NTNU
Norwegian University of
Science and Technology

Multiscale Simulation of Thermal Flow in Porous Media

Stine Brekke Vennemo

Master of Science in Physics and Mathematics
Submission date: February 2016
Supervisor: Knut Andreas Lie, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

ABSTRACT

In this thesis we develop a multiscale method that solves non-isothermal flow in porous media. A sequential formulation is established that decouples corresponding equations into three systems, one pertaining pressure, one pertaining temperature and one pertaining transport. The sequential method is then verified against a standard fully implicit method on two examples with varying degree of complexity on the permeability field. By adjusting a tolerance that decides the number of outer iterations in the sequential method, the method converges toward the fully implicit method. After the sequential method is verified, we use the sequential structure in the development of an accurate and efficient multiscale method for the pressure and temperature systems. This multiscale method is then vigorously tested against the sequential method on several examples with varying degrees of complex grids and permeability fields, as well as being verified for both single-phase and multiphase flow. The multiscale method can, by adjusting the parameters of the original sequential method, converge towards various fully implicit solutions, or give a reasonable approximation on the coarse scale. The experiments show that the multiscale method provides a flexible and stable solver that accurately solves the equations describing non-isothermal flow in porous media more efficiently than both the fully implicit method and the sequential method as long as heterogeneous permeabilities are used.

SAMMENDRAG

I denne mastergradsoppgaven utvikler vi en multiskalametode som løser ikke-isoterm flyt i porøst media. En sekvensiell formulering er etablert, hvor vi omgjør de korresponderende likningene til tre systemer, en tilhørende trykk, en tilhørende temperatur og en tilhørende transport. Den sekvensielle metoden testes deretter mot en standard fullt implisitt metode på to eksempler med varierende grad av kompleksitet på permeabilitetfeltet. Ved å justere en toleranse som bestemmer antall ytre iterasjoner i den sekvensielle metoden, konvergerer metoden mot den fullt implisitte metoden. Etter at den sekvensielle metoden er bekreftet, bruker vi den sekventielle strukturen i utviklingen av en nøyaktig og effektiv multiskalametode for trykk- og temperatursystemene. Denne metoden er deretter kraftig testet mot den sekvensielle metoden på flere eksempler med varierende grad av komplekse grid og permeabilitetfelt, så vel som på både enkeltfase og flerfasestrømning. Multiskalametoden kan, ved å justere parametrene til den sekvensielle metoden, konvergere mot forskjellige fullt implisitte løsninger, eller gi en rimelig approksimasjon på grov skala. Forsøkene viser at multiskalametoden gir en fleksibel og stabil løser som nøyaktig løser likninger som beskriver ikke-isoterm strømning i porøst media mer effektivt enn både den fullt implisitte metoden og den sekvensielle metoden så lenge heterogene permeabiliteter blir brukt.

My favorite part of a book is the dedication.
I would therefore like to thank whomever it was who decided I needed to write this thesis, thereby giving me an opportunity to write a dedication of my own.

I wrote this thesis for myself, and I therefore dedicate this to me.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Knut-Andreas Lie for all the guidance he has given me and for the valuable feedback I have received. I would also like to thank SINTEF Applied Mathematics for not only giving me access to their code, but also welcoming me to the lunch table and into their team. I would further like to thank Sindre Hilden for sharing his office with me and answering my questions, as well as Olav Møyner for all the help this last year. In addition I thank Statoil (operator of the Norne field) and its license partners ENI and Petoro for the release of the Norne data, and I acknowledge the Center for Integrated Operations at NTNU for cooperation and coordination of the Norne Cases. For the big help proofreading this thesis I thank Mathilde Skylstad, Au-Dung Vuong, Kristin Solbakken, and my dad. Finally, I would like to thank my friends and family for everything. I truly know how lucky I am to have you all in my life. Thanks for keeping the world at bay.

CONTENTS

1

INTRODUCTION

Petroleum reservoirs have been studied actively for many years. In mathematics, this is manifested in the study of the equations that describe the reservoirs, as well as the development of numerical methods that can accurately solve the equations and thus simulate the different processes of reservoir simulation.

The reservoirs consist of porous or fractured rock formations, where the hydrocarbons are contained in the pores of the rocks. Pressure differences, capillary forces and buoyancy will force the fluids to flow through these pores, and thermal effects can change the properties of fluid, thereby affecting the flow. When we are to simulate flow in porous media we thus have to account for both the pressure and temperature.

The simplest and most obvious way to solve the equations describing flow in porous media is to order the equations in a system, apply a fully implicit time discretization and solve for all the unknowns simultaneously. This constitutes an accurate solver, but the system is a large one when we model reservoirs. A lot of computer memory will therefore be used, and the method will not be very efficient. To account for this, we will apply a sequential method which decouples the system into several subsystems and solves for each unknown separately and more efficiently. To further improve the efficiency, we will thereafter apply a multiscale method.

The idea behind the multiscale methods is to employ basis functions that restrict the equations from the fine grid that represents the reservoir to a coarser grid where we will have fewer unknowns. Here, the equations can be solved more efficiently. Then we use new basis functions to prolong the coarse scale solution back to the original fine scale. The basis functions are constructed so that fine scale heterogeneities are preserved.

There have been developed several different types of multiscale methods [11, 7, 9, 1, 17, 6, 12, 19]. The difference comes from the individual ways the methods choose to restrict and especially prolong the unknowns. We are going to employ the newly developed multiscale restricted-smoothed basis (MsRSB) method [19, 20, 18, 8] in this thesis. The method uses a control volume sum-

mation operator to restrict the system, and to prolong the system it employs basis functions found through an iterative scheme that constructs the functions so that they are properly smooth and consistent with local properties.

The MsRSB method has previously been tested and found to be working well when we have different isothermal processes [19, 20, 18, 8]. Though it is true that we in many circumstances have isothermal processes, there are other times when heat change is an important part of the reservoir. Warming up the wells has for instance been used as a method in enhanced oil recovery [14]. Temperature is also important in different chemical methods, where the reactions heavily depend on temperature. By heating up the fluids we will change the flow properties, and we have to check how this affects the flow. A good method must work consistently for all processes that are typical for the given model. It is therefore important to test that the MsRSB method works on non-isothermal processes, because non-isothermal flow is a realistic scenario in petroleum reservoirs. This is what will be done in this thesis, where we construct a non-isothermal model and first solve the equations using the sequential method, before testing whether the MsRSB method offer additional efficiency while still preserving the accuracy and advantages of the sequential method.

We are going to start off by deriving the model equations in Chapter 2 to give background information and a thorough introduction to petroleum reservoirs. In Chapter 3, we introduce the standard time and space discretizations used when solving the equations numerically, as well as the well model and the Newton-Raphson method which will be used both as an introductory numerical solver, the fully implicit method, and as a tool in the later methods. Chapter 4 contains a formulation of the sequential method, which will solve the system more efficiently than the fully implicit method. The chapter also includes a couple of examples that check whether we have managed to implement the method correctly. In Chapter 5 we introduce the multiscale method that, used on top of the sequential method, should further improve efficiency. In the end of the chapter we present several examples that vigorously test the multiscale method. A conclusion of the thesis is offered in Chapter 6.

2

MATHEMATICAL MODEL

In this thesis, we are going to study non-isothermal flow in oil reservoirs. Oil plays a big part in the world we live in and it is therefore important to be able to understand and model its recovery. To understand flow properties in petroleum reservoirs one needs to get a knowledge of the physical and mathematical concepts that describe them. We therefore start by presenting the derivation of the equations that make up the mathematical model of reservoirs, before we in later chapters present numerical methods that solve the derived equations.

Reservoirs normally contain multiple components, such as water and different hydrocarbon elements. These components form different phases, such as water, gas and oil, which flow through the reservoir. As these models can be complex, we are going to start with a simplification when we derive the equations, namely the assumption that the reservoirs contain only one phase. By first studying single-phase, it will be easier to understand multiphase, our main field of study, later on. We are going to let the single-phase be oil, but it is also possible to look at gas or water. After studying the single-phase case, we are going to extend the model and derive the equations for multiple components, multiple phases, before we study a specific case, namely the black-oil model. But first, the single-phase equations.

2.1 SINGLE-PHASE FLOW

Several mechanisms come to play when fluids are transported in porous media. Pressure differences will cause bulk flow, as the fluid moves from an area of high pressure to an area of low pressure. Gravity can force the fluid to move in a certain direction, while thermal conductivity and heat transport will heat or cool the fluid, which will affect the flow. In addition comes the properties of the porous media, such as the porosity of the rock. To account for the different mechanisms we need two equations, one that describes the bulk flow, and one that describes the heat transfer.

To derive the model equations, we assume a *continuum hypothesis*. This means that we assume that the fluid properties are defined at infinitesimal small points,

and ignore the fact that the fluid is made up of discrete particles. That is, we say that properties such as pressure, temperature and velocity have a specific value for each infinitesimal point.

2.1.1 Single-Phase Flow Equation

The fundamental principle used to derive equations describing the bulk flow of a single fluid is the *law of conservation of mass*, which states that the mass of a control volume Ω must remain constant over time. Said differently, this is the same as

$$\begin{aligned}\text{rate of change of mass in } \Omega = {} & \text{mass flow across the boundary } \partial\Omega \\ & + \text{mass production/consumption in } \Omega.\end{aligned}$$

By looking at a volume-element of Ω, call it $\mathrm{d}\Omega$, the mass of the volume is given by $\phi\rho\mathrm{d}\Omega$, where ϕ is the porosity of the medium, i.e., the fraction of the rock that contains fluid, and ρ is the density, i.e., the mass per unit of fluid. Note that ϕ depends on the pressure of the fluid, while ρ depends on both pressure and temperature. To get the mass of the reference volume Ω, we sum up all the infinitesimal volume elements. The rate of change of mass is hence given by

$$\frac{\partial}{\partial t}\int_\Omega \phi\rho \,\mathrm{d}\Omega.$$

Mass flow is determined by the flux. As the flow follows the velocity of the fluid, and as we are only interested in the normal component of the boundary, the mass flux is given by $j = -\rho\vec{v}\cdot\vec{n}$, where $\vec{v}$ is the fluid's velocity and $\vec{n}$ is the normal vector of the boundary of the infinitesimal volume-element $\mathrm{d}\Omega$. The mass flow through $\partial\Omega$ is thus obtained by

$$-\int_{\partial\Omega} \rho\vec{v}\cdot\vec{n} \,\mathrm{d}S,$$

where we use a surface integral as we are concerned with the flow through the boundary of Ω.

Mass production/consumption in an infinitesimal volume-element of Ω is given by a source term q. Summing up all the volume-elements of Ω yields

$$\int_\Omega q \,\mathrm{d}\Omega.$$

The law of conservation of mass will thus become

$$\int_\Omega \frac{\partial}{\partial t}(\phi\rho) \,\mathrm{d}\Omega = -\int_{\partial\Omega} \rho\vec{v}\cdot\vec{n} \,\mathrm{d}S + \int_\Omega q \,\mathrm{d}\Omega,$$

where the derivative has been placed inside the integral because Ω is fixed. Using the divergence theorem, we get

$$\int_\Omega \left[\frac{\partial}{\partial t}(\phi\rho) + \nabla \cdot (\rho\vec{v}) - q \right] \, \mathrm{d}\Omega = 0.$$

As this is true for any control volume Ω, we have that the integrand must be zero, and the continuity equation for a single-phase fluid is thus given by

$$\frac{\partial}{\partial t}(\phi\rho) + \nabla \cdot (\rho\vec{v}) = q.$$

The velocity of a fluid flowing through a porous medium is given by *Darcy's law*. It was formulated by Henry Darcy in 1856, who found, through experiments where he looked at the flow of water through sand, that the flux rate of the fluid is related to the hydrostatic pressure difference and the permeability, the mediums ability to transmit fluid. It has later been discovered that the fluid's resistance to flow, the viscosity, also influences the fluid flow. Darcy might have discovered this if he had used more than one fluid in his experiments. The law has later been derived from the Navier-Stokes equations as well. All in all, Darcy's law states that

$$\vec{v} = -\frac{\mathbf{K}}{\mu}(\nabla p - g\rho\nabla z),$$

where $\mathbf{K}$ is the permeability, μ is the viscosity, g is the magnitude of the gravity, and ∇z constitutes a unit vector in the z-direction.

2.1.2 Single-Phase Heat Equation

The heat equation is found in a similar manner as the continuity equation. Instead of using the law of conservation of mass, we use the *law of conservation of energy*, which states that the energy of a control volume Ω must remain constant over time, as energy can neither be created nor destroyed (first law of thermodynamics). This is the same as

$$\begin{aligned}\text{rate of change of energy in } \Omega = {} & \text{energy transported across the boundary } \partial\Omega \\ & + \text{ energy production/consumption in } \Omega.\end{aligned}$$

The thermal energy of Ω is determined by the internal energy, U, of the fluid, and the internal energy, U_r, of the rocks. Again we use the fact that the porosity ϕ gives the percentage of the volume that contains fluid and that the density ρ gives the mass per unit of fluid, and conclude that $\phi\rho U \mathrm{d}\Omega$ gives the thermal energy of the part of the infinitesimal volume $\mathrm{d}\Omega$ that contains fluid. As the heat, unlike mass, can travel through the rocks, we also have to remember the thermal energy through the rocks. This will be given by $(1-\phi)\rho_r U_r$, as ρ_r is the rock density and $1-\phi$ gives the percentage of the volume that contains the rock (or no fluid). In the end, we have that the net thermal energy in an infinitesimal volume-element of Ω is given by $(\phi\rho U + (1-\phi)\rho_r U_r)\,\mathrm{d}\Omega$. Just as with the continuity equation, we get that the rate of change in Ω is obtained by

summing up these volume-elements, and by taking the derivative with respect to time

$$\frac{\partial}{\partial t}\int_{\Omega}(\phi\rho U + (1-\phi)\rho_r U_r)\ \mathrm{d}\Omega.$$

Energy is transported through $\partial\Omega$ by the means of the energy flux. The overall energy flux is made up of different contributions. We will focus on conduction and the flowing fluid, while contributions such as radiation will be neglected. Conduction is the transfer of energy between two points, here caused by a temperature difference between the points. It is determined by the heat flux, j, which is given by *Fick's law*. Fick's law states that $j = -\kappa\nabla T$, where κ is the thermal conductivity coefficient. The coefficient has the same function for heat as the permeability has for the fluid flow. That is, κ describes the material's ability to conduct heat, where there are more heat transfer in a material with a high thermal conductivity. In this thesis, we will for simplicity mostly use a homogeneous value for the coefficient. The thermal conductivity of granite, $\kappa = 4.0$ W/(mK), will be used in almost all experiments. Heat can also be transported by the fluid. The internal energy will follow the fluid flow and this transfer is therefore described by $\rho U\vec{v}$. The energy flux is thus found by summing up the contributions from all the elements of Ω. We are interested in the flux through the boundary of Ω, so this constitutes a surface integral,

$$-\int_{\partial\Omega}[-\kappa\nabla T + \rho U\vec{v}]\cdot\vec{n}\ \mathrm{d}S,$$

where the negative sign comes from the definition of the flux.

Energy production and consumption take place in two ways: either by external sources, or by work done on or by the system. The external sources are given by a heat source q^T. Adding the contributions from all the infinitesimal volume-elements of Ω gives that the external sources in all of Ω are found through

$$\int_{\Omega} q^T\ \mathrm{d}\Omega.$$

Considering the work in the system, we have to use the work rate, as we are working with a system that is changing. We are for brevity going to neglect gravitational work. This can be done because the dimension of the z-direction in a reservoir is small compared to the other directions. The work rate is given by force times velocity. When an element crosses a cross-section ΔA of a volume-element of Ω, the force it exerts on ΔA is determined by the pressure applied on the cross-section. This force is given by $p\Delta A\vec{n}$, and the rate of work over the cross-section is thus given by

$$-p\Delta A\vec{v}\cdot\vec{n}.$$

The negative sign comes from the fact that work done on the system is positive, and $\vec{v}\cdot\vec{n} < 0$ when the fluid is flowing into $\mathrm{d}\Omega$. The work rate in Ω is then given by summing up over the boundary of Ω,

$$-\int_{\partial\Omega} p\vec{v}\cdot\vec{n}\ \mathrm{d}S.$$

Gathering all the terms results in a new formulation of the law of conservation of energy, namely

$$\int_\Omega \frac{\partial}{\partial t}[\phi\rho U + (1-\phi)\rho_r U_r]\,\mathrm{d}\Omega = -\int_{\partial\Omega}[-\kappa\nabla T + \rho U\vec{v}]\cdot\vec{n}\mathrm{d}S + \int_\Omega q^T\,\mathrm{d}\Omega - \int_{\partial\Omega} p\vec{v}\cdot\vec{n}\,\mathrm{d}S.$$

The derivative is placed inside the integral because Ω is fixed. Using the divergence theorem, and the fact that the enthalpy is given as $H = U + \frac{p}{\rho}$, we have that

$$\int_\Omega \left[\frac{\partial}{\partial t}[\phi\rho U + (1-\phi)\rho_r U_r] + \nabla\cdot(\rho H\vec{v}) - \nabla\cdot(\kappa\nabla T) - q^T\right] = 0.$$

As this is true for every control volume Ω, the integrand must be zero, and we get that the heat equation is given by

$$\frac{\partial}{\partial t}(\phi\rho U + (1-\phi)\rho_r U_r) + \nabla\cdot(\rho H\vec{v}) - \nabla\cdot(\kappa\nabla T) = q^T.$$

2.2 MULTIPHASE FLOW

Now that we have our single-phase equations, we want to generalize to multiphase, multicomponent flow. That is because natural reservoirs generally consist of more than one fluid. In fact, there are usually multiple fluids, or phases, in the reservoir, which again consist of multiple components. The phases might include oil, water, gas, etc., while the components might be, among others, natural and dissolved gas, or methane, ethane and propane. We are going to start by deriving the equations for flow, before continuing with heat.

