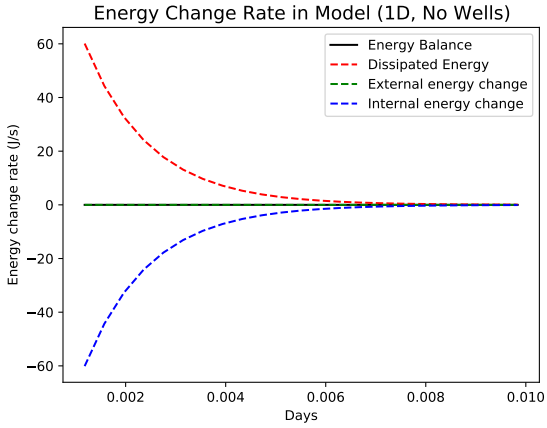
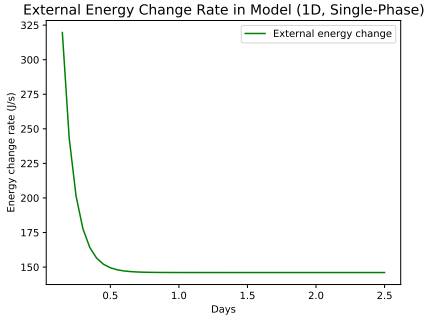
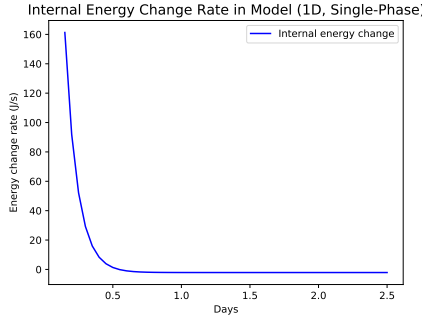


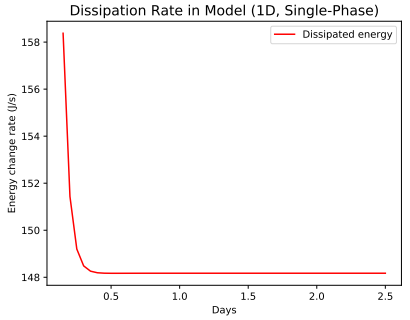
Figure 4.2: Energy balance for 1D single-phase case without wells.



(a) External energy change rate.



(b) Internal energy change rate.



(c) Energy dissipation rate.

Figure 4.3: External energy, internal energy, and dissipation rate for 1D single-phase case with wells.

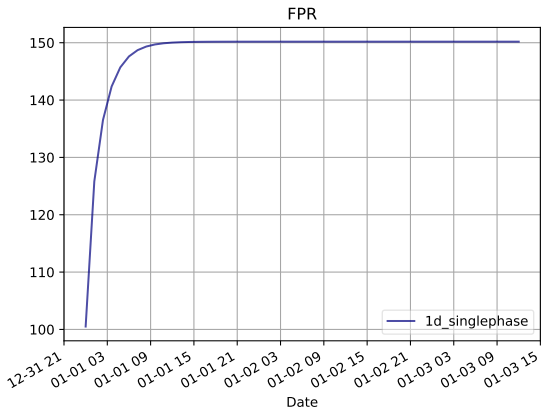
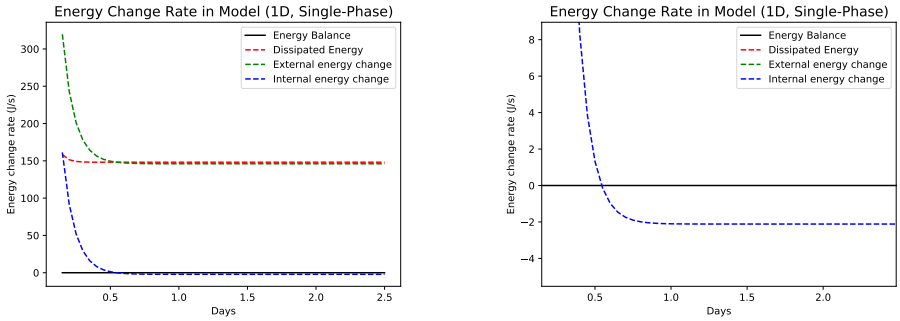


Figure 4.4: Reservoir average pressure (FPR) for 1D single-phase case with wells.

At steady-state condition, internal energy within the reservoir no longer changes and is permanently constant until there are other disturbances added to the system. Meanwhile, external energy change and dissipation eventually reach a constant rate with almost equal values. This indicates that, during steady-state flow, most of the energy required to flow fluids originated from external energy sources, which is dissipated afterwards. Figure 4.5 shows the energy balance for this 1D single-phase case with wells, along with a zoomed in version that highlights that balance is achieved. However, the actual internal energy change rate is not zero, but slightly lower than zero that implies the system still lose some internal energy during steady-state flow. The explanation for this is the fact that the volume of water during flow from the injection to the production well is not constant. As discussed in Figure 2.2, water has some compressibility that cause volume changes when pressure is changed. As water flows to a lower pressure, it undergoes volume expansion which releases some energy. Thus, negative internal energy change shown in Figure 4.5 is caused by fluid expansion during transport in the reservoir system.

1D case, two-phase

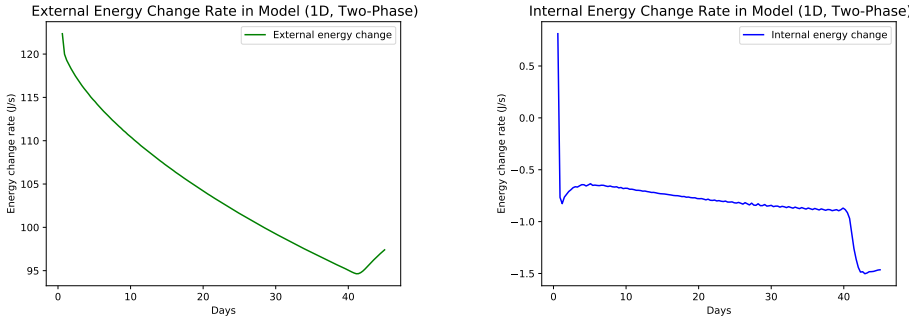
The last 1D case to present and discuss is the 1D case with two fluid phases: oil and water. As can be seen in Figure 4.6, the whole simulation for this case is run for a significantly longer time compared to the previous case, even though the model length and pressure gradient are the same. The reason for this is that the simulation was run until water breakthrough occurs, in order to see how it would affect energy changes in the system. The water saturation front travels slower than the pressure front throughout the reservoir. External energy and dissipation rate look similar to each other, displaying rate decrease as pressure builds within the reservoir and pressure drop from well to reservoir decreases. The internal energy change rate slowly becomes more negative, as water displaces oil in the system. Referring back to Table 3.1, it can be found that water compressibility for the test cases are higher than oil. Thus, when there are more water in the system, energy required to expand the fluid during flow (which comes from internal energy) is higher than



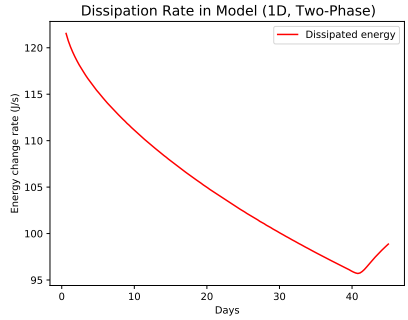
(a) Energy balance. **(b)** Energy balance (zoomed in).

Figure 4.5: Energy balance plots for 1D single-phase case with wells.

the beginning of the simulation when the oil phase dominates the pore space of the reservoir.



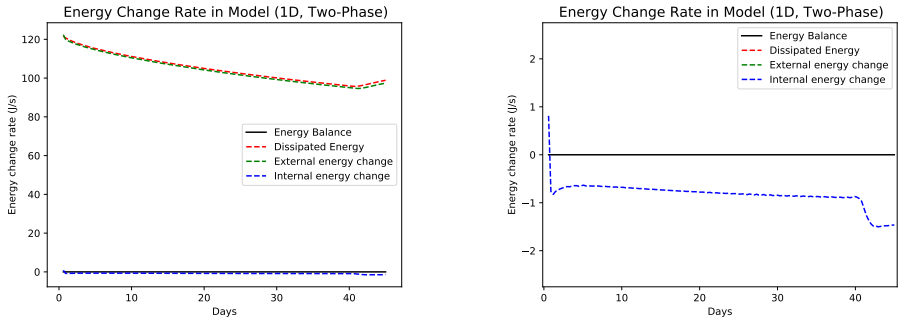
(a) External energy change rate. **(b)** Internal energy change rate.



(c) Energy dissipation rate.

Figure 4.6: External energy, internal energy, and dissipation rate for 1D two-phase case.

After water breakthrough, there are some noticeable effects on the energy change rates as seen in Figure 4.6. This is mainly caused by the changes in producer well. Before water breakthrough, the well only produces oil, but after saturation front reached production well, water is also produced alongside oil. At this time of the simulation, pressure changes in each block is already stabilized and it could be said that the pressure drop is fairly constant between the production well bottomhole and the grid block connected with it. As water began entering the production well, more energy is required to push it into the well because it expands more than oil given the same pressure drop. Thus, more internal energy is spent on expansion and more energy is dissipated to produce water. In turn, the external energy change rate also rises to balance the whole process. Figure 4.7 shows the energy balance plot alongside the zoomed in version for this 1D two-phase case, confirming that energy balance is also achieved for this case. Comparing Figure 4.7 with Figure 4.5, one could notice that the total dissipation rate in the reservoir is generally lower for the two-phase case. Recall Figure 3.3 that shows oil-water relative permeability curves for the test cases. At all S_w values above initial water saturation, total relative permeability is smaller than 1.0, thus total flow rate of oil and water is always lower compared to the single-phase case where there is only water. As defined by Equation 2.20, then it can be concluded that there is less energy dissipation in the two-phase case because of lower flow rates.