The mechanisms that determined the flow for the single-phase case still determine the multiphase, multicomponent flow. These mechanisms will again lead to equations describing bulk flow and heat, where the continuum hypothesis is still in place.

2.2.1 General Continuity Equation for Multiphase, Multicomponent Flow

The derivation of the multiphase, multicomponent flow equation is fairly similar to the one previously shown for single-phase flow. We still have a control volume Ω. It contains $i = 1, \cdots, M$ different components, and $j = 1, \cdots, N$ different phases. When we looked at the mass continuum law for a single phase, we said that the mass of that phase had to be conserved. It would therefore seem natural to say that the mass of each phase has to be constant over time when we extend the conservation law to multiphase flow. However, since the different components can cross over to other phases, the assumption that the mass of each phase is conserved in time does not hold true. The mass of the

components, on the other hand, must stay constant. When we find the general multiphase, multicomponent flow equations, we therefore use the conservation of mass for component i,

$$\begin{aligned}\text{rate of change of } i \text{ in } \Omega = \text{mass of } i \text{ transported across } \partial\Omega \\ + \text{ mass production/consumption of } i \text{ in } \Omega.\end{aligned}$$

To find how much mass of i there is in Ω, we need to find how much of i is in each of the N phases, and then sum up all the phases. Just as with the single-phase case, we study an infinitesimal volume-element $\mathrm{d}\Omega$ of Ω. To determine the amount of i in phase j, there are two different schools. Some, like [14, 22] use mass fractions, while others, such as [3], use partial densities and volume fractions. We are going to follow [14, 22] and use mass fractions, as I found that to be easiest to understand.

Let x_{ij} be the mass fraction of the i'th component in the j'th phase. In other words, x_{ij} tells us how much of component i is present in phase j. We have that

$$\sum_{i=1}^{M} x_{ij} = 1, \quad \forall j.$$

Furthermore, let s_j be the saturation of j, i.e., the volume fraction of phase j present in the pores of the rock, where

$$\sum_{j=i}^{N} s_j = 1.$$

We can then, with the help of the porosity, saturation and density of phase j, ρ_j, find the mass of j in the infinitesimal volume-element, namely $\phi\rho_j s_j \mathrm{d}\Omega$. The mass of component i in the part of phase j that is in $\mathrm{d}\Omega$ will thus be given by $\phi\rho_j s_j x_{ij}\ \mathrm{d}\Omega$. To find the mass of i in all of $\mathrm{d}\Omega$, and not just in the part of the volume-element that contains phase j, we have to sum up the mass of i from all the N phases, $\phi\sum_{j=1}^{N}\rho_j s_j x_{ij}\ \mathrm{d}\Omega$. The rate of change of component i in the entire volume Ω, can now be found by adding all the infinitesimal volume-elements, and by taking the derivative with respect to time:

$$\frac{\partial}{\partial t}\int_\Omega \phi\sum_{j=1}^{N}\rho_j s_j x_{ij}\ \mathrm{d}\Omega = \int_\Omega \frac{\partial}{\partial t}\left(\phi\sum_{j=1}^{N}\rho_j s_j x_{ij}\right)\ \mathrm{d}\Omega.$$

The derivative can be placed inside the integral because Ω is fixed.

To find the mass transport over $\partial\Omega$, we need to know how much of component i in phase j crosses the boundary. As the flux of phase j is given by $-\rho_j\vec{v}_j$, we have that the proportion of i in the flux is given by $-\rho_j x_{ij}\vec{v}_j$. Accumulating all the phase fluxes, that is, taking $\sum_{j=1}^{N}\rho_j x_{ij}\vec{v}_j$, gives the total flux of component i. Only the normal component of the flux crosses the surface, and we therefore have to

multiply the flux with $\vec{n}$, the normal vector. The overall transport through $\partial\Omega$ is found by adding the contributions from all surface elements of Ω, and we get

$$-\int_{\partial\Omega}\sum_{j=1}^{N}\rho_j x_{ij}\vec{v}_j\cdot\vec{n}\,\mathrm{d}S.$$

The mass production/consumption of i is, once again, given by sources. For component i, the overall production/consumption in Ω will be given by

$$\int_{\Omega}\sum_{j=1}^{N}q_i x_{ij}\,\mathrm{d}\Omega,$$

as we have to add the contribution from all the phases.

Now we just have to sum the different integrals, use the divergence theorem and use the fact that the resulting integral is true for any volume Ω. This leads to the general multiphase, multicomponent flow equations, which are of the form

$$\frac{\partial}{\partial t}\left(\phi\sum_{j=1}^{N}\rho_j s_j x_{ij}\right)+\nabla\cdot\left(\sum_{j=1}^{N}\rho_j x_{ij}\vec{v}_j\right)=\sum_{j=1}^{N}q_i x_{ij},\qquad i=1,\cdots,M. \tag{2.1}$$

These equations will have more unknowns than we are able to solve for, and we therefore need some additional equations. To get a link between the different phase pressures in the system, the *capillary pressure* is used. It is the pressure needed for the phases to invade different sized pores already filled with another fluid phase. The smaller the pores, the more capillary pressure is needed. The pressure is given as a function of the phase saturations, and gives the relationship *capillary pressure = non-wetting phase pressure - wetting phase pressure*. Which phase is the wetting phase and which is the non-wetting phase depends on the reservoir. For instance, in an oil-gas system, the oil is usually the wetting phase, while the gas is the non-wetting phase. In an oil-water system however, the water will be the wetting phase, while the oil is the non-wetting phase.

Extending Darcy's law to multiple phases is basically done by treating each phase as the single phase from Section 2.1, except we now have to keep in mind that a phase's ability to flow might be affected by the other phases. So, when we look at the multiphase version of Darcy's law, we have to add the relative permeability, which is the ratio of the effective permeability, i.e., the ability to transmit a fluid when other fluids are present, to the absolute permeability, the permeability the phase would have had if there had been only one phase present. The Darcy velocity of each phase is then given by

$$\vec{v}_j=-\mathbf{K}\frac{k_{rj}}{\mu_j}(\nabla p_j-\rho_j g\nabla z),$$

where k_{rj} is the relative permeability of phase j, μ_j is the viscosity, and p_j the phase pressure.

2.2.2 Black-Oil Equations

Even though the compositional flow model derived above accurately describes flow in porous media, it is comprehensive and not straightforward to derive and solve. We will therefore use a simplified model, namely the black-oil model. It is often used in reservoir simulation, as it is quite simple, while it at the same time manages to describe low-volatile oil systems in a good way.

The black-oil fluid model represents a reservoir where there are three components, one water component and two hydrocarbon pseudo components. The two hydrocarbon pseudo components are oil and gas, which each may contain multiple chemical species. The three components constitute three phases, one water phase containing the water component, one oil phase containing mostly the oil component and one gas phase containing the gas component. It is assumed that there is no mass transfer between the water phase and the two other phases. This means that the water phase only contains the water component. Similarly, the gas phase will only contain the gas component. The oil phase, on the other hand, contains components of oil and components of dissolved gas. The gas component can thus be transferred between the gas and oil phase.

We are here going to derive the black-oil equations for all three components and phases, but later, when we apply the numerical methods on the equations, we will neglect gas. The reason why we still include gas in the derivation of the black-oil equations is that we want our understanding of the equations to be as broad as possible.

In an attempt to make things easier, we are going to follow a common convention from the literature and denote the components with capital letters, and the phases with lowercase letters. The three components are thus W, O, G, and the three phases are w, o, g.

Using (2.1), the water component of this three-component, three-phase model is described by

$$\frac{\partial}{\partial t}\left(\phi\rho_w s_w x_{Ww}\right) + \nabla\cdot\left(\rho_w x_{Ww}\vec{v}_w\right) = q_W x_{Ww}, \tag{2.2}$$

as the only phase that contains the water component is w. Similarly, as the only phase that contains the oil component is o, the oil component is given by

$$\frac{\partial}{\partial t}\left(\phi\rho_o s_o x_{Oo}\right) + \nabla\cdot\left(\rho_o x_{Oo}\vec{v}_o\right) = q_O x_{Oo}. \tag{2.3}$$

The gas component is present both in the oil and the gas phase. The equation for the gas component will hence be given by

$$\frac{\partial}{\partial t}\left(\phi\rho_o s_o x_{Go} + \phi\rho_g s_g x_{Gg}\right) + \nabla\cdot\left(\rho_o x_{Go}\vec{v}_o + \rho_g x_{Gg}\vec{v}_g\right) = q_O x_{Go} + q_G x_{Gg}. \tag{2.4}$$

The water phase will only contain the water component, and x_{Ww} will thus equal one. Equivalently, the gas phase will only contain the gas component, and $x_{Gg} = 1$. Let W_{Oo} be the weight of the oil component in the oil phase, and W_{Go} be the weight of the gas component in the oil phase. The fraction of the oil component in the oil phase will then be given by $x_{Oo} = \frac{W_{Oo}}{W_{Oo}+W_{Go}}$, while the fraction of the gas component in the oil phase will be given by $x_{Go} = \frac{W_{Go}}{W_{Oo}+W_{Go}}$.

The black-oil equations focus on the conservation of volume, rather than the conservation of mass. It is easy to switch between the two as long as we have the volumes and densities. The gas solubility factor, R_{so} is given as the ratio of volume of the gas component in the oil phase and volume of the oil component, both measured at standard conditions, $R_{so} = V_G^s/V_O^s$. Using the weights of the oil and gas components in the oil phase from above, we have that

$$V_O^s = \frac{W_{Oo}}{\rho_O^s}, \qquad V_G^s = \frac{W_{Go}}{\rho_G^s},$$

where ρ_O^s, ρ_G^s are the oil and gas densities at standard conditions. From this, we get that

$$R_{so} = \frac{W_{Go}\rho_O^s}{W_{Oo}\rho_G^s}.$$

Furthermore, we introduce the formation factors, $B_j, j = w, o, g$, defined as the ratio of V_j to $V_i^s, i = W, O, G$. That is, $B_j = V_j/V_i^s$, where V_j is the volume of j's phase measured at reservoir conditions, and V_i^s is the volume of i's component measured at standard conditions. We have already defined V_O^s and V_G^s. Let $V_W^s = \frac{W_{Ww}}{\rho_W^s}$, $V_o = \frac{W_{Oo}+W_{Go}}{\rho_o}$, $V_g = \frac{W_{Gg}}{\rho_g}$ and $V_w = \frac{W_{Ww}}{\rho_w}$. It can then be concluded that

$$\rho_o = b_o(\rho_O^s + R_{so}\rho_G^s), \qquad \rho_g = b_g\rho_G^s, \qquad \rho_w = b_w\rho_W^s,$$

where we have introduced the inverse formation factors b_j, given by $b_j = 1/B_j$. The inverse formation factors are important quantities, as they can be used to model the densities ρ_j.

The mass fraction of the oil and gas components in the oil phase can now be found as,

$$x_{Oo} = \frac{b_o\rho_O^s}{\rho_o}, \qquad x_{Go} = \frac{b_o R_{so}\rho_G^s}{\rho_o}.$$

Using the derived densities and mass fractions in Equations (2.2) to (2.4), and by factoring out the densities given at standard conditions, the equations used to model flow in the black-oil model are given by

$$\begin{aligned}
&\frac{\partial}{\partial t}(\phi b_w s_w) + \nabla\cdot(b_w\vec{v}_w) = b_w q_w,\\
&\frac{\partial}{\partial t}(\phi b_o s_o) \ + \nabla\cdot(b_o\vec{v}_o) \ = b_o q_o,\\
&\frac{\partial}{\partial t}\left(\phi b_o R_{so} s_o + \phi b_g s_g\right) \ + \nabla\cdot\left(b_o R_{so}\vec{v}_o + b_g\vec{v}_g\right) \ = b_o R_{so} q_o + b_g q_g,
\end{aligned}$$

where the source terms are given by $q_j = q_i/\rho_j$.

To get a relationship between the different phase pressures, we will use capillary pressure. Let water be the wetting phase when we look at water and oil, and let oil be the wetting phase when we look at oil and gas. We then have that $p_o - p_w = p_{cow}$, while $p_g - p_o = p_{cgo}$. With this we have the derived multiphase flow equations, and we are now ready to continue with the multiphase expansion of the heat equation.

2.3 MULTIPHASE HEAT EQUATION

The derivation of the multiphase heat equation is similar to the derivation of the single-phase heat equation. While we for the multiphase continuity equations had to look at the different components, we can now look at the entire system, as the energy can be transported all over without caring about what kind of components are present. We do, however, have to add all the elements from the different phases. As the derivation follows that of the single-phase closely, we only give the most important details here. Just as with the single-phase, we use the law of conservation of energy:

$$\begin{aligned}\text{rate of change of energy in } \Omega = {} & \text{energy transported across the boundary } \partial\Omega \\ & + \text{ energy production/consumption in } \Omega.\end{aligned}$$

The thermal energy of the rock is again given by $(1-\phi)\rho_r U_r$. Remember from Section 2.1 that the thermal energy of the fluid in the single-phase case over an infinitesimal volume-element is given by $\phi\rho U \mathrm{d}\Omega$. It is then reasonable that the thermal energy of all the phases over an infinitesimal volume-element in the multiphase case is given by $\phi\sum_{j=1}^{N}\rho_j s_j U_j$, where U_j is the internal energy of phase j. Th, neglecting gravitational and kinetic forces, the rate of change of energy in Ω is given by

$$\frac{\partial}{\partial t}\int_\Omega \left(\phi\sum_{j=1}^{N}\rho_j s_j U_j + (1-\phi)\rho_r U_r\right)\,\mathrm{d}\Omega.$$

Energy transportation will again happen in two ways, through conduction and flow. Conduction happens through heat flux, again described by Fick's law, $-\kappa\nabla T$. Transport of energy with the flow of phase j is, just as with the single phase, given by $\rho_j U_j \vec{v}_j$, and we find the part of the thermal energy that is transported through the flow by adding all the phases in the system. Net transport across $\partial\Omega$ is thus

$$-\int_{\partial\Omega}\left(-\kappa\nabla T + \sum_{j=1}^{N}\rho_j U_j \vec{v}_j\right)\,\mathrm{d}S.$$

The energy production and consumption is given by external sources and work. We have to add all the sources from the different phases, $\int_\Omega \sum_{j=1}^{N} q_j^T\,\mathrm{d}\Omega$. The

work rate is again given by force times velocity. As all the phases have their own force and velocity, we have to add up all the different contributions. The force for a phase is given by the pressure exerted on that phase, and in the end, the work rate for the multiphase fluid is

$$-\int_{\partial\Omega}\sum_{j=1}^{N} p_j\vec{v}_j\cdot\vec{n}\,\mathrm{d}S.$$

Collecting all the different contributions, applying the divergence theorem, using the enthalpy for phase j, $H_j = U_j + \frac{p_j}{\rho_j}$, and using the fact that Ω is a random control volume, we get that the multiphase heat equation is given by

$$\frac{\partial}{\partial t}\left(\phi\sum_{j=1}^{N}\rho_j s_j U_j + (1-\phi)\rho_r U_r\right) + \nabla\cdot\left(\sum_{j=1}^{N}\rho_j H_j\vec{v}_j\right) - \nabla\cdot(\kappa\nabla T) = \sum_{j=1}^{N} q_j^T.$$

2.4 FINAL SET OF EQUATIONS

To get an overview of the equations used, a brief summary is included.

The equations used to model the flow is given by the black-oil equations. We have derived the equations for the three phases water, oil and gas, but in the rest of this thesis we are going to neglect gas. We are thus going to use the two-phase black-oil model, where the two phases and two components present are water and oil. It is assumed that the oil does not dissolve in the water phase, and that the water does not dissolve in the oil phase. To get the two-phase black-oil equations, we use the fact that the weight of the gas component is zero for all phases, $W_{Gj} = 0$. When this is true, we lose all dependency on the gas.

The conservation equations thus consist of the two-phase black-oil equations and conservation of energy, and are

$$\begin{aligned}
&\frac{\partial}{\partial t}(\phi b_w s_w) + \nabla\cdot(b_w\vec{v}_w) = b_w q_w,\\
&\frac{\partial}{\partial t}(\phi b_o s_o) \;\; + \nabla\cdot(b_o\vec{v}_o) \;\; = b_o q_o,\\
&\frac{\partial}{\partial t}\left[\phi(\rho_W^s b_w s_w U_w + \rho_O^s b_o s_o U_o) + (1-\phi)\rho_r U_r\right]\\
&\qquad + \nabla\cdot(\rho_W^s b_w\vec{v}_w H_w + \rho_O^s b_o\vec{v}_o H_o) - \nabla\cdot(\kappa\nabla T) = q_w^T + q_o^T.
\end{aligned} \tag{2.5}$$

The densities are still modeled through the inverse formation factors, but now we have that the oil density is given by $\rho_o = b_o\rho_O^s$, while we still have that $\rho_w = b_w\rho_W^s$.

The velocity of phase j is given by Darcy's law, and is

$$\begin{aligned} \vec{v}_w &= -\mathbf{K}\frac{k_{rw}}{\mu_w}\left(\nabla p_w - \rho_w g \nabla z\right), \\ \vec{v}_o &= -\mathbf{K}\frac{k_{ro}}{\mu_o}\left(\nabla p_o - \rho_o g \nabla z\right). \end{aligned} \tag{2.6}$$

The primary variables of this system are the pressure p, the temperature T and the saturation s. It is normal to solve for oil pressure and water saturation, and this will be done in this thesis as well. As most of the functions in the equations above depend on both pressure and temperature, it constitutes a non-linear system. To be able to solve it, an appropriate numerical method must be applied.

3

STANDARD DISCRETIZATION METHOD

Having found the model equations is all very well, but we cannot do anything unless we are able to solve them. The equations constitute a large non-linear system, which is not possible to solve analytically. A numerical approach must therefore be used. The standard method is to use a so-called *fully implicit solver*, where we implicitly discretize in time and use a space discretization of our choosing, before the discretized system is linearized and solved with a Newton-Raphson method. In this chapter the fully implicit method is defined, with all the discretizations and models needed to produce it. The method will later be used as a reference solution when the more efficient methods are introduced.

3.1 THE FULLY IMPLICIT METHOD

To solve the non-linear model equations (2.5) numerically, we discretize in time and space. To discretize in space, we use discrete div and grad operators defined in SINTEF's *MATLAB Reservoir Simulation Toolbox* (MRST) [15]. The operators will be discussed below. To discretize in time, we use the backward Euler method, which is an implicit method. The backward Euler method does not have a restriction on the time steps, so we are, in principle, able to use large time steps if that is desirable. In practise, however, we find that the solver breaks down when the time steps are too large. This is generally not a problem, as we are still able to use adequately large time steps when we solve the non-linear system.

If we for now disregard the discretization in space, and write the equations on residual form, we have

$$\mathcal{R} = \begin{cases} \mathcal{R}_w = 0, \\ \mathcal{R}_o = 0, \\ \mathcal{R}_T = 0, \end{cases} \tag{3.1}$$

where

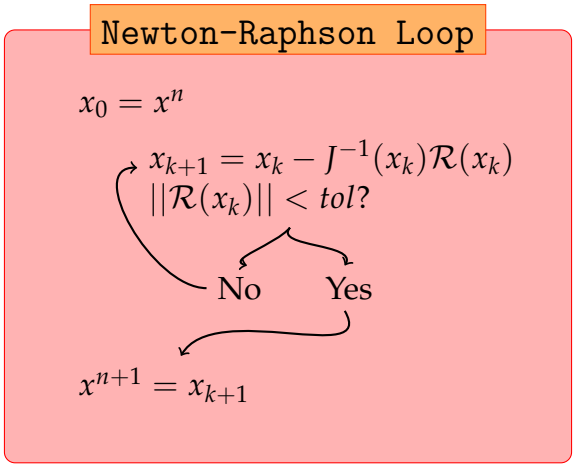
$$
\begin{aligned}
\mathcal{R}_w &= \frac{1}{\Delta t}\left[(\phi b_w s_w)^{n+1} - (\phi b_w s_w)^n\right] - \nabla\cdot(b_w\vec{v}_w)^{n+1} - (b_w q_w)^{n+1} = 0,\\
\mathcal{R}_o &= \frac{1}{\Delta t}\left[(\phi b_o s_o)^{n+1} - (\phi b_o s_o)^n\right] - \nabla\cdot(b_o\vec{v}_o)^{n+1} - (b_o q_o)^{n+1} = 0,\\
\mathcal{R}_T &= \frac{1}{\Delta t}\Big[(\phi\rho_w^s b_w s_w U_w)^{n+1} + (\phi\rho_o^s b_o s_o U_o)^{n+1} + ((1-\phi)U_r)^{n+1}\\
&\quad -(\phi\rho_w^s b_w s_w U_w)^n - (\phi\rho_o^s b_o s_o U_o)^n - ((1-\phi)U_r)^n\Big]\\
&\quad + \nabla\cdot(\rho_w^s b_w\vec{v}_w H_w + \rho_o^s b_o\vec{v}_o H_o)^{n+1} - \nabla\cdot(\kappa\nabla T)^{n+1}\\
&\quad - (q_w^T)^{n+1} - (q_o^T)^{n+1} = 0.
\end{aligned}
\tag{3.2}
$$

To find the numerical solution to (2.5) we thus have to loop through all the time steps, and for each time step we have to solve the system given by (3.1). There are many ways to solve this system, but one of the easiest is to use the *Newton-Raphson method*. Let x be the vector with the unknowns, $x = [p_o, T, s_w]^T$, where we just as easily could have solved for p_w and s_o. The Newton-Raphson method iterates over

$$x_{k+1} = x_k - J^{-1}(x_k)\mathcal{R}(x_k), \tag{3.3}$$

until the residual is below a prescribed tolerance. The matrix J denotes the Jacobian of the system. It is defined as $J = \frac{\mathrm{d}\mathcal{R}}{\mathrm{d}x}$, or $J_{ij} = \frac{\partial\mathcal{R}_i}{\partial x_j}$ on component form. Finding these derivatives analytically is difficult, and the Jacobian can be laborious to find. To sidestep this problem, we use a method known as automatic differentiation, which will be discussed below. The solution to the present time step is given by $x^{n+1} = x_{K+1}$, where K is the converged iteration. As an initial guess we use the solution from the previous time step, $x_0 = x^n$.

To solve the fully implicit system we thus follow the procedure outlined below:



3.2 WELL EQUATIONS

Partial differential equations need boundary conditions to be well posed. Oil reservoirs are, quite naturally, often made up of a closed region, where no fluid can leave unless we insert external sources that pick up the fluid. There might be aquifer in the system, but other than that, the only way to add or extract fluids is through wells. These wells will, in addition, make the phases flow through the reservoir. We have two types of wells, production and injection wells. The production wells remove the fluid from the reservoir, while the injection wells inject liquid, usually water, to increase production by forcing the fluids towards the production wells. As the wells are so important, it is crucial to handle them correctly.

The dimensions of a reservoir can be huge, and expand hundreds or thousands of meters in the lateral directions. It is therefore natural to use a spatial step size that is relatively large. The wells on the other hand, are a different matter. The radius of the wellbore, the drilled hole of the well, is given in centimeters instead of hundreds of meters. It is therefore difficult to represent the wells correctly using the cells. The wells are placed in the cells along the well pathways in the numerical solver. But, as the well is so small compared to the cells, it is difficult to relate the pressure inside the well and the pressure outside, see [31] for more. One way to solve this problem is to locally refine the cells that contain the wells. We are going to use a different approach, however, namely a well model.

In this thesis, Peacemans well model [23] will be used, where the wells are modeled as point sources. Peaceman introduced his model in 1978, and it has since become the standard model for representing wells. The sources will be determined through the phase flux, as the flow rate determines how much of the fluid reaches the wells. We assume that the fluid is immediately carried to the surface and recovered when it reaches the well. Peaceman looked at single phase flow in a two dimensional uniform grid, but his findings can be extended to multiphase flow in more complex grids.

As the wellbore is round, Peaceman used Darcy's law for radial flow as a model for the sources,

$$q = \frac{2\pi \mathbf{K} h}{\mu} \frac{p_c - p_W}{\ln(r_o/r_w)}.$$

Here, h is the formation thickness, p_W is the well pressure, r_w is the wellbore radius, and p_c is the cell pressure. Peaceman defined r_o to be the radius at which the analytical steady-state pressure p in the well is equal to the numerically calculated pressure p_c for the well cell. By doing, among other, numerical experiments where he compared the numerical and analytical single-phase

steady-state pressure on a homogeneous two dimensional uniform grid with different grid sizes and spacing Δx, he found that

$$r_0 = 0.2\Delta x.$$

We can thus define the Peaceman well index,

$$W_I = \frac{2\pi h}{\ln(0.2\Delta x / r_w)},$$

which relates the flow rate q to the pressure drawdown, the difference between the reservoir pressure and the well pressure, $p_c - p_W$

$$q = W_I \lambda (p_c - p_W).$$

Anyone familiar with Peaceman's well index knows that our W_I is slightly modified from the original well index. Peaceman included the mobility in the index, but we have let the mobility be an individual factor in the equation for the flow rate. This way, we can use the same well index for the different phases when we extend the model to multiphase flow. The expansion is done by using the fact that the flow rate of each phase in each well is given by Peaceman's model. This gives our well-model,

$$q_j = W_I \lambda_j (p_c - p_W)$$

where $j = o, w$.

For the heat equation to be well posed we need to define the temperature source terms, q_j^T. To find them, we use the sources defined above, q_j. If we are looking at the injection wells, external heat will flow from the wells throughout the reservoir with the flow rate q_j. This external heat will depend on the temperature in the wells. We therefore have to multiply the enthalpy of the well, calculated with the well temperature and pressure, with the flow rate from the well, as this is the amount of heat that will flow from the wells. If we are looking at a production well, the source terms will carry the heat up the wells, and the temperature source terms are thus determined by multiplying the enthalpy calculated with the pressure and temperature from the cells defining the wells, with the source terms defined above. The temperature source terms are thus given by

$$q_j^T = \begin{cases} H_j(p_W, T_W) q_j & \text{if injection well,} \\ H_j(p_c, T_c) q_j & \text{if production well,} \end{cases}$$

where T_W is the temperature in the well, and T_c is the temperature in the well cell.

If wells are not monitored and controlled, blowouts can occur. It can lead to damage of equipment, and even loss of life. It is therefore important to have mechanisms that control the well, to prevent blowouts from happening. Wells

are usually controlled by the bottom-hole pressure or the flow rate in the wells. This will therefore also be enforced when we model the wells. We can have different controls on different wells. The bottom-hole pressure is the sum of all the pressures acting on the bottom of the wellbore, and by controlling this quantity we are able to control, among other, the hydrostatic pressure drop which might cause the blowouts. To enforce the controls we add extra control equations to our overall system. A constant value, $C_{bh,W}$ or $C_{q,W}$, depending on whether we have bottom-hole or flow rate control, is prescribed per well. Per well, the control equation will then be of the form

$$\mathcal{R}_C = p_{bh} - C_{bh}, \qquad \text{or} \qquad \mathcal{R}_C = q - C_q,$$

where p_{bh} is the bottom hole pressure and q is the net flow in the well. We also check that the surface rates for each phase, q_j^s, equals the flow rate for the phase in the well,

$$\mathcal{R}_{q,j} = q_j^s - \sum_{c \in C_W} q_j[c] = 0,$$

where C_W is the set of cells containing the well.

Our overall system will now consist of the equations on residual form, (3.1), where the well source terms have been added to the equations, and the control equations. As the source terms only contribute to the system in the grid cells defining the wells, we only need to worry about the source terms in these cells. It is therefore natural to first find the main part of the equations for the whole reservoir, and then subtract the sources in the respective cells. But one can, of course, do as one pleases. In the end, the system to be solved will look like

$$0 = \mathcal{R} = [\mathcal{R}_w, \mathcal{R}_o, \mathcal{R}_T, \mathcal{R}_{q,w}, \mathcal{R}_{q,o}, \mathcal{R}_C]^T,$$

where we solve for $x = [p_o, T, s_w, q_w^s, q_o^s, p_{bh}]^T$.

3.3 MATLAB RESERVOIR SIMULATION TOOLBOX

To aid in the numerics, we are, as previously stated, going to use SINTEF's *MATLAB Reservoir Simulation Toolbox* (MRST). We will here go through the MRST functions and operators that will be used in our simulations. MRST is a toolbox designed to simulate various models, for instance flow in porous media. It uses rapid prototyping, a process that builds the model level by level. MRST consists of two parts, the *core*, containing the methods that are unlikely to change over the years, and various *add-on modules*, containing functions and methods subject to change with new research. We will use both parts. See [15, 13, 26] for a thorough introduction to MRST.

MRST's methods are developed to work on different kinds of grids. The grids defined through the toolbox are therefore expressed in the same way, through a class structure. The grids are divided into different cells, and the boundaries of each cell are divided into faces. See Figure 3.1 for an illustration. The grid class gives certain information about the grid, for instance the number of cells, the cell positions and the faces. MRST has also developed tools to make it easy to calculate, among others, the volume and centroids of the cells. The class structure ensures that the grids easily can be switched when a solver is used.

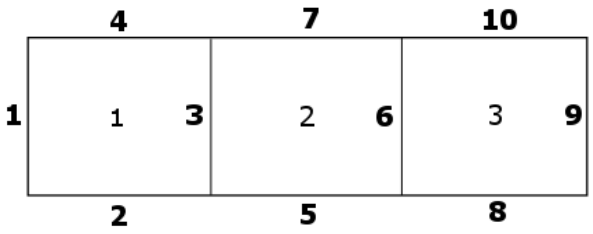


Figure 3.1: Simple grid with faces given in bold numbering, while the cell numbering is placed in the middle of the cell.

Space Discretization

We have already shown how to discretize the equations in time. To discretize in space, we use vector functions defined in MRST. The operators we have to discretize are the divergence and gradient operators. Let f be a face shared by cells $N_1(f)$ and $N_2(f)$. Thus, if we are looking at face number 6, which is shared by cell number 2 and 3, then $N_1(6) = 2, N_2(6) = 3$, see Figure 3.1. MRST's grad operator looks at the difference over the face, so if we have an arbitrary expression of the form ∇h, we will with MRST use

$$[\nabla h]_f \approx \mathsf{grad}(h)[f] = h[N_2(f)] - h[N_1(f)].$$

MRST's div operator, on the other hand, looks at the cells. Let c be the cell in question. The div operator is found by adding up the contributions from all the faces belonging to the cell,

$$[\nabla \cdot h]_c \approx \mathsf{div}(h)[c] = \sum_{f \in f(c)} \mathrm{sgn}(f) h[f],$$

where

$$\mathrm{sgn}(f) = \begin{cases} 1 & \text{if } c = N_1(f), \\ -1 & \text{if } c = N_2(f). \end{cases}$$

To get a better understanding, we refer to Figure 3.1. The faces are given in bold numbers. If we look at the gradient over face **6**, it will be given by the cells belonging to the face, $\mathsf{grad}(h)[\mathbf{6}] = h[3] - h[2]$. The divergence of cell 2, on the other hand, will be given by the faces belonging to the cell, $\mathsf{div}(h)[2] = h[\mathbf{6}] + h[\mathbf{7}] - h[\mathbf{3}] - h[\mathbf{5}]$.

Automatic Differentiation

As a way to avoid laborious computations when finding various derivatives, we use, as previously mentioned, *automatic differentiation* (AD). It is, as the name applies, a method that finds the derivatives automatically. This section explains how MRST performs AD. The values of functions and their corresponding derivatives are updated simultaneously in MATLAB. AD takes advantage of the fact that derivation follows simple rules, and as long as we follow these rules we can differentiate complex expressions. Rules like addition, the product rule for multiplication and the chain rule form the foundation for derivation of all functions, and these rules are not complex, even though the function in itself might be.

When MATLAB applies an operator on a function, it determines the kind of function it is dealing with before applying the operator. Multiplication between matrices is for instance different than multiplication between two scalars. The two types of multiplication still use the same notation though, namely $*$, or else MATLAB would be a confusing language to use. To be able to have operators that change depending on the type of function, MATLAB uses a class system. When implementing AD, these ideas are developed further. A class system is used, where the function and derivatives are updated simultaneously, the function with the help of the original MATLAB operators, and the derivatives with the derivative rules. So, when MATLAB encounters multiplication, it has to apply the correct form of multiplication on the functions, and it must use the product rule on the derivatives.

To see how the functions and derivatives can be updated simultaneously, let us look at two variables, x and y. We know that $\frac{\partial x}{\partial x} = 1$ and $\frac{\partial x}{\partial y} = 0$. We will thus have that

$$x \to x, 1, 0 \qquad \text{in MATLAB,}$$

where the first variable is the normal function, the second is the derivative with respect to x and the last is the derivative with respect to y. One could always use more or less variables of course. With the chain rule, we get that

$$(3xy)^2 \to (3xy)^2, 2 * 3xy * 3y, 2 * 3xy * 3x \qquad \text{in MATLAB.}$$

If the program also knows a bit more complex rules, like the fact that $\frac{\partial \cos(x)}{\partial x} = -\sin(x)$, we can update complex functions with AD:

$$\begin{aligned}(3xy)^2 + \cos(x^2y^4)\sin(3x^5) \to\ & (3xy)^2 + \cos(x^2y^4)\sin(3x^5),\\ & 2 * 3xy * 3y - 2xy^4 \sin(x^2y^4)\sin(3x^5) + 5 * 3x^4 \cos(x^3y^4)\cos(3x^5),\\ & 2 * 3xy * 3x - 4x^2y^3 \sin(x^2y^4)\sin(3x^5).\end{aligned}$$

To see how this will look in MATLAB, we can look at the expression $z = xy^2$ for the values $x = 3$ and $y = 4$. Notice that $\frac{\partial}{\partial x}x\Big|_{x=3,y=4} = 1$, $\frac{\partial}{\partial y}x\Big|_{x=3,y=4} =$

$0, \quad z|_{x=3,y=4} = 48$ and $\frac{\partial}{\partial x} z\Big|_{x=3,y=4} = 16.$

If we now want to apply AD, we simply type

```
[x, y] = initVariables(3, 4);
z = x * y.^2;
```

in MATLAB. This gives:

```
x = ADI with properties:
  val : 3
  jac : {[1]  [0]}

y = ADI with properties:
  val : 4
  jac : {[0]  [1]}

z = ADI with properties:
  val : 48
  jac : {[16]  [24]}
```

which correspond to the correct values and derivatives.

To see how we can find the Jacobian of a system of equations in MATLAB, which is what we will use AD for, we look at $f(x, y) = (xy^2, 3x + y)$, when $x = 3$ and $y = 4$. Again we start with the commands

```
[x, y] = initVariables(3, 4);
z1 = x * y.^2;
z2 = 3 * x + y;
```

To bound them together in a system, we simply type

```
eqs = {z1, z2};
eq = cat(eqs{:});
```

and the Jacobian of the system is now automatically given by typing `eq.jac{1}`

```
full(eq.jac{1}) =
    16    24
     3     1
```

Upwind Discretization

The flow of phase j over cell face f will in large scale depend on the relative permeability k_{rj}. It will also depend on the inverse formation volume factor b_j, as b_j helps model j's density. To deal with k_{rj}, we have to look at the mobility, as $\lambda_j = k_{rj}/\mu_j$. As a result, to properly handle the flow, λ_j and b_j must be properly handled in our simulations.

Let us look at the flow through f, from cell i to j. The mobilities and the inverse formation factors need to reflect the flow over the face. When we have reached

residual saturation for a phase in i, the mobility needs to be zero as flow from i has stopped. This is implemented through an *upwind selection*, where we will use i's λ if the velocity over f is greater than zero. This is to reflect the fact that we still have flow over the face. If this is not the case, however, we will use j's λ. The same holds true for the inverse formation volume factors. In MATLAB and MRST, this is implemented as follows,

$$\mathsf{upw}(h)[f] = \begin{cases} h[N_1(f)], & \text{if } v[f] > 0, \\ h[N_2(f)], & \text{otherwise,} \end{cases}$$

where h is some random function or variable.