(a) Energy balance.

(b) Energy balance (zoomed in).

Figure 4.7: Energy balance plots for 1D two-phase case.

It is also interesting to see how the dissipation is visualized in the reservoir blocks. Figure 4.8 is a snapshot of energy dissipation distribution in the middle of simulation. It can be observed that energy dissipation is highest at the middle, which is where the saturation front is at this time of the simulation. The saturation front is the part in which biggest pressure drop occurs, and in turn this high pressure difference creates the highest dissipation rate compared to other parts of the reservoir.

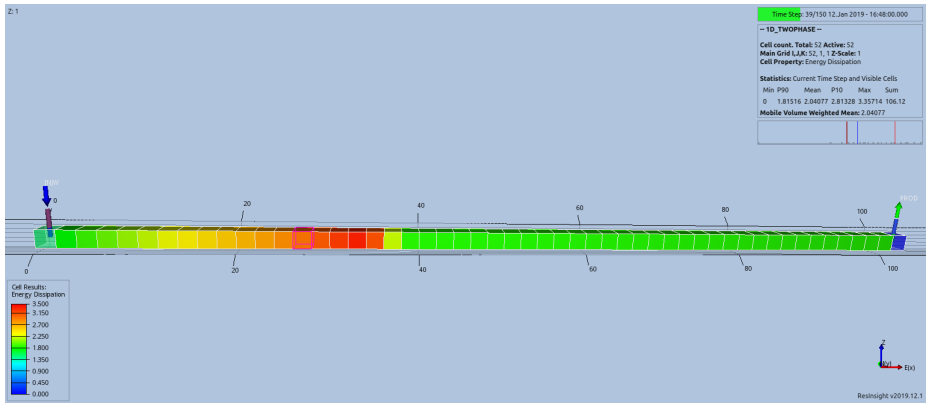


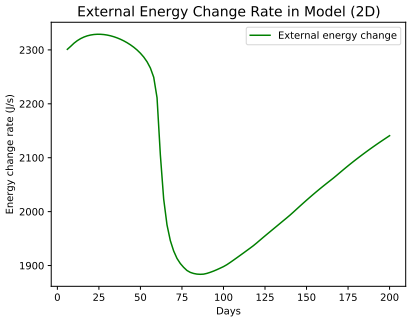
Figure 4.8: Visualization of dissipation rate in each reservoir block for 1D two-phase case.

4.1.2 Two-dimensional cases

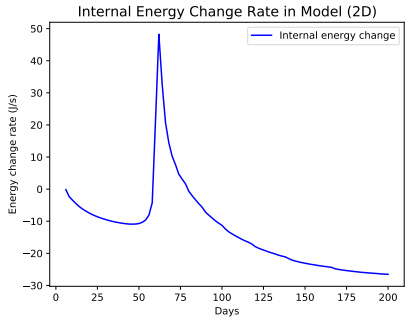
2D case
The first two-dimensional case as defined in Table 3.2 introduces flow in different directions, and total energy change in reservoir is calculated as a sum of energy changes for each flow direction. Figure 4.9 displays energy change rates for external energy, internal energy, and dissipation for the 2D case. Before water breakthrough, external energy and dissipation rate slowly increases, which is the opposite of what happened in Figure 4.6. More energy spent by the system could be attributed to the fact that the model is now two-dimensional, and flow has to be carried out and distributed to more grid blocks than in the 1D cases. Meanwhile, the internal energy change rate become more negative as water, which has higher compressibility, gradually displaces oil in the reservoir.

Directly after water breakthrough, it is observed that external energy and dissipation rate drops quite significantly, decreasing by around 20%. Examining the model, it was found that this was caused by a significant increase in average reservoir pressure shortly after the saturation front has reached both production wells. This occurred at the same time as when external energy and dissipation rate falls, which is around 60-70 days into the simulation, as seen in in Figure 4.10. The sudden increase of average reservoir pressure was distributed throughout the model, and made the interblock pressure difference generally lower than before water breakthrough. Thus, this created lower interblock dissipation across the reservoir. Figure 4.11 shows the pressure distribution before and after water breakthrough. Notice that after breakthrough the pressure is distributed more evenly, with the only significant pressure drop taking place in the middle. Internal energy change experienced a sharp spike a moment after water breakthrough, caused by the sudden pressure increase discussed earlier. Pressure increase in a grid block would compress the constituting fluid within, and as defined by Equation 2.2, by convention compressing a fluid increases its internal energy.

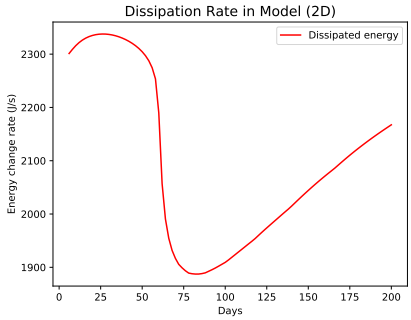
At late simulation times, water continues to sweep more areas in the model and water



(a) External energy change rate.



(b) Internal energy change rate.



(c) Energy dissipation rate.

Figure 4.9: External energy, internal energy, and dissipation rate for 2D case.

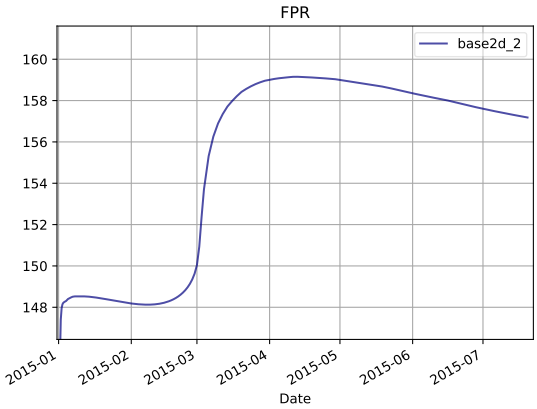
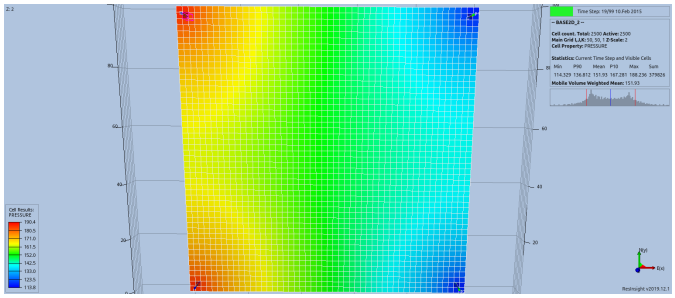
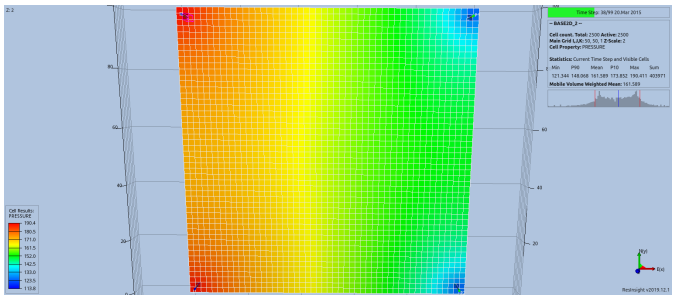


Figure 4.10: Reservoir average pressure (FPR) for 2D case.



(a) Before water breakthrough.



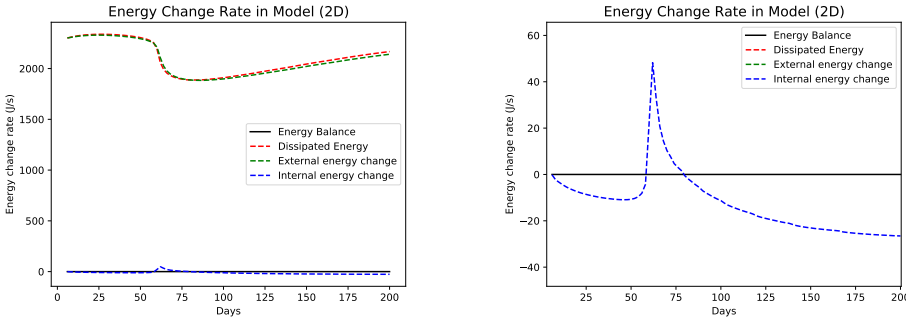
(b) After water breakthrough.

Figure 4.11: Pressure distribution in 2D case before and after water breakthrough.

cut (ratio of produced water to total produced fluid) in the production wells keep increasing. However, unlike in 1D two-phase case, dissipation and external energy rates increases as more water is present in the system. This is caused by gradually decreasing reservoir pressure after the sudden pressure spike (see again Figure 4.10). The decreasing reservoir pressure will result in higher water injection rate, directly affecting external energy change and dissipation rates. Internal energy change grows more negative as water become more dominant in the reservoir, again caused by more internal energy being consumed to expand water across the model. To summarize this case, Figure 4.12 shows the energy balance plots for the 2D case.

2D case with NNCs

The second 2D case features non-neighbor connections (NNCs), and the energy change rates are provided by Figure 4.13. Looking at the plots and comparing them to those at Figure 4.9, one could examine that the shape of the plots are similar, which means that the recovery process between the two 2D cases are comparable to each other. However, a closer look will give some other insights: external energy and dissipation rates are higher compared to 2D case without NNCs, while internal energy rate remains mostly the same. The higher external energy and dissipation is caused by extra interblock connections that are defined in this case. Instead of having only two flow directions along the x and y axis, some blocks have a third connection with other blocks that are not logically connected to them (hence the name *non-neighbor* connections), but could be physically connected.



(a) Energy balance.

(b) Energy balance (zoomed in).

Figure 4.12: Energy balance plots for 2D case.

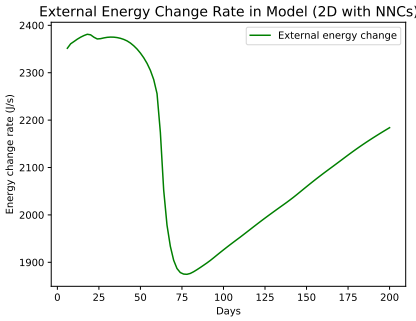
Another point of interest in the plots is how energy change decreases a bit around 20 days into the simulation. This is the effect of the first flow of water through the NNC pairs. The blocks set to have NNCs are located around the middle of the model (see again Figure 3.8). The first flow of water through the NNCs require less energy than through adjacent interblock flows, because transmissibility between NNC pairs is set to be lower than regular interblock transmissibility. This results in lower energy required to first flow water by means of non-neighbor connections.

In conclusion, adding NNCs to the 2D case only affects how much energy is being supplied and dissipated to allow additional flow between the NNCs, and internal energy change rate is not affected. In this case, there are only four (4) pairs of NNCs defined, and the connections have relatively low transmissibilities. Adding more NNC pairs could further amplify the amount of energy required by the model to flow under a same condition. The energy balance plots for this case is given as Figure 4.14.

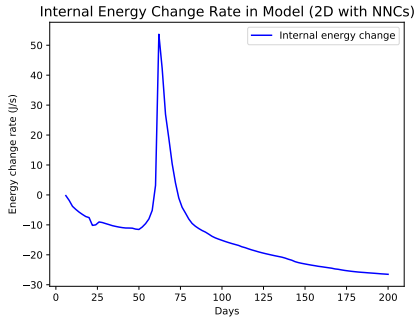
4.1.3 Three-dimensional cases

3D case

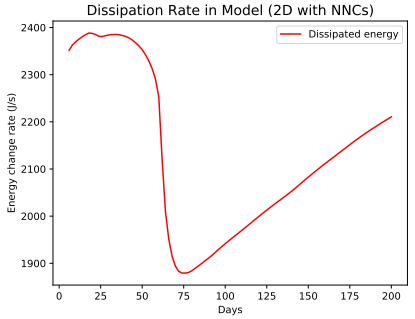
The first 3D case was simply built by extending the 2D case (without NNCs) to include three layers vertically. As a result, processes that occurred during recovery are mostly the same between the two cases. As a result, energy change rates as shown in Figure 4.15 appear very similar to those in Figure 4.9, albeit having slightly higher change rate values for dissipation and external energy. These higher values come from the fact that additional energy is required to flow in the z-direction, but the additional energy needed is relatively small because of how the vertical permeability is defined for this case: as stated in Subsection 3.2.3, vertical permeability is set to be only 10% of horizontal permeability, and as a result vertical transmissibility will be 10% of horizontal transmissibility under the same conditions. Thus, flow in the vertical (z) direction will dissipate less energy compared to flow in horizontal (x and y) directions.



(a) External energy change rate.

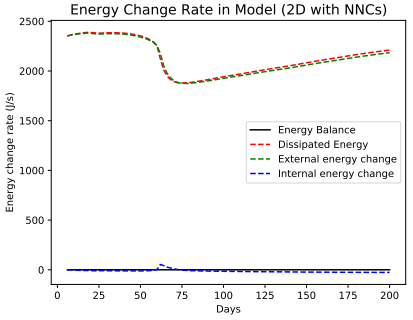


(b) Internal energy change rate.

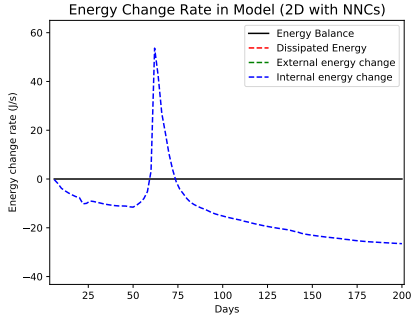


(c) Energy dissipation rate.

Figure 4.13: External energy, internal energy, and dissipation rate for 2D case with NNCs.



(a) Energy balance.

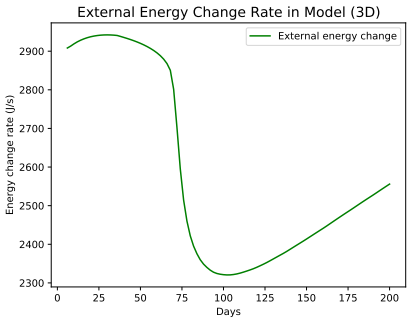


(b) Energy balance (zoomed in).

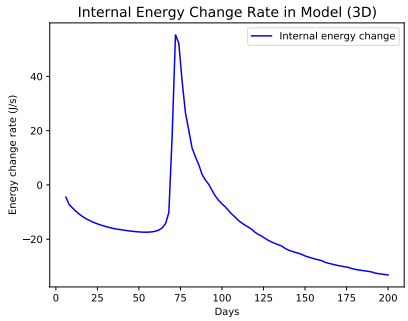
Figure 4.14: Energy balance plots for 2D case with NNCs.

Interestingly, internal energy change in this 3D case is similar to the 2D cases. This signifies that there is a similar amount of fluid expansion in these cases that resulted from flowing fluid from higher to lower pressure points. Even though there are more water in-

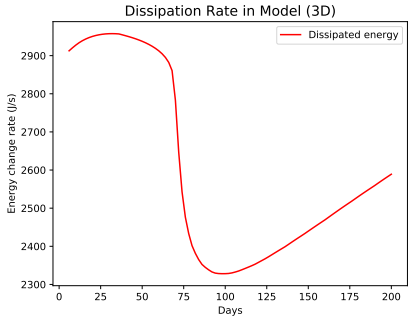
jected and fluids produced per day in this 3D case, the volume difference between injected and produced fluids are the same because of similar flow parameters among the 2D models and this 3D model.



(a) External energy change rate.



(b) Internal energy change rate.



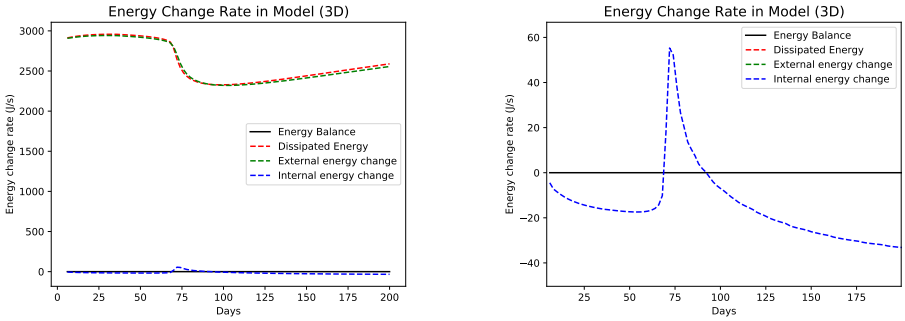
(c) Energy dissipation rate.

Figure 4.15: External energy, internal energy, and dissipation rate for 3D case.

Plots of energy balance in this 3D case is shown as Figure 4.16. As expected, the plots look similar to Figure 4.12, only with slightly higher dissipation and external energy change rates.

3D case with complexities

The final test case is a 3D model that features additional complexities, including NNC pairs, faults, and inactive cells to better resemble real field models like Norne. Calculation of individual interblock flow gets rather difficult in this case, because of how ECL-based simulators index each grid block. The restart file (`.UNRST`) only lists active cells and completely ignore all inactive cells. Meanwhile, the grid model file (`.EGRID`) provides indices for all cells, both active and inactive. This creates indexing mismatch when trying to fetch data the from restart file and matching it with a grid block in the full grid model. As a result, `.EGRID` index of each gridblock has to be matched one by one to `.UNRST` index. Then, pressure results from the restart file is matched by index, also one by one, to



(a) Energy balance.

(b) Energy balance (zoomed in).

Figure 4.16: Energy balance plots for 3D case.

the grid file index. Finally, after flow is calculated, the flow results is also matched again to the grid file index.

```
# Read ACTNUM numbers from EGRID file
actnum = egrid_file["ACTNUM"][0]
active_cells = []
for i in range(len(actnum)):
    if actnum[i] == 1:
        active_cells.append(i+1)
num_active_cells = len(active_cells)
```

```
# Index match for pressure results
for c in range(num_active_cells):
        print("pressure index matching: checking cell", c+1, "of",
    num_active_cells, end="\r")
        idx = active_cells[c]
        plane_num = idx%(Nx*Ny)
        layer_num = math.floor(idx/(Nx*Ny))
        if idx-1 in active_cells and idx%Nx != 0:
            prevx_pres_results[c] = pres_results[active_cells.index(idx-1)
    ]

        if idx+1 in active_cells and (idx+1)%Nx != 0:
            nextx_pres_results[c] = pres_results[active_cells.index(idx+1)
    ]

        if idx-Nx in active_cells:
            prevy_pres_results[c] = pres_results[active_cells.index(idx-Nx
    )]
            if plane_num < Nx:
                prevy_pres_results[c] = 0.0

        if idx+Nx in active_cells:
            nexty_pres_results[c] = pres_results[active_cells.index(idx+Nx
    )]
            if plane_num+Nx >= (Nx*Ny):
                nexty_pres_results[c] = 0.0

```

```
        if idx-(Nx*Ny) in active_cells:
            prevz_pres_results[c] = pres_results[active_cells.index(idx-Nx*Ny)]
            if layer_num == 0:
                prevz_pres_results[c] = 0.0

        if idx+(Nx*Ny) in active_cells:
            nextz_pres_results[c] = pres_results[active_cells.index(idx+Nx*Ny)]
            if layer_num == Nz-1:
                nextz_pres_results[c] = 0.0
    print(end='\r')
```