Now that we have gone through all the discretization methods, it is fairly straightforward to find the discretized equations. For instance, the discrete oil equation from (3.2), will in MATLAB look like

$$\frac{1}{\Delta t}\left[(\phi b_o s_o)^{n+1} - (\phi b_o s_o)^n\right] - \mathsf{div}(\mathsf{upw}(b_o)v_o)^{n+1} - (b_o q_o)^{n+1} = 0,$$

where

$$v_o = -\mathsf{upw}(\lambda_o)T_p\left(\mathsf{grad}(p) - g\mathsf{avg}(\rho_o)\mathsf{grad}(z)\right).$$

We have used the arithmetic average of the density, $\mathsf{avg}(\rho_o)$. As a model for the permeability, we use transmissibilities, which is a measure of how much the phase can be transported through the medium. The discrete grad operator introduced above will only give the exact derivative for certain grids. For general grids, however, the operator will need additional factors which include the geometry of the cells in order to give a good approximation for the derivative. This is obtained through the transmissibilities, which use, among other, the cell geometry to be computed. By using the transmissibility as well as the grad operator, we get a better approximation for the derivatives than if we had just used the grad operator. In the equation above, the transmissibility is given by T_p, the permeability transmissibility. The thermal conductivity will also be given by transmissibilities, T_T. The different transmissibilities can be found in many different ways, for instance through the finite volume method.

4

SEQUENTIAL FORMULATION

The fully implicit method is a good method in that it is robust and comprehensible. A problem with the method, however, is the fact that the system can become very big. It will then use a lot of computer memory, and the computation time might be long. Another drawback is that it is sometimes desirable to apply different numerical methods on the different variables in the system. The equations exhibit various characters, and as different numerical methods work on different types of equations, we could apply distinct methods on the separate parts if everything was not all in one system.

To account for these problems, we are going to use a splitting method, which we in the next chapter will augment by applying a multiscale method. As the name suggests, the splitting method splits the fully implicit system, making it possible to solve for one variable at the time. Several different splitting methods have been introduced, such as IMPES, see for instance [25, 2], explicitly coupled methods [21, 32] and sequential splitting, see for instance [29, 28]. We will use the latter. These methods usually look at isothermal models, but they are fairly straightforward to extend to thermal models. We will here explain the structure of the isothermal methods, before describing how we proceed when we add temperature. The methods start out by decoupling the equations in the fully implicit system. Keeping the saturations fixed, usually using the values from the previous time step, the methods solve for pressure using a pressure equation found through the decoupled equations. When the pressure has been found, saturation is solved for, using special saturation equations where the newly found pressure is used as the pressure variable. While the fully implicit method uses an implicit solver for the whole system, IMPES uses an implicit method to solve for pressure, and an explicit method to solve for the saturations. Explicit solvers are only conditionally stable, so IMPES requires relatively small time steps when it solves for the saturations. This might be restrictive, for instance when we want to simulate very long sequences. We are therefore going to use a slightly different approach. Our sequential splitting method will use an implicit solver for all equations, to ensure that we are always able to use whatever time step we want. We will decouple the fully implicit system into three systems, one system that solves a pressure equation, one that solves a temperature equation,

and one that solves a transport equation. We will thus first solve for pressure, using the pressure equation, keeping the temperature and saturations from the last time step fixed. Next, we will solve for temperature, using the newly found pressure, and the saturations from the last time step. Lastly, we use the net flux to express the pressure gradient in Darcy's law, and use this to update the saturations. The pressure and temperature variables are the updated pressure and temperature from before. The sequential splitting method will thus be of the form

$$\text{pressure} \rightarrow \text{temperature} \rightarrow \text{transport}.$$

4.1 PRESSURE EQUATION

In order to find the equation used to find the pressure, i.e., the pressure equation, we use the semi-discretized equations defined by (3.2). To discretize in space, we use the div and grad operators defined in Section 3.3. As we are only solving for one variable, we need to find one equation that is independent of the temperature and saturations of the next time step, T^{n+1}, s^{n+1}. All three equations in (3.2) depend on pressure, so we want our pressure equation to be a combination of the three equations. We therefore assume that there are factors β_w, β_o and β_T such that

$$\widetilde{\mathcal{R}}_p = \beta_w \mathcal{R}_w + \beta_o \mathcal{R}_o + \beta_T \mathcal{R}_T = 0$$

is independent of T^{n+1} and s^{n+1}. Notice that $\widetilde{\mathcal{R}}_p$ equals zero, as $\mathcal{R}_w, \mathcal{R}_o$ and $\mathcal{R}_T$ all equal zero. Now, choosing the factors

$$\beta_w = \frac{1}{b_w^{n+1}}, \qquad \beta_o = \frac{1}{b_o^{n+1}}, \qquad \beta_T = 0,$$

lead to the pressure equation

$$\begin{aligned}\widetilde{\mathcal{R}}_p =& \frac{\phi^{n+1}}{\Delta t} - \frac{\phi^n}{\Delta t}\left[\frac{(b_w s_w)^n}{b_w^{n+1/3}} + \frac{(b_o s_o)^n}{b_o^{n+1/3}}\right] \\ &- \frac{\nabla \cdot (b_w \vec{v}_w)^{n+1/3}}{b_w^{n+1/3}} - \frac{\nabla \cdot (b_o \vec{v}_o)^{n+1/3}}{b_o^{n+1/3}} - q_w^{n+1} - q_o^{n+1} = 0.\end{aligned}$$

The velocities and the inverse formation factors are now taken at time step $n + 1/3$, as the equation only is independent of temperature and saturations of next time step when $b_w^{n+1/3} = b_w(p^{n+1}, T^n)$, $b_o^{n+1/3} = b_o(p^{n+1}, T^n)$ and

$$\begin{aligned}\vec{v}_w^{n+1/3} &= -\mathbf{K}\frac{k_{rw}(s_w^n)}{\mu_w(p^{n+1}, T^n)}\left(\nabla p_w^{n+1} - \rho_w g \nabla z\right), \\ \vec{v}_o^{n+1/3} &= -\mathbf{K}\frac{k_{ro}(s_o^n)}{\mu_o(p^{n+1}, T^n)}\left(\nabla p_o^{n+1} - \rho_o g \nabla z\right).\end{aligned}$$

The time step $n+1/3$ signifies the fact that there are three steps in the sequential algorithm, where this first step uses values that will be updated in step two and three. The porosity and source terms are still taken at time step $n+1$, because these are only dependent on pressure. That is, $\phi^{n+1} = \phi(p^{n+1}), q_j^{n+1} = q_j(p^{n+1})$.

The control equations for the wells are also incorporated into this system. The equations are calculated just as in Section 3.2, and the pressure system is then expanded to include the equations,

$$0 = \mathcal{R}_p = [\widetilde{\mathcal{R}}_p, \mathcal{R}_{q,w}, \mathcal{R}_{q,o}, \mathcal{R}_C]^T.$$

We can now solve for p_o^{n+1} (or p_w^{n+1} if that is desirable, as $p_o - p_w = p_{cow}$) by using $\mathcal{R}_p$ and the Newton-Raphson method, where the primary variables are $[p_o, q_w^s, q_o^s, p_{bh}]^T$.

As we only have one equation that describes the pressure, we can for practical purposes treat it as a single-phase equation with constant temperature. After all, we are only solving for one phase pressure, and the saturations and temperature are constants. This is in many ways an advantage, as we can now employ methods designed to work on single phase flow. This can speed up the numerical process.

4.2 TEMPERATURE EQUATION

We get the temperature equation by taking $\mathcal{R}_T$ from (3.2), using the newly updated pressure, and saturations from the last time step. That is, every time (3.2) use p^{n+1} and s^{n+1}, we are now going to use $\bar{p}$ and s^n, where we have called the updated pressure $\bar{p}$ in an attempt to make things simpler. The functions that use a combination of $\bar{p}, T^{n+1}$and s^n will be denoted $n+2/3$, as this is the second step in the sequential algorithm. As an example, we can look at the inverse formation factors. In (3.2) we use the formation factors from the next time step, $b_j^{n+1} = b_j(p^{n+1}, T^{n+1})$, as well as $b_j^n = b_j(p^n, T^n)$. Here, the formation factor from the next time step will be replaced with $b_j^{n+2/3} = b_j(\bar{p}, T^{n+1})$, while we keep the formation factor from the last time step as it is. Doing this with all functions that include p^{n+1} and s^{n+1}, leads to a temperature equation that is independent of pressure and saturation of the next time step. Keeping all this in mind, we find that the temperature equation equals

$$\begin{aligned}\mathcal{R}_T = \frac{1}{\Delta t}\Big[&s_w^n \cdot (\rho_W^s \phi b_w U_w)^{n+2/3} + s_o^n \cdot (\rho_O^s \phi b_o U_o)^{n+2/3} + ((1-\phi)U_r)^{n+2/3} \\ &-(\phi\rho_W^s b_w s_w U_w)^n - (\phi\rho_O^s b_o s_o U_o)^n - ((1-\phi)U_r)^n\Big] \\ &+ \nabla\cdot(\rho_W^s b_w \vec{v}_w H_w + \rho_O^s b_o \vec{v}_o H_o)^{n+2/3} \\ &- \nabla\cdot(\kappa\nabla T)^{n+1} - (q_w^T)^{n+2/3} - (q_o^T)^{n+2/3} = 0.\end{aligned}$$

As ρ_i^s is a constant, it will not need to be updated. It is shown here with a time step just to try to make the temperature equation easier to read. The conduction term is denoted with the time step $n+1$ because it is only dependent on temperature. When a function has time step n it means that all its variables are taken from the last time step. Furthermore, we have that

$$\begin{aligned}
\phi^{n+2/3} &= \phi(\bar{p}), \\
U_j^{n+2/3} &= U_j(\bar{p}, T^{n+1}), \\
U_r^{n+2/3} &= U_r(\bar{p}, T^{n+1}), \\
H_j^{n+2/3} &= H_j(\bar{p}, T^{n+1}), \\
v_j^{n+2/3} &= \vec{v}_j(\bar{p}, T^{n+1}, s^n).
\end{aligned}$$

The only unknown in the temperature equation is therefore T^{n+1}, which is the entity we solve for.

Backward Euler has been used to discretize time. The velocity is very similar to the velocity used for the pressure equation, but now we use T^{n+1} (and $\bar{p}$) instead;

$$\begin{aligned}
\vec{v}_w^{n+2/3} &= -\mathbf{K}\frac{k_{rw}(s_w^n)}{\mu_w(\bar{p}, T^{n+1})}\left(\nabla \bar{p}_w - \rho_w g \nabla z\right), \\
\vec{v}_o^{n+2/3} &= -\mathbf{K}\frac{k_{ro}(s_o^n)}{\mu_o(\bar{p}, T^{n+1})}\left(\nabla \bar{p}_o - \rho_o g \nabla z\right).
\end{aligned}$$

Just as with the pressure equation, the temperature equation is solved with the Newton-Raphson method. Again we have an equation that mimics a single-phase equation, which enables us to apply appropriate numerical methods.

4.3 TRANSPORT EQUATION

To be able to update the saturations, we first look at the total flux $\vec{v}_T = \vec{v}_w + \vec{v}_o$, where $\vec{v}_w$ and $\vec{v}_o$ are given in (2.6). This is done in order to get a fractional formulation for the Darcy velocities, making it easier to solve for saturation. By looking at the total flux we find an expression for ∇p_o, which we can reinsert into the expression for the phase fluxes. This way, we get phase fluxes that depend on the total velocity instead of on the pressure gradient. Remembering that $p_o - p_w = p_{cow}$, we have that the total flux is given by

$$\vec{v}_T = \vec{v}_o + \vec{v}_w = -\mathbf{K}\frac{k_{ro}}{\mu_o}\left(\nabla p_o - g\rho_o \nabla z\right) - \mathbf{K}\frac{k_{rw}}{\mu_w}\left(\nabla(p_o - p_{cow}) - g\rho_w \nabla z\right).$$

Which, with the mobilities $\lambda_o = \frac{k_{ro}}{\mu_o}$ and $\lambda_w = \frac{k_{rw}}{\mu_w}$, yields

$$\nabla p_o = -\frac{\vec{v}_T}{\mathbf{K}(\lambda_o + \lambda_w)} + \frac{g\nabla z}{\lambda_o + \lambda_w}(\lambda_o \rho_o + \lambda_w \rho_w) + \frac{\lambda_w}{\lambda_o + \lambda_w}\nabla p_{cow}.$$

Now, by reintroducing this expression into Darcy's law (2.6), and presenting the fractional flow $f_o = \frac{\lambda_o}{\lambda_o+\lambda_w}$, we find that the oil phase flux is given by

$$\begin{aligned}
\vec{v}_o &= -\mathbf{K}\frac{k_{ro}}{\mu_o}\left(\nabla p_o - g\rho_o\nabla z\right) \\
&= -\mathbf{K}\lambda_o\left[-\frac{\vec{v}_T}{\mathbf{K}(\lambda_o+\lambda_w)} + \frac{g\nabla z}{\lambda_o+\lambda_w}\left(\lambda_o\rho_o+\lambda_w\rho_w\right) + \frac{\lambda_w}{\lambda_o+\lambda_w}\nabla p_{cow} - g\rho_o\nabla z\right] \\
&= \frac{\lambda_o}{\lambda_o+\lambda_w}\left[\vec{v}_T + \mathbf{K}(-g\nabla z(\lambda_o\rho_o+\lambda_w\rho_w) - \lambda_w\nabla p_{cow} + (\lambda_o+\lambda_w)g\rho_o\nabla z)\right] \\
&= f_o\left[\vec{v}_T + \mathbf{K}\lambda_w(\rho_o g\nabla z - \rho_w g\nabla z - \nabla p_{cow})\right].
\end{aligned}$$

Following the same procedures yields the water phase flux

$$\vec{v}_w = f_w\left[\vec{v}_T - \mathbf{K}\lambda_o(\rho_o g\nabla z - \rho_w g\nabla z - \nabla p_{cow})\right],$$

where we instead of fractional flow for oil use the fractional flow for water, $f_w = \frac{\lambda_w}{\lambda_o+\lambda_w}$.

We thus have expressions for the fluxes that depend on the total flux v_T, instead of depending on pressure. The total velocity is held constant throughout the transport simulation, and is found by adding the fluxes after having solved for pressure. By keeping v_T fixed, we will have phase fluxes where the saturation is the only unknown, as the variables that originally depend on pressure use the already updated pressure. The phase fluxes at time $n+1$ used to update the saturations are thus

$$\begin{aligned}
\vec{v}_o^{n+1} &= f_o^{n+1}\left[\vec{v}_T + \mathbf{K}\lambda_w^{n+1}(\rho_o g\nabla z - \rho_w g\nabla z - \nabla p_{cow}^{n+1})\right], \\
\vec{v}_w^{n+1} &= f_w^{n+1}\left[\vec{v}_T - \mathbf{K}\lambda_o^{n+1}(\rho_o g\nabla z - \rho_w g\nabla z - \nabla p_{cow}^{n+1})\right].
\end{aligned}$$

These expressions are used in one of the conservation equations, $\mathcal{R}_w$, $\mathcal{R}_o$ or $\mathcal{R}_T$, from (3.2), to find the saturations. As we have already found p^{n+1} and T^{n+1}, and as $s_o + s_w = 1$, we would overdetermine the system if more than one equation were to be used. Furthermore, as $\mathcal{R}_T$ was used as the temperature equation, the method works best if we choose $\mathcal{R}_w$ or $\mathcal{R}_o$ to find the saturations. We have mostly used $\mathcal{R}_o$ in this thesis.

To discretize the equation in time, the backward Euler method is once again used to prevent the dependency on small enough time steps. To solve the transport equation we use the Newton-Raphson method.

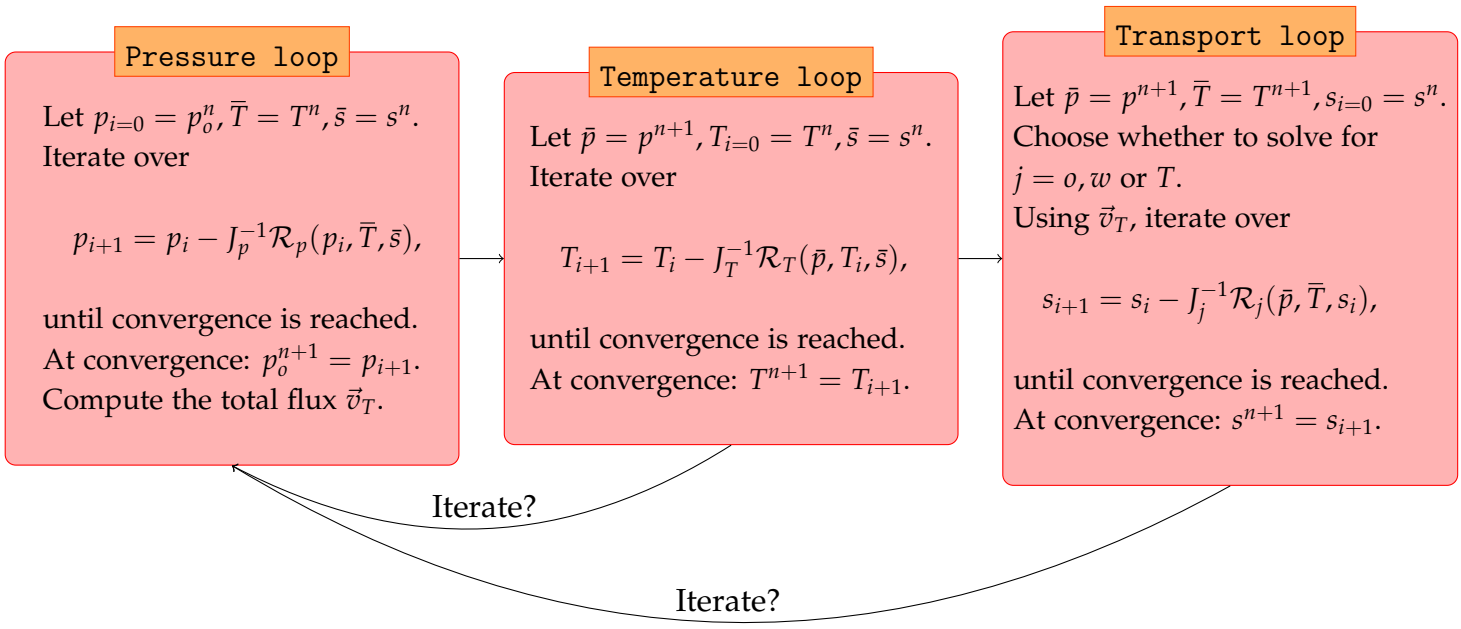
4.4 SOLVING THE SEQUENTIAL SYSTEM

There are a lot of different components to consider when solving the three nonlinear systems arising from the sequential formulation. We have therefore pro-

vided a flowchart to get a better overview. Each subsystem in the overall sequential system is at each time step solved using the Newton-Raphson method, (3.3).