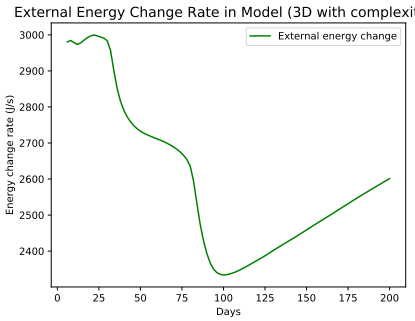
```
# Index match for flow results
for c in range(num_active_cells):
        print("flow index matching: checking cell", c+1, "of", num_active_cells, end="\r")
        idx = active_cells[c]
        plane_num = idx%(Nx*Ny)
        layer_num = math.floor(idx/(Nx*Ny))
        if idx-1 in active_cells and idx%Nx != 0:
            prevx_flow_results[c] = flow_x_results[active_cells.index(idx-1)]

        if idx-Nx in active_cells:
            prevy_flow_results[c] = flow_y_results[active_cells.index(idx-Nx)]
            if plane_num < Nx:
                prevy_flow_results[c] = 0.0

        if idx-(Nx*Ny) in active_cells:
            prevz_flow_results[c] = flow_z_results[active_cells.index(idx-Nx*Ny)]
            if layer_num == 0:
                prevz_flow_results[c] = 0.0
    print(end='\r')
```

The whole index matching mechanism is rigorous and takes the most of computational time required in the whole energy calculation process.

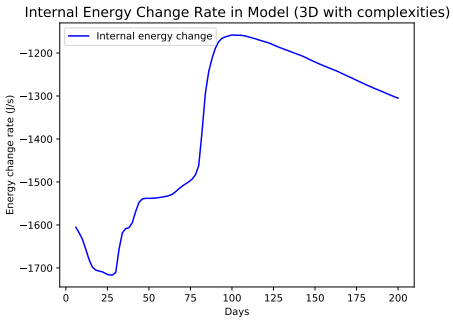
Energy change rate plots for the 3D case with complexities are displayed as subfigures in Figure 4.17. Compared to plots in Figure 4.15, there is a significant increase in dissipation rate and also a much more negative internal energy change. The main reason for these phenomena is the existence of faults in the middle part of this model (see Figure 3.10). The transmissibility multiplier through the faults were designated a value of 0.005. This means flow through the faults would be greatly hindered, reducing the flow rate to be only 0.5% of its value without faults. However, because the faults do not completely block the whole vertical plane of the model, the flowing fluids could find a way around these faults and merge with other fluid streamlines, creating higher pressure drops near the edges of the faults. Thus, there would be higher energy dissipation in these areas. Figure 4.18 displays the ResInsight visualization of the northern area of the reservoir model. The left and right ends of the model have high dissipation rate due to presence of wells, while some blocks in the middle dissipate a lot of energy because they are located in the edges

of a fault.

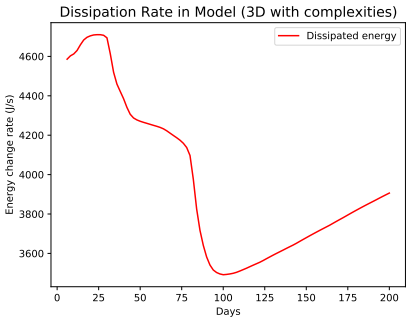
The same effect of water breakthrough in production wells are manifested in the same way as in previous 2D and 3D cases. Dissipation and external energy rates drop quite sharply at first water breakthrough, and rises slowly afterwards. Meanwhile, internal energy rates became less negative at first, but turns to be more negative as water cut increases.



(a) External energy change rate.



(b) Internal energy change rate.



(c) Energy dissipation rate.

Figure 4.17: External energy, internal energy, and dissipation rate for 3D case with complexities.

Energy balance plot for this 3D case featuring NNCs, faults, and inactive cells is shown in Figure 4.19. It can be observed from the plot that dissipation rate is more dominant in this case due to faults, while external energy required to inject and produce fluids is not significantly different to the simple 3D case (Figure 4.16). Internal energy stored within the system is spent more compared to previous cases, in order to let fluids flow through and around the two faults.

Effects of inactive cells are not accentuated in any meaningful way in the energy balance plot, but its purpose of improving energy calculation development is fulfilled. The fact that energy balance is achieved for this case, and there are no inconsistencies compared to other cases, mean that the energy calculation used for this 3D case with complexities is

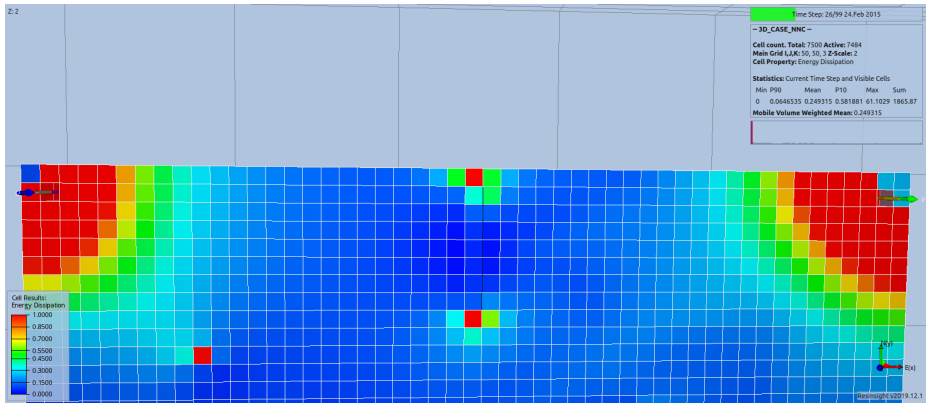


Figure 4.18: Visualization of dissipation rate near a fault (top middle part of the model) in the 3D case with complexities.

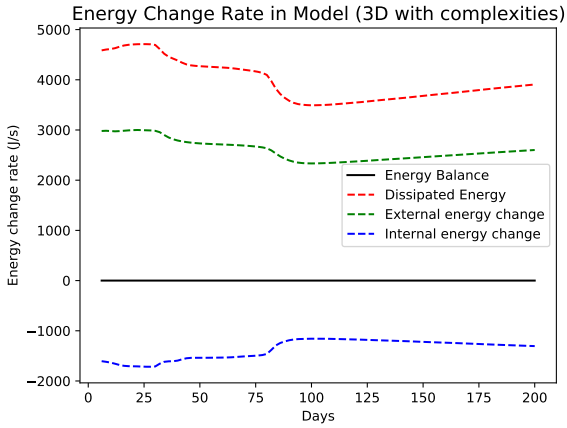


Figure 4.19: Energy balance for the 3D case with complexities.

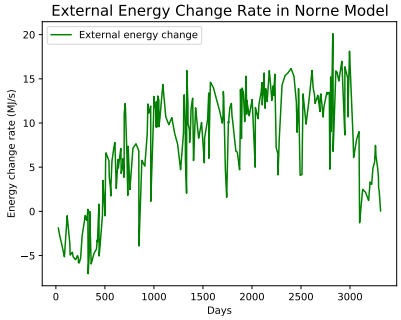
valid.

4.2 Energy Calculation Results on the Norne Model

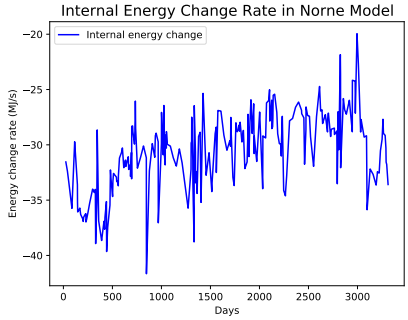
After testing the developed energy calculations against synthetic reservoir models in Section 4.1 and proving that the calculations yield valid results, they are then applied to the Norne reservoir model. This full-field model is simulated for 248 time steps, starting from 6 November 1997 and ending on 1 December 2006 for a total of 3312 simulation days. The 36 defined wells are activated gradually: at the beginning of the simulation, only three (3) wells are active, while the last well is opened just two months before the end of the simulation. All reservoir complexities tested in the last three-dimensional cases are present

in Norne. There are 63 faults in the model; some act as barriers inside the reservoir, while others collectively serve as boundaries of the active model itself. The total amount of NNC pairs is 11287, but this is not one-on-one correspondence, as some cells are able to receive/forward fluid flow from/to multiple non-neighbor connections at once. There are 113344 grid cells defined in the Norne model with dimensions of 46 x 112 x 22. Among those cells, only 44431 of them are active. All grid blocks outside the bordering faults are inactive cells.

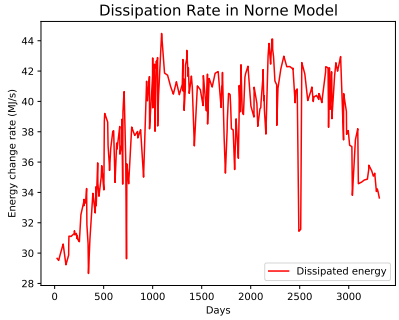
Plots of various energy components calculated for the Norne model are given as Figure 4.20. As one could observe, the energy rate in the Norne model is orders of magnitude higher than test cases at around 10 MJ/s (10 megawatts). Energy dissipation started slow but steadily increased its rate up to the level of 40-44 megawatts. Near the end of the simulation, the dissipation rate decreased by almost half. The initial external energy change was negative because at the beginning, only production wells are opened. A few years into the simulation, injection wells started injecting water and gas to supply energy into the reservoir, and the external energy change turned to positive. Throughout the simulation, the model constantly releases its stored internal energy at the rate of around 20 to 40 megawatts to drive flow through the reservoir and into the wells.



(a) External energy change rate.



(b) Internal energy change rate.



(c) Energy dissipation rate.

Figure 4.20: External energy, internal energy, and dissipation rate for the Norne model.

Figure 4.21 shows the energy balance plot for the Norne model. Energy balance (in solid black line) is achieved throughout the simulation, confirming the developed energy calculations' validity. However, the calculations for Norne are much more difficult than what was discussed step-by-step in Section 3.1, even though they are developed using the same procedure in Figure 3.2. Energy calculations in the reservoir for Norne encompass three different dimensions, meaning there are three flow directions. In addition to that, extra interblock flow from NNC pairs has to be calculated. Inactive cells also complicate the calculations, as discussed when calculating energy balance in the 3D case with complexities (Subsection 4.1.3). One feature that did not exist in the test cases but plays a vital role in Norne is transmissibility multipliers. Geological zonation of Norne (Table 3.3) lists different formations present in the model. These different formations could be thought of as compartments that allow very little to no flow between them. This is implemented in the model by transmissibility multipliers that have values close to zero. Energy calculations in wells require each connection in each well to be checked in order to fetch its injection/production rate data. In Norne, there are 36 simulation wells, and each of them has multiple connections with the reservoir. The restart file (`.UNRST`) provides the location of each connection, and the injection/production rate of each connection of each well is fetched from the summary file (`.UNSMRY`).

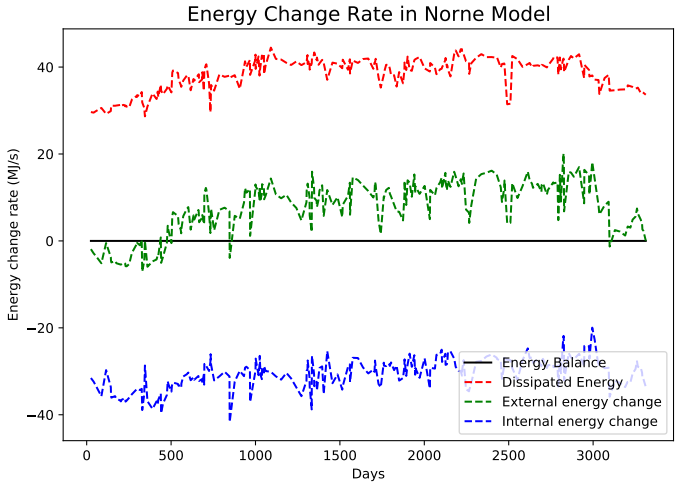


Figure 4.21: Energy balance plot for the Norne reservoir model.

Figure 4.22 presents the full overview of the Norne model, showing energy dissipation distribution throughout the reservoir at time step 138, around the middle part of the simulation, when the dissipation rate is at its peak. Observe that the dissipation rate is highest at layer #21, the second layer from the bottom. This corresponds to Tilje 2 formation according to Rwechungura et al. (2010). Checking the layer individually, it is found that

dissipation is indeed very high across the layer, as shown in Figure 4.23. In fact, this layer alone dissipates 43% of the total reservoir dissipation. The reason for this is the high vertical permeability of Tilje 2 and Tilje 1, the layer directly below it. This creates high vertical transmissibility between the two layers. In addition to that, a large pressure difference exists between them. The combination of this is a large vertical flow rate from Tilje 1 to Tilje 2 that dissipates a huge amount of energy.

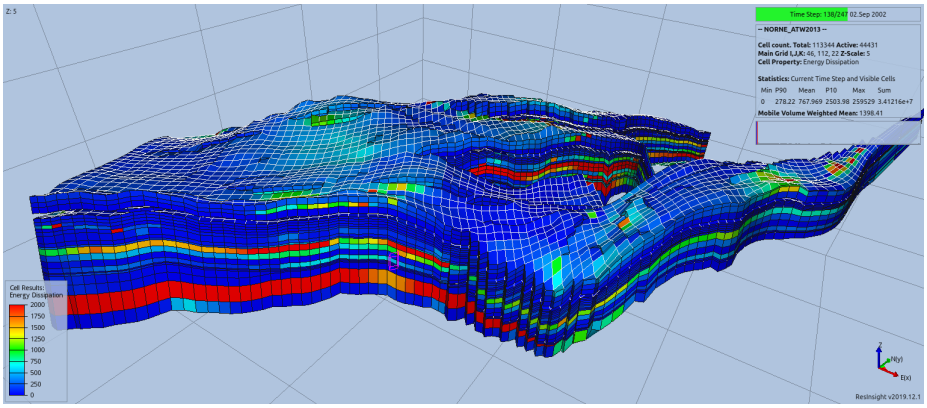


Figure 4.22: Energy dissipation distribution throughout the Norne reservoir model at timestep 138, 2 September 2002.

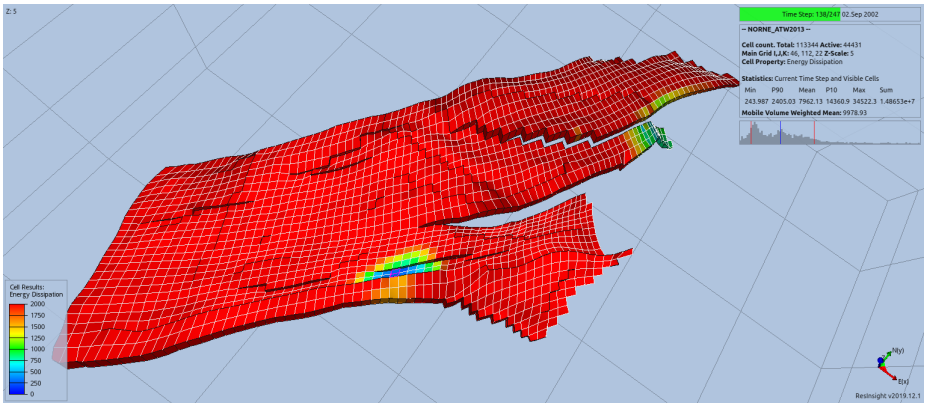


Figure 4.23: Energy dissipation distribution throughout layer #21 (Tilje 2) of the Norne simulation model.

Another layer that dissipates a large amount of energy is layer #13, which represents Tofte 2.1.3 (Rwechungura et al., 2010). This is where most of the oil resides for the main structure of the Norne model. Oil saturation in this part varies around 0.80 to 0.95 at the beginning of the simulation. For this reason, most wells have connection with this layer in order to extract hydrocarbon or to inject fluids to assist recovery. For the same observed

time step as before (2 September 2002), Figure 4.24 provides a look at dissipation on layer #13. Simulation wells visibility is turned on to show that most of the wells produce/inject to this layer.

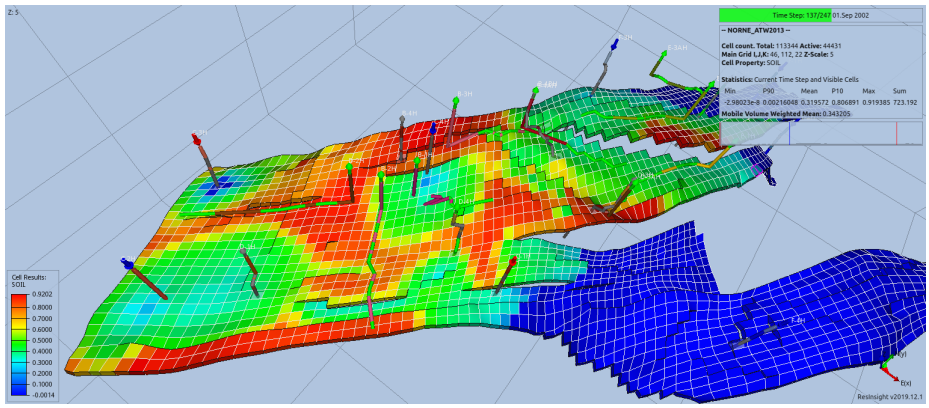


Figure 4.24: Energy dissipation distribution throughout layer #13 (Tofte 2.1.3) of the Norne simulation model. Simulation wells are shown.

Internal energy distribution in the model presents a look at how each layer and each formation communicates with each other, as shown in Figure 4.25. The figure also displays which layers belong to each formation. Blue colored cells indicate negative internal energy change, caused by a net negative flow rate (higher amount of fluid volume exiting the cells than entering them). Red colored cells mean they are gaining more internal energy from a net positive flow rate. Meanwhile, green colored cells are that those have little change in internal energy, implying little or no flow rate into/out of the grid blocks. For each formation, except for Tilje, it is seen that the top formation layer is colored red, meaning that flow in each formation all directs to the top layer. This flow against gravity is driven by pressure difference, especially after the top layers are opened to production wells. The bottom layers of each formation lose internal energy due to fluid flow escaping the layers. Meanwhile, the layers with mostly green cells are layers with low vertical transmissibilities that could only support little flow rates. The two top layers of Tilje are some examples for this, essentially blocking the flow of water from Tilje 1 and 2 to upper formations. Lower layers of Ile also act as barrier to separate oil flow in Tofte and the gas flow in upper layers of Ile. The gas-bearing Garn is separated from other formations by the Not shale formation, located between Garn and Ile, and is set as inactive cells in the model.