The sequential and fully implicit solution will differ to a certain extent. But, the smaller the time steps, the closer the solutions will be. This makes it easy to check if the sequential solution is wrong, because if the two solutions do not converge as the time steps decrease, something is wrong. It is important to know that the sequential system is implemented correctly, because we are later going to apply a multiscale method on the sequential system. To make sure the multiscale method is correct, the sequential system will be used as a reference solution. The solutions should in principle also converge if we apply outer iterations in the sequential method, as it will drive the residual towards zero. That is, we can place a loop on the pressure and temperature system so that after the temperature system has converged we go back to the pressure system and resolve it. We then proceed to the temperature system and resolve that. This is done n_1 times. We can also place a loop after the transport equation has converged, so that after having found the saturations we go back and resolve the pressure and temperature, keeping the n_1 iterations if that is desirable, and then proceed to resolve the transport system. This is done n_2 times, where n_1 and n_2 are small. Instead of a fixed number of iterations n_1, n_2, we could also check whether we have reached a preset tolerance, and iterate if it is not the case. If we apply the outer iterations after the temperature has converged, or at the end of each time step, the overall fine scale residual should decrease.

In the end, the three steps executed each time step in the sequential algorithm are:



Here, J_p is the Jacobian of $\mathcal{R}_p$, J_w is the Jacobian of $\mathcal{R}_w$, J_o is the Jacobian of $\mathcal{R}_o$ and J_T is the Jacobian of $\mathcal{R}_T$. The Jacobians are found using automatic differentiation.

4.5 EXAMPLES

To check the correctness of the sequential method, we are going to compare the solver to the fully implicit solver, checking whether the discrepancies between the methods are small. We present two examples, which both model two-phase flow. The first example is rather simple. It has a homogeneous permeability field, and is included as a first test to check that everything is implemented correctly. The second example has a more complex permeability field, making it more difficult to solve correctly. Both examples have water and oil as their two phases.

To check that the sequential method is correct, we consider the discrepancies between the fully implicit solutions and the sequential solutions. The discrepancies are measured in L^2 and L^∞ norms, and are

$$\begin{aligned} e^2 &= \frac{||x_{FI} - x_S||_2}{||x_{FI}||_2} = \frac{\left(\sum_{i=1}^{n} |x_{FI,i} - x_{S,i}|^2 |\Omega_i|\right)^{1/2}}{\left(\sum_{i=1}^{n} |x_{FI,i}|^2 |\Omega_i|\right)^{1/2}}, \\ e^\infty &= \frac{||x_{FI} - x_S||_\infty}{||x_{FI}||_\infty} = \frac{\max_{i=1,\cdots,n} |x_{FI,i} - x_{S,i}|}{\max_{i=1,\cdots,n} |x_{FI,i}|}, \end{aligned} \tag{4.1}$$

where x_{FI} is the fully implicit solution, x_S is the sequential solution, $|\Omega_i|$ gives the volume of cell i and n is the number of fine cells in the system.

The two examples use some of the same values and functions. The thermal conductivity will be that of granite, so that $\kappa = 4.0$ W/(mK). Furthermore, the initial temperature inside the wells is set to be 300K. As for the different functions needed in the heat equation, we have that the enthalpy for phase j will be given by

$$H_j = c_r \cdot T + (1 - c_h \cdot T_{\text{ref}}) \frac{p - p_{\text{ref}}}{b_j \rho_j^s}, \tag{4.2}$$

where $c_r = 2.17 \cdot 10^6, T_{\text{ref}} = 310$ K, $c_h = 10^{-4}, p_{\text{ref}} = 300$ bar, $\rho_W^s = 1000$ and $\rho_O^s = 700$. The internal energy of the rocks is given by

$$U_r = c_r T, \tag{4.3}$$

while the internal energy is given by

$$U_j = H_j - \frac{p}{b_j \rho_j^s}.$$

Lastly, the inverse formation volume factors are given by

$$b_j = \frac{\rho_r}{\rho_j^s}\left(1 + c_f(p - p_{\text{ref}})\right) e^{-c_h(T - T_{\text{ref}})},$$

where $\rho_r = 850\ \text{kg/m}^3$, and $c_f = 10^{-3}\ \text{bar}^{-1}$. These functions are similar to what was used in [13], though that paper looked at single-phase flow.

Machinery

The following tests were performed on an Ubuntu 12.04 LTS 64-bit OS, with an Intel Core i7-2600 CPU processor, having 7.8 GiB memory. The tests were executed using MATLAB 2013b. SINTEF's *MATLAB Reservoir Simulation Toolbox* (MRST) was in addition used heavily. The newest release, version 2015b, was released in December 2015. It can be downloaded from `http://www.sintef.no/projectweb/mrst/downloadable-resources/download/`. See [15] for more information about MRST.

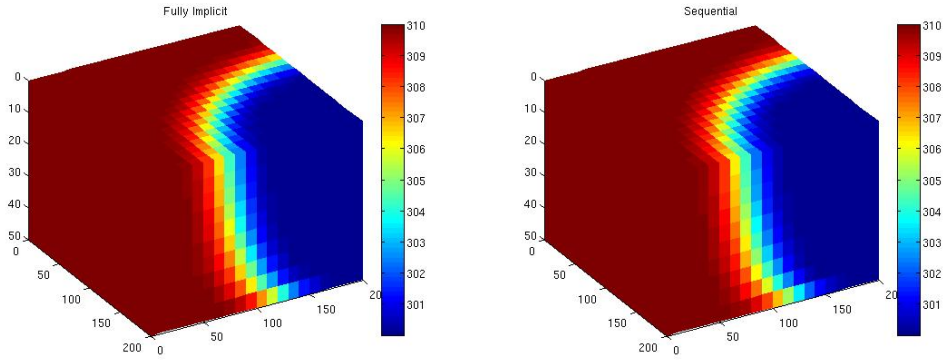
4.5.1 Test Case: Homogeneous Permeability

We start by presenting a test case consisting of a $20 \times 20 \times 5$ Cartesian grid, or 4000 fine cells, modeling a $200 \times 200 \times 50\ \text{m}^3$ two-phase reservoir with homogeneous permeability of 0.030 Darcy, where the two phases are oil and water. The model has two wells, one production well and one injection well. They are situated in opposite corners, and will make the fluids flow through the reservoir. The producer well is controlled by a bottom-hole pressure of 4000 psia, and the injector is controlled by the surface rate.

The discrepancies in the temperature and pressure at times $t = 5, 100, 200$ days, measured in the norms defined in (4.1), are given in Table 4.1. A time step of $\Delta t = 5$ days was used, and the simulation ran for 40 time steps, or 200 days. The tolerance of the fully implicit solver was $tol = 10^{-6}$, as was the tolerance for the pressure, temperature and transport equation in the sequential solver. As can be seen from the table, the accuracy of the sequential model is good. The first pressure solution, $t = 5$ days, has a little high discrepancy, but it decreases as the simulation proceeds. This error is most pronounced in the supremum norm. The difference between the two solutions occur initially at the injection well, before the discrepancy decreases and spreads a little. The temperature solutions for the last time step ($t = 200$ days) for both the fully implicit system and the sequential system are given in Figure 4.1. The figure is in line with the error findings, and the two solutions are identical to the naked eye.

Table 4.1: Temperature and pressure discrepancy for the test case, for $t = 5, 100, 200$ days. The discrepancy in temperature is given by e_T, while the pressure discrepancy is given by e_p.

	Time (days)	L^2	L^∞
e_T	t = 5	0.00024	0.00295
	t = 100	0.00041	0.00189
	t = 200	0.00053	0.00197
e_p	t = 5	0.08699	1.42820
	t = 100	0.00387	0.00927
	t = 200	0.00149	0.00344

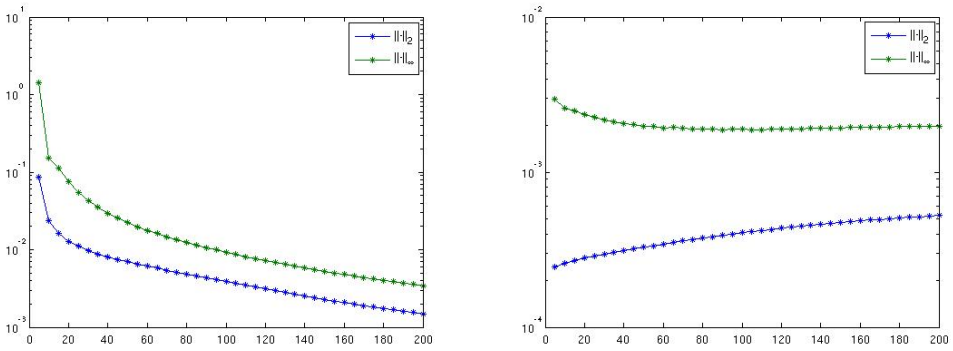


(a) Fully implicit (b) Sequential

Figure 4.1: Fully implicit and sequential temperature solutions for the final time step, $t = 200$ days, in the test case simulation.

The pressure and temperature discrepancy as a function of time is given in Figure 4.2. The same parameters as above are used. Again we see that the two solvers match, because the discrepancies are small for all times, except for the very first pressure discrepancy which we already know from Table 4.1 is bigger. The temperature discrepancy increases slightly for both norms, but seem to stabilise as time goes on. This is confirmed when we let the simulation run for longer times as well.

The sequential solution should converge to the fully implicit solution when the time step decreases. While we in Table 4.1 and Figure 4.1 used a time step of 5 days, the time steps used in Figure 4.3 are $\Delta t = 0.3125, 0.625, 1.25, 2.5, 5, 10, 20$ days. The temperature discrepancy is plotted as a function of Δt when $t = 20, 100, 200$ days. It is easy to see that the error decreases when Δt decreases, which is satisfactory. The tolerances used were 10^{-6}.



(a) Pressure (b) Temperature

Figure 4.2: The L^2 and L^∞ pressure and temperature discrepancy for the test case versus time. A time step of $\Delta t = 5$ days was used.

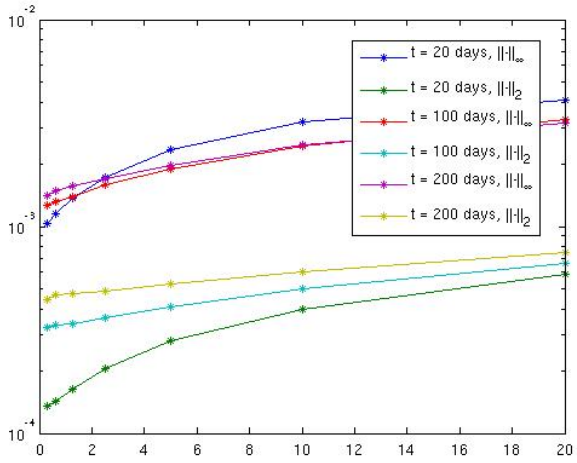
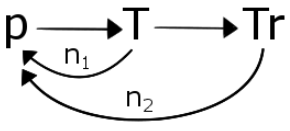


Figure 4.3: Temperature discrepancy for the test case as a function of Δt. The discrepancy for time $t = 20, 100, 200$ days is plotted for $\Delta t = 0.3125, 0.625, 1.25, 2.5, 5, 10, 20$ days.

It is possible to add outer iterations to our sequential method. There are two places to add outer loops. Either after having solved the temperature system, or after having solved the transport system. That is, we can have n_1 pressure-temperature iterations and n_2 pressure-temperature-transport iterations. The latter encompasses the pressure-temperature iterations as well. Whenever we add the outer iterations our sequential system will be of the form



where p denotes the pressure system, T denotes the temperature system and Tr denotes the transport system.

Up to now, we have had $n_1 = n_2 = 0$. In testing we found that the second outer iteration, the one that iterates after the transport equation has converged, is more effective at driving the discrepancies between the fully implicit solver and the sequential solver towards zero. We consider some cases to illustrate the point. In the first case, we go back to solve the pressure equation after the temperature equation has converged. That is, $n_1 = 1, n_2 = 0$. In the second case, we have added an iteration after the transport equation, so now $n_1 = n_2 = 1$. We then added a second iteration after the transport equation, but dropped the iteration after the temperature equation, $n_1 = 0, n_2 = 2$. This iteration is included in the fourth case, where $n_1 = 1, n_2 = 2$. We thus have the four cases

CASE 1: $n_1 = 1, n_2 = 0$

CASE 2: $n_1 = 1, n_2 = 1$

CASE 3: $n_1 = 0, n_2 = 2$

CASE 4: $n_1 = 1, n_2 = 2$.

The temperature and pressure discrepancies measured in the L^2 norm between the sequential and fully implicit solver for $t = 20, 100, 200$ days are given in Table 4.2. A time step of $\Delta t = 20$ days was used, the Newton tolerances were 10^{-6} and the simulation ran for 200 days. The first column in the table represents the discrepancy obtained when no iterations are used, $n_1 = n_2 = 0$. Comparing with the next column, where we have an iteration after the temperature equation, we see that the iteration does improve the accuracy of the temperature solution, the discrepancy is more than halved, but it does not really affect the pressure discrepancy. Adding an iteration after the transport equation has a big effect, however, which can be seen from CASE 3 in the table. The accuracy has improved both for the pressure and temperature equation, the accuracy for the pressure equation has actually improved by an order of 2. The outermost iteration is thus the iteration that is able to really drive the discrepancies down. This can also be seen when we compare the original case with CASE 3 and 4. If we apply two outermost iterations (CASE 3), the discrepancies for both temperature and pressure drastically decrease compared with the original case. If we then add an iteration after the temperature system, and keep the two outermost iterations (CASE 4), we see that the discrepancies remain fairly similar to those of CASE 3, and in some instances, it has increased a tiny bit. This inner iteration is thus not as good at driving the discrepancies towards zero.

One of the advantages of the sequential method is that it converges quicker than the the fully implicit method. The fully implicit method uses approximately 33 seconds to converge, while the sequential method with the same simulation

Table 4.2: Temperature and pressure discrepancies between the fully implicit and sequential methods, measured in the L^2 norm for $t = 20, 100, 200$ days, as well as the run time for the sequential method for the all the different cases. The run time of the fully implicit method was $t \approx 33$ sec. The temperature discrepancy is given by e_T, while e_p denotes the pressure discrepancy.

	Time (days)	ORIGINAL	CASE 1	CASE 2	CASE 3	CASE 4
e_T	$t = 20$	$5.842 \cdot 10^{-4}$	$3.007 \cdot 10^{-4}$	$2.888 \cdot 10^{-5}$	$5.500 \cdot 10^{-6}$	$4.190 \cdot 10^{-6}$
	$t = 100$	$6.596 \cdot 10^{-4}$	$2.509 \cdot 10^{-4}$	$3.567 \cdot 10^{-5}$	$3.906 \cdot 10^{-6}$	$4.475 \cdot 10^{-6}$
	$t = 200$	$7.533 \cdot 10^{-4}$	$2.470 \cdot 10^{-4}$	$3.735 \cdot 10^{-5}$	$4.057 \cdot 10^{-6}$	$4.614 \cdot 10^{-6}$
e_p	$t = 20$	$1.212 \cdot 10^{-1}$	$1.223 \cdot 10^{-1}$	$1.734 \cdot 10^{-3}$	$4.065 \cdot 10^{-5}$	$4.067 \cdot 10^{-5}$
	$t = 100$	$1.508 \cdot 10^{-2}$	$1.505 \cdot 10^{-2}$	$2.284 \cdot 10^{-4}$	$4.944 \cdot 10^{-6}$	$5.145 \cdot 10^{-6}$
	$t = 200$	$5.793 \cdot 10^{-3}$	$5.738 \cdot 10^{-3}$	$7.385 \cdot 10^{-5}$	$1.551 \cdot 10^{-6}$	$1.580 \cdot 10^{-6}$
Run time (sec)		11.3	15.7	28.8	24.5	37.6

properties and no outer iterations uses 11.3 seconds, see the first column in Table 4.2. Applying outer iterations will increase the run time of the sequential method however, which can be seen from the other columns of the table. The more iterations applied, the longer the run time. The run time of CASE 4 is in fact higher than the run time of the fully implicit method, and as the discrepancy is smaller for CASE 3, it would be sensible to use the outer iterations outlined by CASE 3. When applying outer iterations you thus have to weigh the goal of high accuracy to the goal of a fast solver when deciding on the number and placement of iterations.

Another way to add outer iterations, would be to use tolerances instead of setting a fixed number for n_1 and n_2. This way, we check whether the pressure residual has reached the tolerance either after temperature or transport has converged. If tolerance is reached, we continue to the next time step, or else we go back to solving the pressure system. With this approach, we get a higher number of iterations for the first time steps, before the number of iterations gradually declines. We are going to study this approach at the end of the next example.

4.5.2 SPE10, Layer 5

As the second example we study a model taken from the *Tenth SPE Comparative Solution Project* [4]. The project looked at two different models, and we are using Model 2. It is made up of $60 \times 220 \times 85$ cells, on a regular Cartesian grid. As this model is large (it has 1,122,000 fine cells to be exact), we are only going to be looking at one of the layers in the model. Layer 5 comes from the Tabert for-

mation, which is represented on the 35 first layers. The Tabert formation depicts a prograding near shore environment, and changes in the permeability field are smooth. The bottom 50 layers represent Upper Ness, a fluvial deposition. This formation thus consists of a permeability field with abrupt changes.

We assume the reservoir contains the two phases water and oil. To be able to extract the oil, we place five wells in an inverted five-spot pattern in the model. There are four production wells controlled by the bottom hole pressure, and one injection well controlled by the surface rate. The production wells are placed in each corner, while the injection well is placed in the middle of the layer. The permeability field of Layer 5, as well as the placement of the wells, are shown in Figure 4.4.

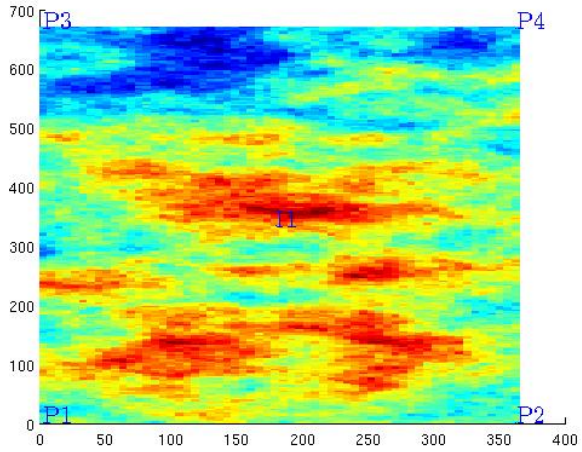


Figure 4.4: The logarithm of the permeability field of Layer 5 from the SPE10 data set, with the five wells used in this example.

The discrepancies between the fully implicit solution and the sequential solution are given in Table 4.3 for both pressure and temperature. Contour plots of the two temperature solutions for $t = 200$ days are given in Figure 4.5. The sequential contours are given in black, while contours given in color belong to the fully implicit solution. The time step used is $\Delta t = 5$ days, and the simulation ran for 200 days. The fully implicit, pressure, temperature and transport tolerances are all set to be 10^{-6}, and we have gone back to using $n_1 = n_2 = 0$. As can be seen from the table and figure, the two solutions match, though they are not completely identical. The contour plot shows that the sequential solver slightly overestimates the propagation of the heated region.

Test of Δt: The time steps used to generate Table 4.3 and Figure 4.5 are relatively small, a $\Delta t = 5$ days was used. Increasing the time step leads to an increase in the discrepancy, but not by much, see Table 4.4, where $\Delta t = 20$ days was used. The increase in error is by no means surprising, it is a known fact that

Table 4.3: Pressure and temperature discrepancy for SPE10, Layer 5. The temperature discrepancy is given by e_T, while e_p denotes the pressure discrepancy. We have $\Delta t = 5$ days, the different tolerances are 10^{-6} and the simulation ran for 200 days

	Time (days)	L^2	L^∞
e_T	t = 20	0.00029	0.00402
	t = 100	0.00039	0.00261
	t = 200	0.00049	0.00251
e_p	t = 20	0.00092	0.00629
	t = 100	0.00039	0.00539
	t = 200	0.00029	0.00337

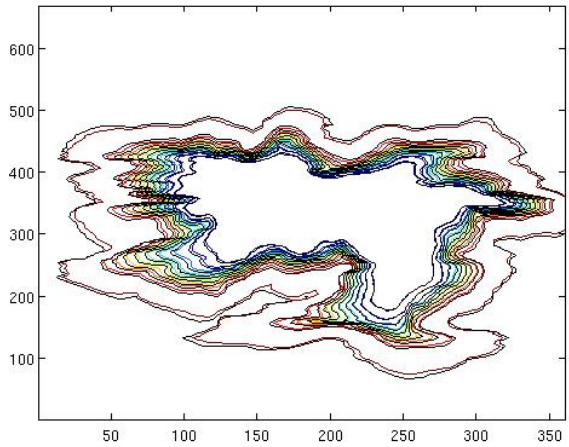


Figure 4.5: Contour plot of the fully implicit temperature solution and the sequential temperature solution when $t = 200$ days. The sequential contours are black, while the fully implicit solution has colorful contours.

most numerical methods tend to be less accurate when larger time steps are used. It is more surprising that the increase in the error is so small. This might mean that there is a weak connection between temperature and pressure, so an increase in the time step will not significantly alter the overall solution. If we were to use even bigger time steps, the fully implicit solver stops working. This happens around time steps of $\Delta t = 25$ days, and arises from the fact that negative formation factors occur. This is unphysical, and leads to non-finite values on the right hand side of the system. To account for this problem, one could use a slightly smaller time step. Interestingly enough, the sequential solver appears to have no such restriction on the time step Δt, and works even with a time step of 200 days. This is, of course, an advantage of the sequential solver over the fully implicit solver. Lastly, we would just like to note that the model from the previous example, the test case, accepts a little bigger time steps

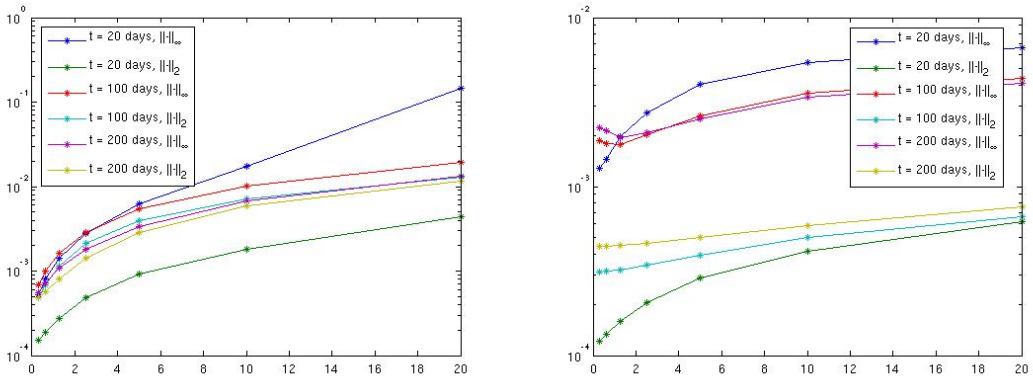
before the fully implicit system breaks down. That system breaks down with a time step of $\Delta t \approx 28$ days. The fact that the time step is slightly larger is not surprising, as that example is an easier model to simulate. For the example above, the dimensions were $D_x = 200, D_y = 200, D_z = 50$. All in all, we found that the simulator accepts larger time steps Δt when the dimensions are bigger, and smaller time steps when the dimensions are smaller. When, for instance, the dimensions are $D_x = 400, D_y = 400, D_z = 100$, the system breaks down with time steps $\Delta t \approx 31$ days, while the dimensions $D_x = 100, D_y = 100, D_z = 25$ leads to a breakdown when $\Delta t \approx 19$ days. The simulator seemed especially sensitive to changes in D_z. Equivalently, the solver tends to accept smaller Δt's when a higher number of grid points are used. The solver breaks down for the test case with $\Delta t \approx 5$ days when $30 \times 30 \times 20$ grid points are used, while a grid with $10 \times 10 \times 5$ cells does not break down before a time step of $\Delta t \approx 95$ days is used.

Table 4.4: Temperature error for the second example with larger time steps, $\Delta t =$ 20 days. The Newton-Raphson tolerance is 10^{-1} and the simulation ran for 200 days.

Time (days)	L^2	L^∞
t = 20	0.00062	0.00663
t = 100	0.00067	0.00439
t = 200	0.00076	0.00407

Figure 4.6 shows the difference between the fully implicit pressure and temperature solution and the sequential pressure and temperature solution for different time steps, Δt. We have looked at $\Delta t = 0.3125, 0.625, 1.25, 2.5, 5, 10, 20$ days, while the tolerances, n_1, n_2 and end time are kept as before. As can be seen from the figure, the error increases with larger time steps, it tends to be smallest when $\Delta t = 0.3125$ days, and largest when $\Delta t = 20$ days. This is just as it should be, as the sequential solution should converge to the fully implicit solution as the time step decreases.

Test of tolerance: Up until now, a Newton-Raphson tolerance of $tol = 10^{-6}$ has been used for all the different systems. Changing the tolerance to $tol_T = 10^{-1}$ for the temperature equation $\mathcal{R}_T$ in the sequential formulation does not affect the solution discrepancies to a large degree, see Table 4.5. The discrepancies are still of the same order as before. This is true for both the small and the larger time steps. The fact that we only have a small change in the error might be because the sequential system solves three different systems, pressure, temperature and transport. The change in the temperature tolerance will not affect the overall solution to a large degree, as the temperature does not affect the pressure and transport equation in the same manner as the pressure affects temperature and transport. When we change the *pressure tolerance* for $\mathcal{R}_p$, using the smaller temperature tolerance again, we find that the difference between the discrepancies are somewhat bigger than when we changed the temperature tolerance, see



(a) Pressure (b) Temperature

Figure 4.6: Discrepancies between the fully implicit pressure and temperature solution and the sequential pressure and temperature solution as a function of different time step sizes. The L^2 and L^∞ errors for $t = 20, 100, 200$ days are shown for $\Delta t = 0.3125, 0.625, 1.25, 2.5, 5, 10, 20$ days.

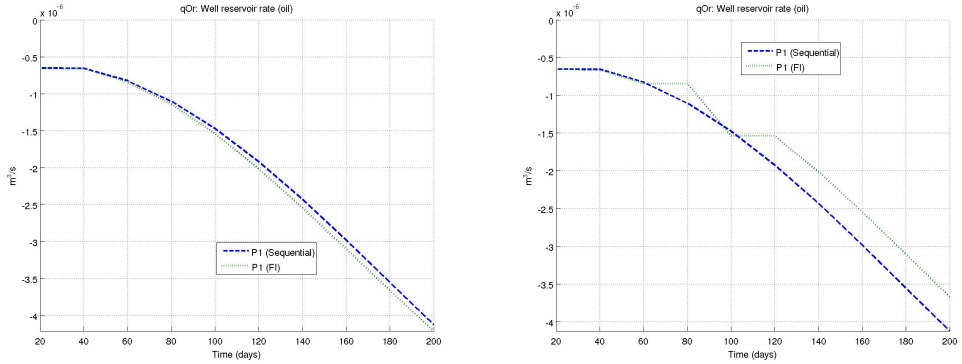
Table 4.5. The difference is not significant however, the errors are still mostly of the same order as the original errors. There is a difference between the plots of the sequential solution for the last time step, and the fully implicit solutions for the same time step. The difference is small enough to ignore however, and the plots have been omitted here for brevity. When we changed the temperature tolerance, the pressure errors were not affected. When changing the pressure tolerance, however, both the temperature and pressure error are affected. This might indicate that the change in temperature follows the change in pressure, as a difference in the sequential pressure solution leads to a difference in the sequential temperature solution, but not the other way around. Changing both the temperature and the pressure tolerances leads to even bigger discrepancies, naturally. We have omitted these results from Table 4.5, in an attempt to make the table easier to read. The discrepancies are still of the same order, but they are bigger than before.

Changing the *fully implicit tolerance* leads to some changes as well. There is a change between the discrepancies of course, the discrepancies arising from the larger tolerance are larger than the ones with the smaller tolerance, just as expected. But, when you look at the well reservoir oil rate, qOr, you see that the fully implicit solution is odd when a less strict tolerance is used, regardless of the fact that the discrepancy is still small, see Figure 4.7. The figure shows that the fully implicit solution is jagged when a fully implicit tolerance of 10^{-1} is used. A time step of $\Delta t = 20$ days is used, and the simulation ran for 200 days. The simulations shows the same tendencies for all the production wells in the system, and shows similar tendencies for the total well surface rate as well. Increasing the tolerance for the sequential method does not affect the oil rates in

Table 4.5: Temperature and pressure discrepancies between the fully implicit and sequential solutions. The temperature discrepancy is denoted e_T, while the pressure discrepancy is denoted e_p. A time step of $\Delta t = 20$ days was used, The first two columns used a tolerance of 10^{-6} for the fully implicit system, the pressure equation and the temperature equation. The tolerance of the next two columns was $tol = 10^{-1}$ for the temperature equation and 10^{-6} for the fully implicit system and pressure equation, and the tolerance for the last two columns was 10^{-1} for the pressure equation and 10^{-6} for the fully implicit system and temperature equation.

		$tol = 10^{-6}$		$tol_T = 10^{-1}$		$tol_p = 10^{-1}$	
	Time (days)	L^2	L^∞	L^2	L^∞	L^2	L^∞
	t = 20	0.000624	0.006633	0.000625	0.006629	0.000634	0.006647
e_T	t = 100	0.000666	0.004394	0.000668	0.004406	0.000672	0.004406
	t = 200	0.000761	0.004073	0.000764	0.004081	0.002287	0.004656
	t = 20	0.004442	0.14645	0.004442	0.14645	0.004436	0.14627
e_p	t = 100	0.012775	0.01927	0.012776	0.01927	0.012767	0.01925
	t = 200	0.011744	0.01339	0.011744	0.01339	0.012806	0.01666

the same way. The curves produced with the sequential method and a tolerance of 10^{-1} are still smooth and mimic the ones produced with a tolerance of 10^{-6} to a large degree. We thus need a smaller tolerance when working with the fully implicit system, adding to the advantages of the sequential system. The strange behaviour also disappears if we use a smaller time step Δt, so if we do not want to use a smaller fully implicit tolerance, a smaller time step can be used.

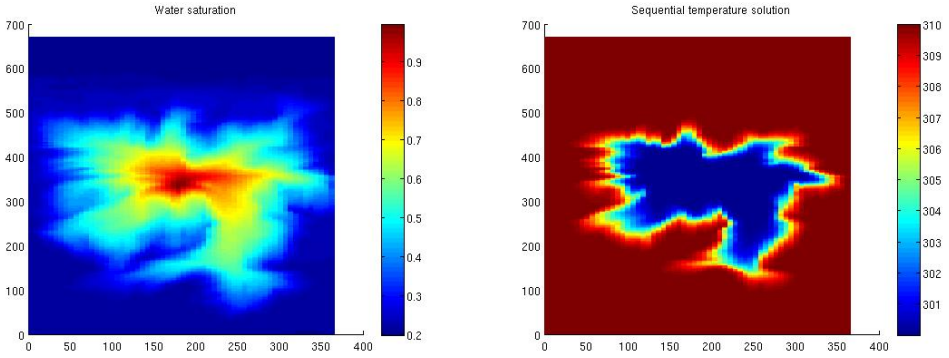


(a) $tol = 10^{-6}$ (b) $tol = 10^{-1}$

Figure 4.7: Reservoir oil rate for the first production rate, with different tolerances for the fully implicit system. The tolerance for the sequential system is kept at 10^{-6}.

Switching the order of pressure and temperature: As previously described, the sequential system is solved by first solving the pressure equation, before continuing and solving the temperature equation and transport equation. It is therefore interesting to see what happens if we solve the temperature equation first, before solving the pressure equation and then the transport equation. There is, after all, no theoretical reason why this should not work. But, when we try to solve for temperature first, we find that the temperature solver does not converge for the first time step, and the simulation breaks down. It has been a little difficult to figure out what causes the breakdown though. It probably has something to do with the advection term, because the method will often run if we change the enthalpy or flux velocities, thereby changing the advection term, but the problem is that the method then produces the wrong solution. The shape of the solution might be correct, but the values are wrong. The fact that the problem arises with the advection term does make sense, because advection describes the heat flow with the phases, and this depends on the pressure change. So we might have to solve for pressure first in order to have the correct pressure change. As the method seems to work well when we solve for pressure first, we are going to discard the idea of solving for temperature first and continue as normal.

Testing different enthalpies and internal energies: The sequential water saturation for the last time step is given in Figure 4.8. The figure also gives the temperature solution for the last time step. It is the same figure as the sequential solution in Figure 4.5. The time step used is $\Delta t = 5$ days, and the tolerance is $tol = 10^{-6}$.



(a) Water saturation (b) Temperature solution

Figure 4.8: Sequential water saturation and sequential temperature solution for the last time step, $t = 200$ days.

As can be seen from Figure 4.8, the temperature solution has the same form as the saturation plot. We can thus conclude that the temperature is transported through the reservoir by means of the water flow. It follows the motion of the fluid from the injection well throughout the reservoir. The temperature solution is thus strongly determined by the advection term in the temperature equation.

This all changes, however, if we change the internal energy or the enthalpy of the system. This is quite natural as the advection term depends on the enthalpy. Changing these quantities can thus make the contribution from the advection term smaller, and the solution will depend more on the dispersion term. We then find that the temperature moves throughout the reservoir independently of the flow, the warmer temperature will be able to flow to the colder parts on its own accord.