Figures 4.22 to 4.25 are some examples of reservoir layers/segment observations regarding energy changes. More observations like these could be done to gain better understanding of various processes taking place throughout a hydrocarbon reservoir's field life. For future continuations of this subject, the developed energy calculations could be used when simulating a reservoir model prior to development and production. Energy dissipation among multiple production and injection wells could be determined, and the values could be compared for different development scenarios. In an era where production efficiency

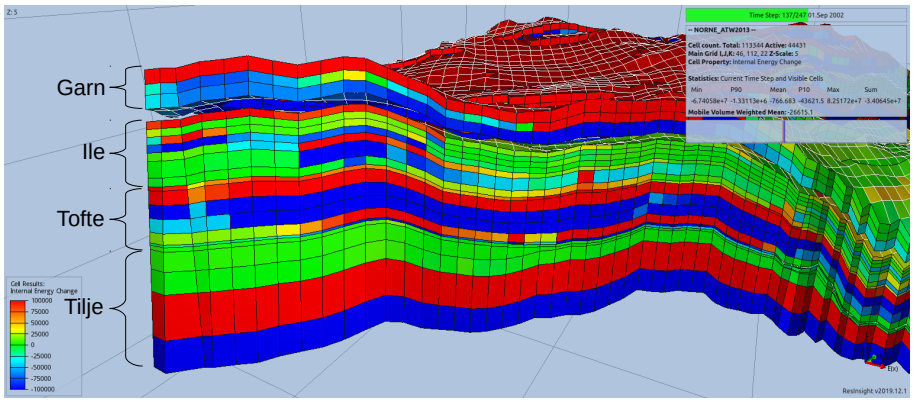


Figure 4.25: Internal energy change distribution among different layers in the Norne reservoir model.

is highly demanded, picking an optimal well placement based on energy efficiency could be beneficial for the overall production strategy. It would also be great to examine and plot energy changes for each simulation well, as it has not been done here. Using well production/injection rates and well dissipation rate, one could find which of the wells that waste energy unnecessarily. Optimal well control scenarios could be developed by careful observations on this subject.

Development of the open source software used in this study is encouraged. OPM Flow as a reservoir simulator has undergone a lot of improvements over the years, but some features important for energy calculations are still missing. For example, output of flow rates for individual well connections are currently not supported. Other than that, calculation of FVF for each grid block cannot be accessed directly using the `.DATA` file, but rather has to be defined using specific command line parameters. On one hand, improvements of open-source software could help further the subject of this thesis. On the other hand, this study and its future iterations could help improve reservoir engineering software development. As an example, ResInsight could include the energy calculations in this study as a feature. Furthermore, if energy efficiency calculation between wells as discussed earlier could be implemented, this could also be a beneficial feature to help reservoir management and decision making.

Chapter 5

Conclusions and Recommendations

5.1 Conclusions

This study has been done based on the objectives stated in Section 1.2:

1. Identify components required to calculate energy dissipation and energy balance, and develop the calculations to be used on reservoir models.

2. Apply the developed calculations for a full field, real reservoir simulation model.

Results of this study has been presented and discussed in Chapter 4. From the discussion, the conclusions can be taken and summarized as

1. Energy calculations for reservoir models have been developed. In order to get valid results from the calculations, the energy balance between various components has to be achieved. Figure 2.3 and Table 2.1 summarizes the different energy forms involved in reservoir recovery and the calculation of the energy balance.

    - Components of the energy balance have been identified: external energy, internal energy, and energy dissipation.
    - External energy is energy associated with systems outside the reservoir itself. Commonly, change in external energy is due to supply/discharge of energy through wells drilled in the reservoir.
    - Internal energy is the stored form of energy within the reservoir, and could be thought of as the ”potential energy” of the reservoir. Internal energy change is an effect of fluid compression/expansion due to changes in pressure.
    - Energy dissipation is a loss of energy as heat to the surrounding environment due to thermodynamic processes that happen within the reservoir, mainly fluid flow. As a fluid move due to a pressure difference, some energy is converted to heat and dissipated.

- Calculations of each energy component are developed while ensuring the energy balance among the external, internal, and dissipation energy is achieved according to Equation 1.1.

2. The developed calculations have been applied to several synthetic cases to test its robustness.
   - Seven test cases, as listed in Table 3.2, are built for testing the calculations. The cases have different features, starting from a simple single-phase one-dimensional model, up to a three-dimensional model with various complexities.
   - Results from the test cases have proven the validity of the developed energy calculations, and energy balance is achieved in all test cases.
   - The test cases show that the developed calculations are capable to handle energy computations in reservoir and in well bottomholes, energy changes due to non-neighbor connections, and effects of faults and inactive simulation cells.
3. Finally, the energy calculations have been applied to the Norne reservoir model, to show whether it is capable to function as they are supposed to in a full field, real reservoir model.
   - Calculation of each energy components have been completed for the Norne model, and the energy balance is successfully achieved.
   - Energy changes in the reservoir are visualized in the Norne model, showing how dissipation and internal energy changes are distributed throughout the model.
   - Results from energy calculation and visualization could provide insights on the reservoir itself and the processes that happen within it throughout its lifetime.

5.2 Recommendations

For any future works regarding energy calculations in reservoir models, these are some suggestions to expand and improve upon the research done in this study:

1. Calculation for energy dissipation between injection-production well pairs. Finding which well pairs are the most energy efficient could help in decision making processes prior to development and production.
2. Plotting and observation of energy changes in wells throughout a simulation. Optimal well control parameters could be obtained based on energy changes and production/injection rates for each well.
3. Implementation of energy calculations used in this study as a feature to open-source reservoir engineering software. This would create an opportunity to let more people access the research subject, providing benefits of this study to the public and at the same time attract more interest to expand on this subject.

Bibliography

Batchelor, C.K., Batchelor, G.K., 2000. An Introduction to Fluid Dynamics. Cambridge University Press. Google-Books-ID: Rla7OihRvUgC.

Baxendale, D., 2019. Open Porous Media Flow Documentation Manual version 2019-10 , 1755URL: https://opm-project.org/wp-content/uploads/2019/12/OPM-Flow-Documentation-2019-10-Rev-1.pdf.

Baxendale, D., 2020. Open Porous Media Flow Documentation Manual version 2020-04 , 1897URL: https://opm-project.org/wp-content/uploads/2020/05/OPM_Flow_Documentation_2020-04_Rev-0.pdf.

Berg, C.F., Slotte, P.A., 2020. Reservoir Simulation Through Open Source Software , 198.

Craft, B.C., Hawkins, M.F., Terry, R.E., 1991. Applied Petroleum Reservoir Engineering. Prentice Hall. Google-Books-ID: uDFQAQAAIAAJ.

Dake, L.P., 1978. Fundamentals of Reservoir Engineering. Number 8 in Developments in petroleum science, Elsevier Scientific Pub. Co. ; distributors for the U.S. and Canada Elsevier North-Holland, Amsterdam ; New York : New York.

Dale, S.I., Sjaastad, M., Hogstol, H., Rustad, A.B., 2012. Improving Visualization of Large Scale Reservoir Models, Society of Petroleum Engineers. URL: https://www.onepetro.org/conference-paper/SPE-163088-MS?sort=&start=0&q=resinsight&from_year=&peer_reviewed=&published_between=&fromSearchResults=true&to_year=&rows=25#, doi:10.2118/163088-MS.

Ertekin, T., Abou-Kassem, J.H., King, G.R., 2001. Basic Applied Reservoir Simulation.

Gjerstad, H.M., Steffensen, L., Skagen, J.I., 1995. The Norne Field - Exploration History & Reservoir Development Strategy, Offshore Technology Conference. URL: https://www.onepetro.org/conference-paper/OTC-7924-MS, doi:10.4043/7924-MS.

Green, D.W., Wilhite, G.P., 2018. Enhanced Oil Recovery. SPE, Richardson, TX. URL: https://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=5381116.

Hassanizadeh, S.M., Gray, W.G., 1990. Mechanics and Thermodynamics of Multiphase Flow in Porous Media Including Interphase Boundaries. Advances in Water Resources 13, 169–186. URL: http://www.sciencedirect.com/science/article/pii/030917089090040B, doi:10.1016/0309-1708(90)90040-B.

Khanamiri, H.H., Berg, C.F., Slotte, P.A., Schlüter, S., Torsæter, O., 2018. Description of Free Energy for Immiscible Two-Fluid Flow in Porous Media by Integral Geometry and Thermodynamics. Water Resources Research URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018WR023619.

Kundu, P.K., Cohen, I.M., Dowling, D.R., 2015. Fluid Mechanics. 6 ed., Academic Press. URL: http://gen.lib.rus.ec/book/index.php?md5=cf6c245996b5a115628399da38330559.

Muskat, M., 1981. Physical Principles of Oil Production URL: https://www.osti.gov/biblio/5423807-physical-principles-oil-production.

Neidinger, R.D., 2010. Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming. SIAM Review 52, 545–563. URL: http://epubs.siam.org/doi/10.1137/080743627, doi:10.1137/080743627.

Rasmussen, A.F., Sandve, T.H., Bao, K., Lauser, A., Hove, J., Skaflestad, B., Klöfkorn, R., Blatt, M., Rustad, A.B., Sævareid, O., Lie, K.A., Thune, A., 2019. The Open Porous Media Flow Reservoir Simulator. arXiv:1910.06059 [physics] URL: http://arxiv.org/abs/1910.06059.

Rwechungura, R., Suwartadi, E., Dadashpour, M., Kleppe, J., Foss, B., 2010. The Norne Field Case—A Unique Comparative Case Study , 14.

Whitson, C.H., Brulé, M.R., 2000. Phase Behavior. Number v. 20 in Monograph / SPE ; Henry L. Doherty series, Henry L. Doherty Memorial Fund of AIME, Society of Petroleum Engineers, Richardson, Tex.

Appendix

A Energy Calculation Scripts used in the Norne Reservoir Model

The scripts included here are also available at this GitHub repository:
https://github.com/fdlberylian/TPG4920-Masters-Thesis

A.1 `energy_reservoir_norne.py`

```
import rips
import time
import grpc
import math
from operator import itemgetter
from ecl.eclfile import EclFile
from ecl.grid import EclGrid

# Connect to ResInsight
resinsight = rips.Instance.find()
start = time.time()
case = resinsight.project.cases()[0]
num_tsteps = len(case.time_steps())
name = case.name
grids = case.grids()
for grid in grids:
    dimension = grid.dimensions()
Nx = dimension.i
Ny = dimension.j
Nz = dimension.k

# Read output files
egrid_file = EclFile("%s.EGRID" % name)
init_file = EclFile("%s.INIT" % name)
egrid = EclGrid("%s.EGRID" % name)

# Read ACTNUM numbers from EGRID file
actnum = egrid_file["ACTNUM"][0]
active_cells = []
for i in range(len(actnum)):
    if actnum[i] == 1:
        active_cells.append(i+1)
num_active_cells = len(active_cells)

# Read NNC pairs from EGRID and INIT file
```

```
nnc_list = []
if egrid_file.has_kw("NNC1"):
    nnc1 = egrid_file["NNC1"][0]
    nnc2 = egrid_file["NNC2"][0]
    tran = init_file["TRANNNC"][0]
    for g1,g2,t in zip(nnc1,nnc2,tran):
        nnc_list.append((g1,g2,t))

# Calculation of energy dissipation for each cell
total_ed = 0.0
for tstep in range(219,num_tsteps):
    print("Timestep", tstep, "of", num_tsteps)

    # Read results into list
    tranx_results = case.active_cell_property('STATIC_NATIVE', 'TRANX', 0)
    trany_results = case.active_cell_property('STATIC_NATIVE', 'TRANY', 0)
    tranz_results = case.active_cell_property('STATIC_NATIVE', 'TRANZ', 0)
    multx_results = case.active_cell_property('STATIC_NATIVE', 'MULTX', 0)
    multy_results = case.active_cell_property('STATIC_NATIVE', 'MULTY', 0)
    multz_results = case.active_cell_property('STATIC_NATIVE', 'MULTZ', 0)
    pres_results = case.active_cell_property('DYNAMIC_NATIVE', 'PRESSURE',
     tstep)
    krw_results = case.active_cell_property('DYNAMIC_NATIVE', 'WATKR',
    tstep)
    kro_results = case.active_cell_property('DYNAMIC_NATIVE', 'OILKR',
    tstep)
    krg_results = case.active_cell_property('DYNAMIC_NATIVE', 'GASKR',
    tstep)
    muw_results = case.active_cell_property('DYNAMIC_NATIVE', 'WAT_VISC',
    tstep)
    muo_results = case.active_cell_property('DYNAMIC_NATIVE', 'OIL_VISC',
    tstep)
    mug_results = case.active_cell_property('DYNAMIC_NATIVE', 'GAS_VISC',
    tstep)

    # Determination of upstream/downstream cells in each direction
    prevx_pres_results = [0.0] * num_active_cells
    nextx_pres_results = [0.0] * num_active_cells
    prevy_pres_results = [0.0] * num_active_cells
    nexty_pres_results = [0.0] * num_active_cells
    prevz_pres_results = [0.0] * num_active_cells
    nextz_pres_results = [0.0] * num_active_cells

    for c in range(num_active_cells):
        print("pres calculation: checking cell", c+1, "of",
    num_active_cells, end="\r")
        idx = active_cells[c]
        plane_num = idx%(Nx*Ny)
        layer_num = math.floor(idx/(Nx*Ny))
        if idx-1 in active_cells and idx%Nx != 0:
            prevx_pres_results[c] = pres_results[active_cells.index(idx-1)
    ]

        if idx+1 in active_cells and (idx+1)%Nx != 0:
            nextx_pres_results[c] = pres_results[active_cells.index(idx+1)
    ]

```

```
        if idx-Nx in active_cells:
            prevy_pres_results[c] = pres_results[active_cells.index(idx-Nx
    )]
            if plane_num < Nx:
                prevy_pres_results[c] = 0.0

        if idx+Nx in active_cells:
            nexty_pres_results[c] = pres_results[active_cells.index(idx+Nx
    )]
            if plane_num+Nx >= (Nx*Ny):
                nexty_pres_results[c] = 0.0

        if idx-(Nx*Ny) in active_cells:
            prevz_pres_results[c] = pres_results[active_cells.index(idx-Nx
    *Ny)]
            if layer_num == 0:
                prevz_pres_results[c] = 0.0

        if idx+(Nx*Ny) in active_cells:
            nextz_pres_results[c] = pres_results[active_cells.index(idx+Nx
    *Ny)]
            if layer_num == Nz-1:
                nextz_pres_results[c] = 0.0
    print(end='\r')

    # Calculate interblock flowrates in + direction(s)
    flow_x_results = []
    flow_y_results = []
    flow_z_results = []
    zip_results_f = list(zip(tranx_results, trany_results, tranz_results,
    multx_results, multy_results, multz_results, pres_results,
    prevx_pres_results, nextx_pres_results, prevy_pres_results,
    nexty_pres_results, prevz_pres_results, nextz_pres_results,
    krw_results, kro_results, krg_results, muw_results, muo_results,
    mug_results))
    for (tranx, trany, tranz, multx, multy, multz, pres, prevx_pres,
    nextx_pres, prevy_pres, nexty_pres, prevz_pres, nextz_pres, krw, kro,
    krg, muw, muo, mug) in zip_results_f:
        if nextx_pres > 0.0 and tranx > 0.0:
            flow_x = tranx * multx * ((krw/muw)+(kro/muo)+(krg/mug)) * (
    pres-nextx_pres)
        else:
            flow_x = 0.0

        if nexty_pres > 0.0 and trany > 0.0:
            flow_y = trany * multy * ((krw/muw)+(kro/muo)+(krg/mug)) * (
    pres-nexty_pres)
        else:
            flow_y = 0.0

        if nextz_pres > 0.0 and tranz > 0.0:
            flow_z = tranz * multz * ((krw/muw)+(kro/muo)+(krg/mug)) * (
    pres-nextz_pres)
        else:
            flow_z = 0.0

        flow_x_results.append(flow_x)
```

```
        flow_y_results.append(flow_y)
        flow_z_results.append(flow_z)

    # Determination of downstream flowrates in each direction
    prevx_flow_results = [0.0] * num_active_cells
    prevy_flow_results = [0.0] * num_active_cells
    prevz_flow_results = [0.0] * num_active_cells
    for c in range(num_active_cells):
        print("flow calculation: checking cell", c+1, "of",
    num_active_cells, end="\r")
        idx = active_cells[c]
        plane_num = idx%(Nx*Ny)
        layer_num = math.floor(idx/(Nx*Ny))
        if idx-1 in active_cells and idx%Nx != 0:
            prevx_flow_results[c] = flow_x_results[active_cells.index(idx
    -1)]

        if idx-Nx in active_cells:
            prevy_flow_results[c] = flow_y_results[active_cells.index(idx-
    Nx)]
            if plane_num < Nx:
                prevy_flow_results[c] = 0.0

        if idx-(Nx*Ny) in active_cells:
            prevz_flow_results[c] = flow_z_results[active_cells.index(idx-
    Nx*Ny)]
            if layer_num == 0:
                prevz_flow_results[c] = 0.0
    print(end='\r')

    # Generate energy change lists as results
    e_dis = []
    e_int = []
    # 1: Calculate energy changes in reservoir
    zip_results_e = list(zip(pres_results, prevx_pres_results,
    nextx_pres_results, prevy_pres_results, nexty_pres_results,
    prevz_pres_results, nextz_pres_results, flow_x_results, flow_y_results
    , flow_z_results, prevx_flow_results, prevy_flow_results,
    prevz_flow_results))
    for (pres, prevx_pres, nextx_pres, prevy_pres, nexty_pres, prevz_pres,
     nextz_pres, flow_x, flow_y, flow_z, prevx_flow, prevy_flow,
    prevz_flow) in zip_results_e:
        ed_x = flow_x * (pres - nextx_pres)
        ed_y = flow_y * (pres - nexty_pres)
        ed_z = flow_z * (pres - nextz_pres)

        ei_x = pres * (prevx_flow - flow_x)
        ei_y = pres * (prevy_flow - flow_y)
        ei_z = pres * (prevz_flow - flow_z)

        e_dis.append((ed_x + ed_y + ed_z) * (1e5/86400))     # Convert
    unit to J/s
        e_int.append((ei_x + ei_y + ei_z) * (1e5/86400))

    #2: Calculate energy changes from flow between NNC pairs
    for n in range(len(nnc_list)):
        print("Checking NNC", n+1, "of", len(nnc_list), end="\r")
```

```
        orig = active_cells.index(nnc_list[n][0])
        dest = active_cells.index(nnc_list[n][1])
        param1 = zip_results_f[orig]
        param2 = zip_results_f[dest]
        tran = nnc_list[n][2]
        flow_nnc = tran * ((param1[13]/param1[16])+(param1[14]/param1[17])
    +(param1[15]/param1[18])) * (param1[6]-param2[6])
        ed_nnc = flow_nnc * (param1[6]-param2[6]) * (1e5/86400)
        ei_nnc_1 = -param1[6] * flow_nnc * (1e5/86400)
        ei_nnc_2 = param2[6] * flow_nnc * (1e5/86400)

        e_dis[orig] += ed_nnc
        e_int[orig] += ei_nnc_1
        e_int[dest] += ei_nnc_2
    print("Total energy dissipation:",sum(e_dis),"J/s")
    print("Total internal energy change:",sum(e_int),"J/s")

    try:
      # Send back output result
        case.set_active_cell_property(e_dis, 'GENERATED', 'Energy
    Dissipation', tstep)
        case.set_active_cell_property(e_int, 'GENERATED', 'Internal Energy
     Change', tstep)
    except grpc.RpcError as e:
        print("Exception Received: ", e)

end = time.time()
print("Time elapsed: ", end - start)
print("Transferred all results back\n")

view = case.views()[0].apply_cell_result('GENERATED', 'Energy Dissipation'
    )
```