There are several ways of changing the temperature equation's dependency on the flow. In Figure 4.9 we have made three different changes to the system. In Figure 4.9a, we have changed the constant c_r in the enthalpy equation and the equation describing the internal energy of the rocks, Equation (4.2) and Equation (4.3) respectively, to $c_r = 0.5 \cdot 10^3$. In Figure 4.9b we have changed the internal energy to be $U_j = C_U T$, where $C_U = 4.1813 \cdot 10^6$ for $j = w, o$. While we in Figure 4.9c have used this new internal energy, and $c_r = 0.5 \cdot 10^3$. That is, we have three cases, where

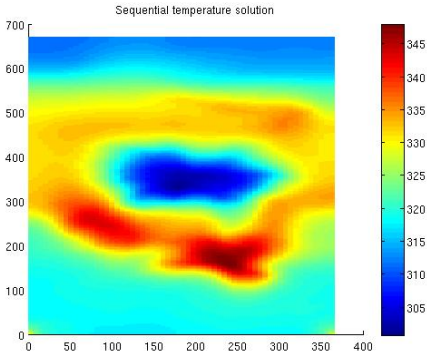
CASE 1: $c_r = 0.5 \cdot 10^3$,

CASE 2: $U_j = C_U T$,

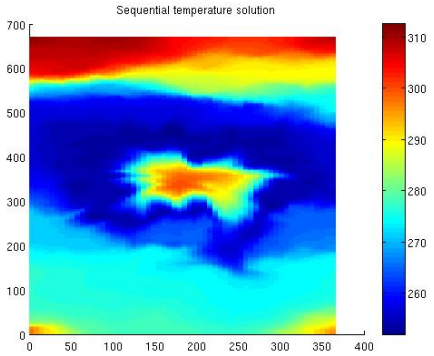
CASE 3: $c_r = 0.5 \cdot 10^3, U_j = C_U T$.

In the two first figures, Figures 4.9a and 4.9b, you can still see a little of the shape of the saturation. Here, the temperature moves both with the flow, and it moves through the reservoir on its own accord. The solution is thus dependent on both the advection and the diffusion term, the enthalpy and internal energy are such that the advection term is not too small. In the last figure, Figure 4.9c, it is clear that the temperature is less determined by the flow, it can move through the reservoir independently of the fluids.

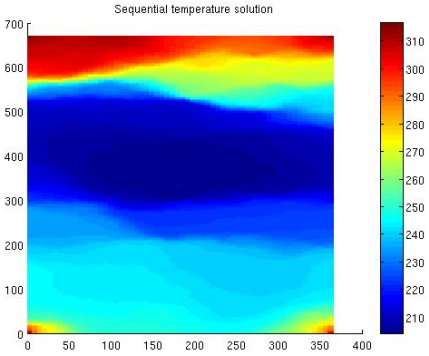
The error between the fully implicit system and the sequential system arising with the new enthalpy and internal energy changes is given in Table 4.6. As can be seen, the error is still small with these new enthalpy and internal energy values. The time step used was $\Delta t = 5$ days, $n_1 = n_2 = 0$, and the tolerances were all sat to be $tol = 10^{-6}$. If we again look at the effects of changed time steps and tolerances, we find that changing the fully implicit tolerance influences the error the most when the enthalpy is changed. The discrepancy difference is not big however, and the results have therefore been omitted for brevity. Changing the time steps also has a small effect on the error. Changing only the temperature tolerance does not have that much to say, but this might come from the fact that the pressure and transport tolerances are still small, which might make the discrepancy stay somewhat the same. The temperature tolerance does not affect the error to a large degree when the internal energy or when both the internal energy and the enthalpy are changed either. When these are changed, it is the time steps that has the most to say for the errors. As we saw no significant change, we have also omitted these results for brevity.



(a) Case 1: Changed enthalpy



(b) Case 2: Changed internal energy



(c) Case 3: Changed internal energy and enthalpy

Figure 4.9: The effect of different enthalpies and internal energies on the temperature solution.

Table 4.6: Temperature discrepancy for the figures in Figure 4.9. The first two columns represents the enthalpy change, the next two columns represents the change in internal energy, and the two last columns represents the change of both internal energy and the enthalpy.

	Case 1		Case 2		Case 3	
Time (days)	L^2	L^∞	L^2	L^∞	L^2	L^∞
t = 20	0.00116	0.00862	0.00069	0.00269	0.00116	0.00374
t = 100	0.00139	0.00352	0.00239	0.00414	0.00399	0.00533
t = 200	0.00195	0.00368	0.00372	0.00450	0.00654	0.00705

Test of run time: Just as with the first example, the sequential solver is faster than the fully implicit solver. The run time of the sequential solver when $\Delta t = 20$ days, and the simulation run for 200 days is 16.8 seconds, while the run time of the fully implicit solver with the same parameters is 40.3 seconds. The original enthalpies and internal energies are used, and the different Newton-Raphson tolerances are 10^{-6}.

We have previously seen that we are able to further improve the sequential method by applying outer iterations, but that this comes at a cost of a less efficient method. We have previously operated with a fixed number of outer iterations, n_1 and n_2, but we could also check the pressure residual after the transport system has converged and if the residual is higher than a preset *outer tolerance*, tol_{outer}, we re-run the time step. This way, we can control the final residual, but we will not be able to control the number of outer iterations, n_2, which of course will affect the run time. Figure 4.10 gives a comparison of the computational time for the fully implicit method and the sequential method where different outer tolerances have been used. We have tested the run time for $tol_{outer} = 10^{-1}, \cdots, 10^{-10}$. As can be seen from the figure, the run time of the sequential method is less than the run time of the fully implicit method as long as the outer tolerance is less than 10^{-9}. We thus have a flexible method that even with strict outer tolerances for the pressure residual is more efficient than the fully implicit method. As the outer iterations help drive the discrepancy between the two methods to zero, and as the method often is still more efficient than the fully implicit method, outer iterations should be seriously considered when the sequential method is used. We can also see that the sequential method uses approximately the same time for the first five outer tolerances ($tol_{outer} = 10^{-1}, \cdots, 10^{-5}$) so if we choose to use outer iterations, there really is no reason to choose the most relaxed tolerances because we get better results with the more strict tolerances with approximately the same run time.

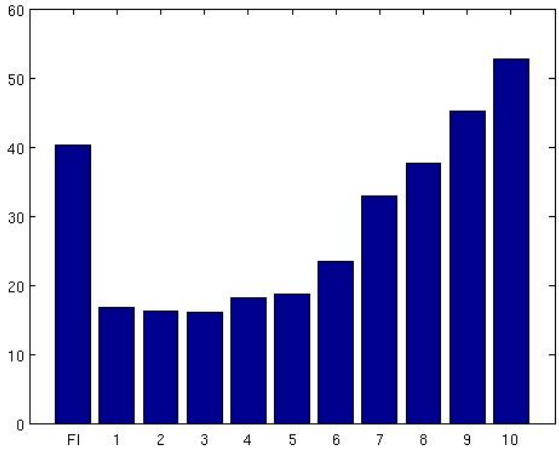


Figure 4.10: Run time of the fully implicit (FI) method, as well as the sequential method with ten different outer tolerances $tol_{outer} = 10^{-1}, \cdots, 10^{-10}$. The numbering on the x-axis corresponds to the power of the outer tolerance for that run. So bar number 4 corresponds to the run time of the sequential method with an outer tolerance of 10^{-4}. A time step of $\Delta t = 20$ days has been used for all runs, and the simulations ran for 200 days.

When we talk about computational cost, it is important to remember that part of the code, like automatic differentiation, is not built for speed. The run times presented in Figure 4.10 could therefore be significantly reduced if we were to go through the code and optimize with respect to efficiency.

4.6 SUMMARY AND OBSERVATIONS

The sequential method solves our model equations (2.5) by decoupling the system we obtain with the fully implicit method into three subsystems, one for pressure, one for temperature and one for transport. This results in a more flexible method that in addition is more stable and more efficient than the fully implicit method. By adjusting different parameters, like the size of the time step Δt and different tolerances such as the Newton-Raphson tolerance and the outer tolerance, we can improve the accuracy of the sequential method. This comes at a cost of efficiency, however. The method has a restriction in that we have to solve for pressure before temperature, as the method fails when we try to solve for temperature first.

To further improve the efficiency, we can apply numerical methods on each subsystem. This will be done in the next chapter, where we introduce a multiscale method to the system.

5

THE MULTISCALE METHOD

From Chapter 4, we know that the Newton-Raphson method is used extensively. This method requires the inverse of the Jacobian matrix. With the sequential method, we have three different Jacobians; one for the pressure equation, one for the temperature equation and one for the transport equation. The sizes of the different Jacobians will generally be of size $n \times n$, where n is the number of cells in the grid. The Jacobian of the pressure system will be slightly larger, as this system includes the wells, though the wells constitutes a relatively small part of the system. For the kinds of problems we are studying, the grid will be quite large as it represents a reservoir. This in turn makes the Jacobians large. Finding the inverse of the matrices can thus be time consuming and difficult. Sometimes we might even run into memory problems. The time consuming property of the inverse might have been okay if we were to find the inverse only once. This is not the case, however. For each time step, we have three different rounds of the Newton-Raphson method, and in each Newton-Raphson iteration we have to find the inverse of a Jacobian matrix. This leads to a high number of inverses of Jacobians, which again reduces the efficiency of the code. We therefore want to apply a numerical method to the system in order to speed up the process of solving $\Delta x = J_j^{-1}\mathcal{R}_j$, where the advantage of speedup does not come at the cost of inaccuracy. This is where the multiscale method comes in.

Multiscale methods have been the interest of an active research field in recent years, and there are thus several different multiscale solvers one could use [11, 7, 9, 1, 17, 6, 12]. We are going to use a fairly recent method, namely the *multiscale restriction-smoothed basis* (MsRSB) method [19, 20, 18, 8]. The method has previously been proven to work well on isothermal systems, and it is interesting to see how it reacts when we introduce the temperature system. We will therefore apply the method on the pressure and temperature equations, while the transport equation will be solved through normal means. Herein comes a thorough study of the MsRSB method, before we in the last section apply the method on numerous examples to vigorously test the method.

5.1 MULTISCALE RESTRICTION-SMOOTHED BASIS METHOD

The idea behind MsRSB and other multiscale methods, is to compress the equations to a smaller, coarser grid than the original, fine grid, and solve the system on the coarse grid. We are thus making the overall system smaller, with fewer unknowns. After having solved the smaller system, we expand the solution back to the fine grid. To do this, we need operators that make the system small, and operators which can take the coarse scale solution and expand it back to its original size.

To introduce the MsRSB method, we start by introducing the grids and operators. Let Ω be the fine grid, which encompasses $i = 1, \cdots, n$ cells. Similarly, let $\overline{\Omega}$ be the coarse grid, with $j = 1, \cdots, m$ grid blocks, where $m < n$. The coarse grid is defined through a partition of the fine grid, and a cell from Ω can only belong to one coarse block. There are thus no overlap of the blocks. The operator that expands the coarse system is called the *prolongation operator* and is denoted P. It is a sparse matrix of size $n \times m$, and consists of basis functions that map unknowns from blocks to cells. We have one basis function for every grid block, and each column in P constitutes a basis function. Let x denote either the fine scale pressure solution, or the fine scale temperature solution. Then, if we have already found the coarse scale solution x_c, the fine scale solution can be approximated by

$$x \approx P x_c.$$

We will not be able to recreate the fine scale solution exactly, the operator will only give an approximation. But this approximation will in many cases be able to imitate the fine scale solution closely, and we are satisfied.

The operator that compresses the fine scale system into the coarse scale system is an $m \times n$ sparse matrix called the *restriction operator*, and is denoted R. This R should not be confused with the $\mathcal{R}$ used for the equations on residual form. At each Newton-Raphson iteration k, we want to solve the fine scale system $J_{x,k}\Delta x_k = -\mathcal{R}_{x,k}$, where $\Delta x = x_{k+1} - x_k$ and $x = p$ or T. To do so, we use the two operators to find a coarse scale system from which we can find a coarse scale solution and then approximate the fine scale solution through $\Delta x \approx P\Delta x_c$. By applying the two operators, and omitting the iteration number, we find that the coarse scale system is given by

$$\begin{aligned} J_x P \Delta x_c &= -\mathcal{R}_x, \\ R J_x P \Delta x_c &= -R\mathcal{R}_x, \\ J_{x,c} \Delta x_c &= -\mathcal{R}_{x,c}. \end{aligned}$$

Notice that $J_{x,c} = RJP$ is a matrix of size $m \times m$, which makes this new system less costly to solve, at least as long as its sparsity structure and condition num-

ber are not significally worse than that of J.

Now that we have shown how to find the coarse scale system, we are ready to define the prolongation and restriction operators. We first define the restriction operator, as this operator is easiest to define. We are going to use a *control volume summation operator* for R,

$$R_{ij} = \begin{cases} 1, & y_j \in \overline{\Omega}_i \\ 0, & \text{otherwise.} \end{cases}$$

We thus have that $R_{ij} = 1$ if cell j belongs to coarse block i. Applying R to a vector or matrix is therefore equivalent to adding up these cells. We are thus adding the variables defined in the cells. Other operators could also be used as the choice for R. But, as the control volume summation operator works well for the MsRSB method, this choice of R ensures the important property of mass conservation, and as it is the standard for both the MsRSB and the multiscale finite volume method [19, 11], we are going to use R as defined above.

When it comes to the prolongation operator P, MsRSB uses, as many of the other multiscale methods, basis functions to define P. Where the other methods use localized flow problems that in different ways depend on the fine grid to find the basis functions, MsRSB uses an iterative scheme to find the functions. This way we can forgo the complicated set-up of the local problems. The scheme is defined so that the method can be used on as many, and as complex, grid types as possible. It is further designed to ensure that the basis functions are independent of time, and that they have *partition of unity*.

As we want a method that works on general grids, we will focus on the error instead of the given grid, and try to find basis functions that keep the coarse grid error smooth when we interpolate to the fine scale. Generally, when we restrict a problem to a coarser grid, the multiscale method is able to handle so-called global errors. That is, we are able to recreate global tendencies, factors such as gravity effects that are present on the coarser grid. These global errors are errors that iterative numerical methods, often used as smoothers in such methods as multigrid, are not able to dispose off quickly. The errors are therefore often called *smooth errors* or, to prevent confusion with geometrical smooth errors, *algebraically smooth errors* [5]. Instead of focusing on the given problem when we design the prolongation operators, we focus on the algebraically smooth errors. Having handled the global error on the coarse scale, we want to preserve the small algebraically smooth errors when we prolong to the fine scale, and the prolongation operators should therefore be defined accordingly. We thus want prolongation operators that ensure that if we have an algebraically smooth solution on the coarse grid, then the fine scale solution is smooth as well. If we always manage to keep Δx algebraically smooth, the method can work on many different problems, independently of what type of grid and physical properties

we have.

Algebraically smooth errors are characterized by small residuals [5]. We have a residual whose norm is given by $||r|| = ||\mathcal{R} + JP\Delta x_c||$. To minimize this, we have to minimize JP, as the other factors are already known and thus unable to be minimized. To find the prolongation operator, MsRSB therefore employs a method that minimizes $||JP||_1$. We thus have to minimize JP_j for all j, where P_j is basis function j, placed in the jth column of P.

If we have to change P every time we are to apply the multiscale method, we will use a lot of time finding the operator, and the method will be less efficient. The MsRSB method will therefore use a slightly different matrix than the Jacobian when finding the prolongation operator. It uses a matrix that enforces the sum of the fluxes to be equal to zero. This means that we use a local system that is similar to the solution of an elliptic Poisson system. By doing this, the basis functions will be fairly similar to steady-state basis functions, and we have lost the time dependency of the functions. The matrix MsRSB uses is of the form

$$J_P = \frac{1}{2}\left(J_x + J_x^T - I_{n\times n}\left(J_x + J_x^T\right)\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix}\right),$$

where $I_{n\times n}$ is the $n \times n$ identity matrix, and $[1 \cdots 1]^T$ is a column vector of ones of length n. By studying J_p, one can see that we have $\sum_j J_{P,ij} = 0 \ \forall i$, and the sum of the fluxes will be zero.

In addition, we want the basis functions to have partition of unity. A function that has partition of unity will sum all its function values to one, i.e., $\sum h(y) = 1 \ \forall y$. For the basis functions, this is equivalent with

$$\sum_j P_{ij} = 1.$$

The desire for partition of unity comes from the fact that the basis functions approximate the fine scale solution x through

$$x_i \approx \bar{x}_i = \sum_j^m P_{ij} x_{c,j},$$

which means that if we have found a coarse solution such that $x_{c,j} = x_i$ and if P_j has partition of unity, then

$$\bar{x}_i = \sum_j P_{ij} x_{c,j} = x_i \sum_j P_{ij} = x_i,$$

and the approximation will give the exact solution for cell number i.