A.2 energy_well_norne.py

```
import rips
import time
import grpc
import math
import os
from operator import itemgetter
from ecl.eclfile import EclFile
from ecl.grid import EclGrid
from ecl.summary import EclSum
import matplotlib.pyplot as plt

# Define the function to calculate energy changes in well bottomholes
def energywell():
    # Connect to ResInsight
    resinsight = rips.Instance.find()
    case = resinsight.project.cases()[0]
    num_tsteps = len(case.time_steps())
    name = case.name
    grids = case.grids()
    for grid in grids:
        dimension = grid.dimensions()
    Nx = dimension.i
```

```
    Ny = dimension.j
    Nz = dimension.k

    class well:
        def __init__(self, name, idx, welltype):
            self.name = name
            self.idx = idx
            self.type = welltype


    # Read EGRID, RST and INIT files
    summary_file = EclSum("%s.UNSMRY" % name)
    egrid_file = EclFile("%s.EGRID" % name)
    rst_file = EclFile("%s.UNRST" % name)
    timestep_width = []
    days = []
    for tstep in range(num_tsteps):
        if tstep==0:
            width = rst_file.iget_restart_sim_days(tstep)
        else:
            width = rst_file.iget_restart_sim_days(tstep) - rst_file.
    iget_restart_sim_days(tstep-1)

        timestep_width.append(width)
        days.append(rst_file.iget_restart_sim_days(tstep))

    # Read ACTNUM numbers from EGRID file
    actnum = egrid_file["ACTNUM"][0]
    active_cells = []
    for i in range(len(actnum)):
        if actnum[i] == 1:
            active_cells.append(i)

    # Convert summary file timesteps to restart file timesteps
    summary_days = summary_file.days
    idx_trim = []
    for d in days[2:]:
        add = summary_days.index(d)
        idx_trim.append(add)

    energy_balance = []
    energy_external = []
    energy_internal = []
    energy_dissipated = []
    for tstep in range(2,num_tsteps):
        print("Timestep", tstep, "of", num_tsteps)

        # List active wells in the timestep
        zwel = rst_file.iget_named_kw("ZWEL",tstep)
        nwells = int(len(zwel)/3)
        well_list = []
        for wel in range(nwells):
            welname = rst_file["ZWEL"][tstep][wel*3]
            welname = welname.rstrip()
            niwelz = int(len(rst_file["IWEL"][tstep]) / nwells)

            if rst_file["IWEL"][tstep][wel*niwelz + 6] == 1:
```

```
                weltype = "PROD"
            else:
                weltype = "INJE"

            ncwmax = int(len(rst_file["ICON"][tstep]) / (25*nwells))
            welidx = []
            for ncw in range(ncwmax):
                numicon = 25*(wel*ncwmax + ncw)
                weli = rst_file["ICON"][tstep][numicon+1]
                welj = rst_file["ICON"][tstep][numicon+2]
                welk = rst_file["ICON"][tstep][numicon+3]
                if weli != 0:
                    welidx.append(int(welk-1)*(Nx*Ny) + int(welj-1)*(Nx) +
 int(weli-1))

            well_list.append(well(welname,welidx,weltype))

        # Read results into list
        porv = case.active_cell_property('STATIC_NATIVE', 'PORV', 0)
        pres = case.active_cell_property('DYNAMIC_NATIVE', 'PRESSURE',
tstep)
        bo = case.active_cell_property('DYNAMIC_NATIVE', 'BO', tstep)
        bw = case.active_cell_property('DYNAMIC_NATIVE', 'BW', tstep)
        bg = case.active_cell_property('DYNAMIC_NATIVE', 'BG', tstep)

        # Fetch results from the reservoir energy calculation
        edis = case.active_cell_property('GENERATED', 'Energy Dissipation'
, tstep)
        eint = case.active_cell_property('GENERATED', 'Internal Energy
Change', tstep)

        # Calculate energy changes in well bottomholes
        e_external = []
        e_internal_well = []
        ed_well = []
        for wel in well_list:
            idx = idx_trim[tstep-2]
            pcel = case.active_cell_property('DYNAMIC_NATIVE', 'PRESSURE',
 tstep)
            for wel_idx in wel.idx:
                wel_idx_act = active_cells.index(wel_idx)

                pwel = pcel[wel_idx_act]
                bo_wel = bo[wel_idx_act]
                bw_wel = bw[wel_idx_act]
                bg_wel = bg[wel_idx_act]
                bhp = summary_file["WBHP:%s" % wel.name][idx].value

                if wel.type == "PROD":
                    wpr = summary_file["CWPR:%s:%i" % (wel.name,wel_idx+1)
][idx].value
                    opr = summary_file["COPR:%s:%i" % (wel.name,wel_idx+1)
][idx].value
                    gpr = summary_file["CGPR:%s:%i" % (wel.name,wel_idx+1)
][idx].value
                    e_external.append(-bhp * (wpr*bw_wel+opr*bo_wel+gpr*
bg_wel) * 1e5/86400)
```

```
                    e_internal_well.append(-pwel * (wpr*bw_wel+opr*bo_wel+
    gpr*bg_wel) * 1e5/86400)
                    ed_well.append((wpr*bw_wel+opr*bo_wel+gpr*bg_wel) * (
    pwel-bhp) * 1e5/86400)
                else:
                    wir = summary_file["CWIR:%s:%i" % (wel.name,wel_idx+1)
    ][idx].value
                    gir = summary_file["CGIR:%s:%i" % (wel.name,wel_idx+1)
    ][idx].value
                    e_external.append(bhp * (wir*bw_wel+gir*bg_wel) * 1e5
    /86400)
                    e_internal_well.append(pwel * (wir*bw_wel+gir*bg_wel)
    * 1e5/86400)
                    ed_well.append((wir*bw_wel+gir*bg_wel) * (bhp-pwel) *
    1e5/86400)

        edis_res = sum(edis)
        edis_well = sum(ed_well)
        e_dissipated = edis_res + edis_well
        e_internal = sum(eint) + sum(e_internal_well)

        # Calculate each component in energy balance
        e_balance = (sum(e_external) - (e_dissipated + e_internal))
        energy_balance.append(e_balance)
        energy_dissipated.append(e_dissipated)
        energy_external.append(sum(e_external))
        energy_internal.append(e_internal)
    days = days[2:]

    return(days,energy_balance,energy_external,energy_internal,
    energy_dissipated)
```

A.3 energy_balance_norne.py

```
import rips
import time
import grpc
import math
import os
from operator import itemgetter
from ecl.eclfile import EclFile
from ecl.grid import EclGrid
from ecl.summary import EclSum
import matplotlib.pyplot as plt
from energy_well_norne import energywell

start = time.time()

# Fetch energy balance calculation from energy_well.py
days,energy_balance,energy_external,energy_internal,energy_dissipated =
    energywell()

folder_name = os.path.dirname(os.path.abspath(__file__))
dirname = os.path.join(folder_name, 'resinsight_props')
if os.path.exists(dirname) is False:
    os.mkdir(dirname)

```

```
# Write calculation results in text files for future use
eext_filename = os.path.join(dirname, 'energy_external.txt')
eint_filename = os.path.join(dirname, 'energy_internal.txt')
edis_filename = os.path.join(dirname, 'energy_dissipation.txt')
ebal_filename = os.path.join(dirname, 'energy_balance.txt')
days_filename = os.path.join(dirname, 'days.txt')
with open(eext_filename,'w') as txt_file:
    for ii in range(len(energy_external)):
        txt_file.write('%f\n' % energy_external[ii])

with open(eint_filename,'w') as txt_file:
    for ii in range(len(energy_internal)):
        txt_file.write('%f\n' % energy_internal[ii])

with open(edis_filename,'w') as txt_file:
    for ii in range(len(energy_dissipated)):
        txt_file.write('%f\n' % energy_dissipated[ii])

with open(ebal_filename,'w') as txt_file:
    for ii in range(len(energy_balance)):
        txt_file.write('%f\n' % energy_balance[ii])

with open(days_filename,'w') as txt_file:
    for ii in range(2,len(days)):
        txt_file.write('%f\n' % days[ii])

end = time.time()
print("Time elapsed: ", end - start)
print("Transferred all results back")

# Generate plots
plt.figure(1)
plt.title('Energy Change Rate in Norne Model', fontsize=15)
plt.plot(days,energy_balance,color='k',label='Energy Balance')
plt.plot(days,energy_dissipated,color='r',label='Dissipated Energy',
    linestyle='--')
plt.plot(days,energy_external,color='g',label='External energy change',
    linestyle='--')
plt.plot(days,energy_internal,color='b',label='Internal energy change',
    linestyle='--')
plt.legend(loc='best')
plt.ylabel("Energy change rate (J/s)")
plt.xlabel("Days")

plt.figure(2)
plt.title('External Energy Change Rate in Norne Model', fontsize=15)
plt.plot(days,energy_external,color='g',label='External energy change',
    linestyle='-')
plt.legend(loc='best')
plt.ylabel("Energy change rate (J/s)")
plt.xlabel("Days")

plt.figure(3)
plt.title('Internal Energy Change Rate in Norne Model', fontsize=15)
plt.plot(days,energy_internal,color='b',label='Internal energy change',
    linestyle='-')
plt.legend(loc='best')
```

```
plt.ylabel("Energy change rate (J/s)")
plt.xlabel("Days")

plt.figure(4)
plt.title('Dissipation Rate in Norne Model', fontsize=15)
plt.plot(days,energy_dissipated,color='r',label='Dissipated energy',
    linestyle='-')
plt.legend(loc='best')
plt.ylabel("Energy change rate (J/s)")
plt.xlabel("Days")

plt.show()
```

NTNU

Norwegian University of
Science and Technology