We can now introduce the iterative scheme. To minimize $J_P P_j$, MsRSB uses a *weighted Jacobi method*. Let w be a relaxation factor, usually set to $2/3$, and let D be a diagonal matrix containing the diagonal part of J_P. Furthermore, let n be the iteration number of the iterative scheme. The method used to find the basis functions is then given by

$$P_j^{n+1} = P_j^n - wD^{-1}J_P P_j^n. \tag{5.1}$$

The iterations carry on until the prolongation operator is sufficiently smooth, that is, until the error in the iterative scheme has reached a preset tolerance. We also have to make sure the iterative scheme does not produce basis functions that cover the whole domain. This will be discussed further below. First we introduce the initial guess for the basis functions, which will be given as a constant function for each coarse block,

$$P_{ij}^0 = \begin{cases} 1, & \text{if cell } i \text{ belongs to coarse block } j, \\ 0, & \text{otherwise.} \end{cases}$$

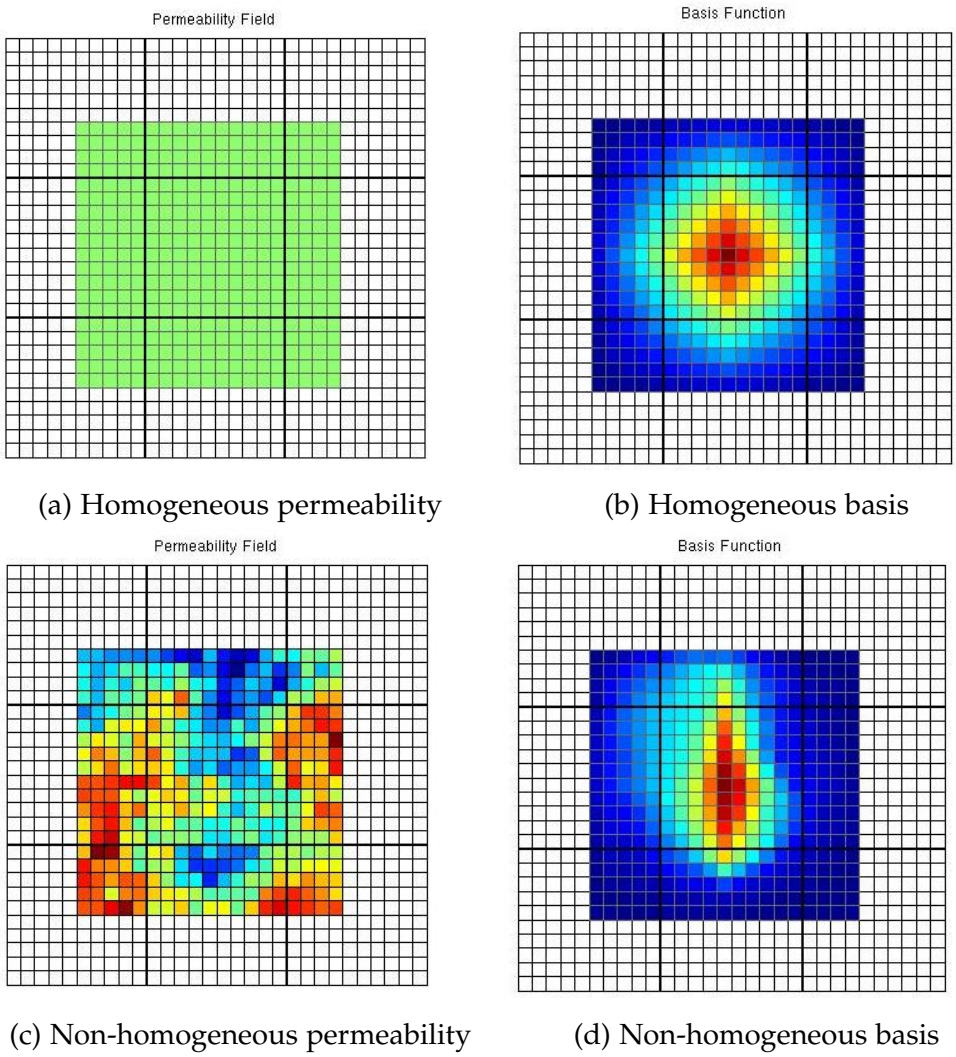
Said differently, we have that $P^0 = R^T$.

For our initial guess of P, partition of unity is inherently true, as $\sum_j P_{ij}^0 = 1$ by definition. Further, we have that MsRSB ensures the updated basis functions to have partition of unity as well. Remember from above that $\sum_k J_{P,ik} = 0 \ \forall i$. By assuming that the basis functions have partition of unity for iteration n, which we already know is true when $n = 0$, we have that

$$\begin{aligned} \sum_j P_{ij}^{n+1} &= \sum_j (P_{ij}^n - \frac{w}{J_{P,ii}} \sum_k J_{P,ik} P_{kj}^n) \\ &= \sum_j P_{ij}^n - \frac{w}{J_{P,ii}} \sum_j \sum_k J_{P,ik} P_{kj}^n \\ &= 1 - \frac{w}{J_{P,ii}} \sum_k J_{P,ik} \sum_j P_{jk}^n = 1. \end{aligned}$$

The partition of unity is thus maintained for all n.

It is not enough to just introduce the weighted Jacobi method and show partition of unity when designing the basis functions. We also have to restrict the iterative scheme, or else it will eventually give basis functions that cover the whole domain. We therefore define *support regions* that determine the support of the basis functions. This of course, means that the functions are nonzero only inside the regions. How MsRSB defines these support regions will be discussed below. We first want to show examples of the actual functions. Figure 5.1 gives two different basis functions. Note that the functions are only defined inside a region, the support region. The figures to the left, Figures 5.1a and 5.1c, give the logarithm of two different permeability fields, both inside the support region of the middle block, while the figures to the right, Figures 5.1b and 5.1d,



(a) Homogeneous permeability (b) Homogeneous basis

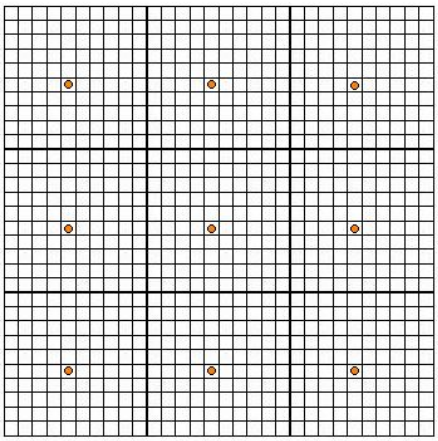
(c) Non-homogeneous permeability (d) Non-homogeneous basis

Figure 5.1: The logarithm of two different permeability fields inside the support region for the middle coarse block and their resulting basis function. The topmost permeability is a homogeneous permeability, while the bottom permeability is taken from layer 5 of the SPE10 data set [4]. The grids are equal for the two different examples, and both the fine and coarse scale grids can be seen.

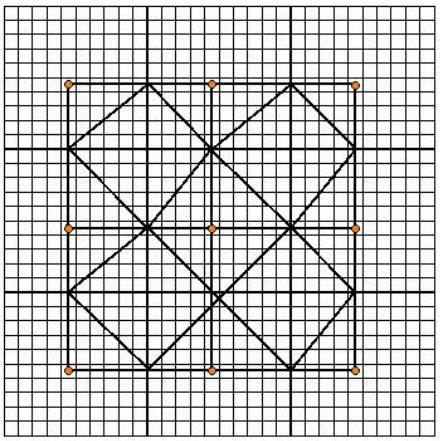
give the resulting basis functions. The top figures have a homogeneous permeability field, while bottom have a heterogeneous permeability field. The fine and coarse scale can both be distinguished in the figures. As can be seen, the basis functions are not equal, the functions are able to adjust depending on the permeability field. The fact that MsRSB's prolongation operator is able to adjust to the physical properties of the problem and captures the influence of the permeability is convenient as the permeability field is important to the flow.

As previously stated, we need to make sure that the individual basis functions do not cover the whole domain. This is done through support regions, which determine the support of the basis functions. To find the support region for

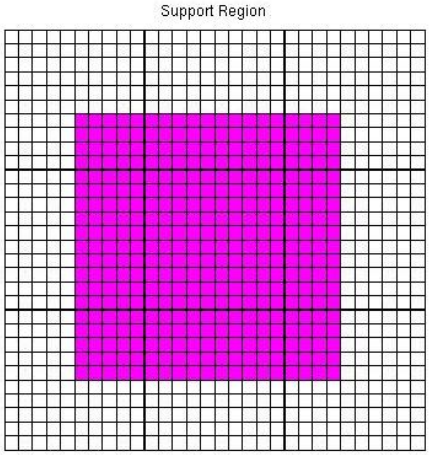
coarse block i, MsRSB first collects all the coarse blocks that share a face with i, and the blocks that share a single point with i, which for instance occurs on the corners of i when the grid is a regular Cartesian grid. We then make a triangulation between all the selected block centroids and block face centroids. In the end, the support region for block i consists of all cells within this triangulation. The process is illustrated in Figure 5.2. We want to find the support region of the middle block, the one used in Figure 5.1. We start by collecting the blocks that share a face or a single point with the block, as shown in Figure 5.2a. In Figure 5.2b, the triangulation is made, before the support region is defined in Figure 5.2c.



(a) Collect blocks



(b) Triangulation



(c) Support region

Figure 5.2: Illustration of the process required to find the support region of the middle coarse block.

As the support regions are found through block and face centroids, and as these are fairly easy to find no matter how complicated the grid structure is, it is clear that the support regions can be found for many different grid structures. This again means that the basis functions can be found for many different grid struc-

tures, without having to change the implementation for each grid.

As we have now defined the support regions, we can proceed to show the full iterative method MsRSB uses to update the basis functions. Let n be the n'th iteration in the scheme to find the j'th basis function. Each iteration follows four steps. First, we define

$$d_j = -wD^{-1}J_P P_j^n.$$

This is just the update from Equation (5.1). As previously stated, this update will grow to cover the whole domain unless we use the support regions to restrict d_j. To enforce this, we will have three different cases: One for when we are outside the support region, one for when we are inside only one support region, and one for when we are on the boundary of another support region. The boundary of a support region is given by the cells that are adjacent to the outermost cells in the region. The last case occurs because the support region of a block generally is defined a little outside of the block, not just on the block, and each block might thus have several support regions defined on its area, see Figure 5.3 for an illustration. If we are outside of the support region of j,

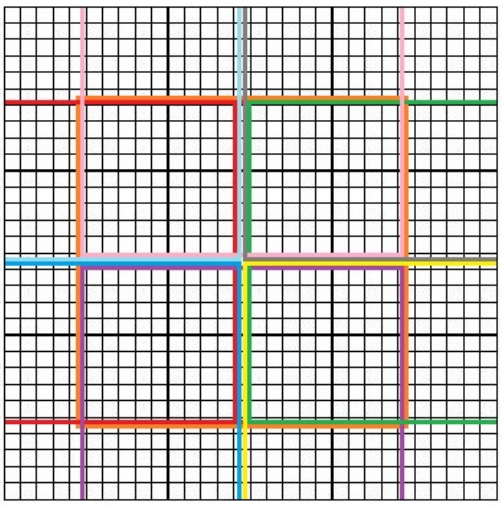


Figure 5.3: Support boundaries for all the coarse blocks in the coarse grid. Each boundary is given by a different color and form a box around its coarse block. The support boundary of the middle block is given in orange, and the support boundary of the uppermost corner on the left side is given in light blue. Where there are many colors, the support boundaries overlap. As can be seen, a coarse block can have several support boundaries defined on its area.

then we must have $d_j = 0$, so that $P_j = 0$ outside the support region. If we are inside the support region, and not on a boundary of another region, we will not have any problems, and we have that $d_j = d_j$ as defined above. This automatically preserves partition of unity. When we encounter a cell i that is part of the boundary of one or more support regions, we have to normalize over

the cell's support regions in order to keep the partition of unity. This means that we have $d_{ij} = \frac{d_{ij} - P_{ij}^n \sum_{k \in H_i} d_{ik}}{1 + \sum_{k \in H_i} d_{ik}}$, where H_i gives the support regions that cell i is on the support boundary of. If we let I_j be the support region of basis function j and G consists of all the cells in all support boundaries, then all of this yields the second step in the algorithm, namely

$$d_{ij} = \begin{cases} 0, & i \notin I_j, \\ d_{ij}, & i \in I_j, i \notin G, \\ \frac{d_{ij} - P_{ij}^n \sum_{k \in H_i} d_{ik}}{1 + \sum_{k \in H_i} d_{ik}}, & i \in I_j, i \in G. \end{cases}$$

We can now proceed to the third step, which updates the basis function,

$$P_j^{n+1} = P_j^n + d_j.$$

As a quick aside, we can show that this really does preserve partition of unity. We assume the property is true for iteration n, that is, we assume $\sum_j P_{ij}^n = 1$. Then we have

$$\sum_j P_{ij}^{n+1} = \sum_j (P_{ij}^n - d_{ij}).$$

Now, if $d_{ij} = 0$, we already have the answer. Similarly, we have already shown that the functions have partition of unity when $d_{ij} = -wD^{-1}J_P P_j^n$. All that remains is thus to show that the statement is true when $i \in I_j, i \in G$. Then, we have that

$$\begin{aligned} \sum_j P_{ij}^{n+1} &= \sum_j (P_{ij}^n - d_{ij}) = \sum_{j \in H_i} (P_{ij}^n - d_{ij}) \\ &= 1 - \sum_{j \in H_i} \frac{d_{ij} - P_{ij}^n \sum_{k \in H_i} d_{ik}}{1 + \sum_{k \in H_i} d_{ik}} \\ &= 1 + \frac{1}{1 + \sum_{k \in H_i} d_{ik}} \left(- \sum_{j \in H_i} d_{ij} + \sum_{j \in H_i} P_{ij}^n \sum_{k \in H_i} d_{ik} \right) \\ &= 1 + \frac{1}{1 + \sum_{k \in H_i} d_{ik}} \left(- \sum_{j \in H_i} d_{ij} + 1 \cdot \sum_{k \in H_i} d_{ik} \right) \\ &= 1. \end{aligned}$$

It is thus clear that MsRSB's basis functions truly preserve partition of unity.

To decide whether we have found the converged basis function, we continue to the last step, which checks the local error outside the boundary regions

$$e_j = \max_j (|d_{ij}|), i \notin G.$$

If $||e_j||_\infty < tol$, we have found our basis function. If the opposite is true we go back to the first step and start a new iteration by finding d_j.

To apply the MsRSB method we thus have to find R through the volume summation operator, and then find the support regions so we can produce P with the iterative method. This is then used to produce the coarse scale system $J_{x,c}\Delta x_c = -\mathcal{R}_{x,c}$, from which we can find the coarse scale solution Δx_c. Finally, we use P to find the approximation of the fine scale solution through $\Delta x \approx P\Delta x_c$.

To further improve the approximation, one could introduce an iterative procedure to the multiscale method, aimed to drive the fine scale residual towards zero. This results in an iterative multiscale method, which we will now discuss.

5.2 ITERATIVE MULTISCALE

The multiscale method is not the only method that interpolates a solution on a coarse grid to a solution on a fine grid. This is also the basic idea behind the class of numerical solvers known as *multigrid methods*. There are many different types of multigrid methods, some designed to solve specific problems while others are designed to handle a broader spectre of problems. They are all based on a few defining steps, however, which utilize a hierarchy of grids. The methods are built on the fact that global, low frequency errors, which many iterative numerical methods use a long time to solve, can be removed on a coarse grid, while local, high frequency errors can be removed quickly by the iterative methods. The multigrid methods are thus built on a recursive call of the general steps:

1. Smoother: Removes high frequency error,
2. Restriction: Downsizes the problem to a coarse grid and finds the coarse scale solution,
3. Interpolation: Prolong the coarse solution to the fine scale.

Going back to the multiscale method, we know that the method will generally not give the exact fine scale solution. The method will only give an approximation of the solution, as the prolongation operator is only able to approximate the fine scale solution. The multiscale method is good at capturing the global tendencies, but not as good at capturing smaller, more local tendencies, such as density change. Still, we want to capture the local effects, because we want the multiscale solution to be as close to the fine scale solution as possible. To do this, we are going to use some of the same ideas as the ones used in the multigrid methods. The method we are going to use is called an *iterative multiscale* method. This idea has been used for other multiscale methods as well, for instance [7, 17].

The iterative multiscale method is, as the name suggests, an iterative method. Just as the multigrid methods, it uses the fact that many linear solvers are able to remove high frequency errors quickly. It therefore employs n_s steps of a fine scale smoother to drive the fine scale residual towards zero and thus remove

the local errors. To remove the global errors, it applies the multiscale method to solve a coarse scale system that is then prolonged back to fine scale. Each iteration is thus similar to a 2-level multigrid method, where we start with a *smoother* to remove high frequency errors before we *downsize* to a coarse grid with the restriction operator R and lastly *interpolate* the coarse solution to the fine scale with the prolongation operator P.

Let l denote the iteration number in the iterative multiscale scheme. We first apply a smoother on the so-called defect d^l to correct the local error. The defect should contain the high frequency error. The smoothing step is denoted $y^l = S(d^l)$, where $S(\cdot)$ is the smoothing method, an inexpensive linear solver. To find the defect, we remember from above that we for each Newton-Raphson iteration solve $J_c U_c = -\mathcal{R}_c$ instead of solving the fine scale system, where we use $U = \Delta x$ in an attempt to make the notation simpler. By applying the multiscale method to the fine scale system, the local effects will be lost and the local error remains more or less unchanged. Thus, to capture the local effects, the defect at iteration l will be given by the fine scale system,

$$d^l = -\mathcal{R} - JU^l,$$

where U^l denotes the iterative multiscale solution for iteration l.

Instead of finding the Newton update U by directly solving the coarse system and prolonging back to the fine scale, we are now going to apply the iterative multiscale method to find the update U. We want an iterative method that finds the Newton update and captures both the local and global effects. We have already found the local effects through y, and we know that the global effects can be captured through the multiscale method. Further, we know that $U = x_{k+1} - x_k = -J^{-1}\mathcal{R}$, which is where we will start. The iterative Newton update is found by simple arithmetic tricks like adding and subtracting variables, and by using the fact that $J_c = RJP$, and thus $J = R^{-1}J_cP^{-1}$. We have

$$\begin{aligned} U &= -J^{-1}\mathcal{R} + U - U \\ &= U - J^{-1}(R + JU) \\ &= U + J^{-1}d + y - y \\ &= U + J^{-1}(d - Jy) + y \\ &= U + PJ_cR(d - Jy) + y. \end{aligned}$$

The iterative scheme is thus given by

$$U^{l+1} = U^l + PJ_cR(d^l - Jy^l) + y^l,$$

where $y^l = S(d^l)$. One round of the smoother and update is called a multiscale cycle.

We let the iterative scheme run until a predefined tolerance is reached, or, in some cases, until we have reached a predefined limit on the maximum number of iterations. The new Newton iteration can be found when the method has converged, and it will be given by

$$x_{k+1} = x_k + U^L,$$

where L is the last iteration in the iterative multiscale method.

It should be noted that the terms *iterative multiscale* and *multiscale* are used interchangeably in the rest of this thesis. Both terms will refer to the use of the multiscale cycles unless otherwise specified.

When it comes to the smoother, there are many methods one could choose. As long as it is inexpensive, you can basically pick as you like. Methods such as the Jacobi method, the Gauss-Seidel method and the incomplete LU factorization are all good choices. We are mostly going to use the latter, and will therefore give a quick review of the method.

Incomplete LU factorization

We will look at the general case and consider the system $Ax = b$. Most matrices can be factored into two matrices, a lower triangular matrix L and an upper triangular matrix U, such that $A = LU$. The system can then be solved by first solving $Ly = b$ and then $Ux = y$. These two systems can be solved quickly, as L and U are triangular.

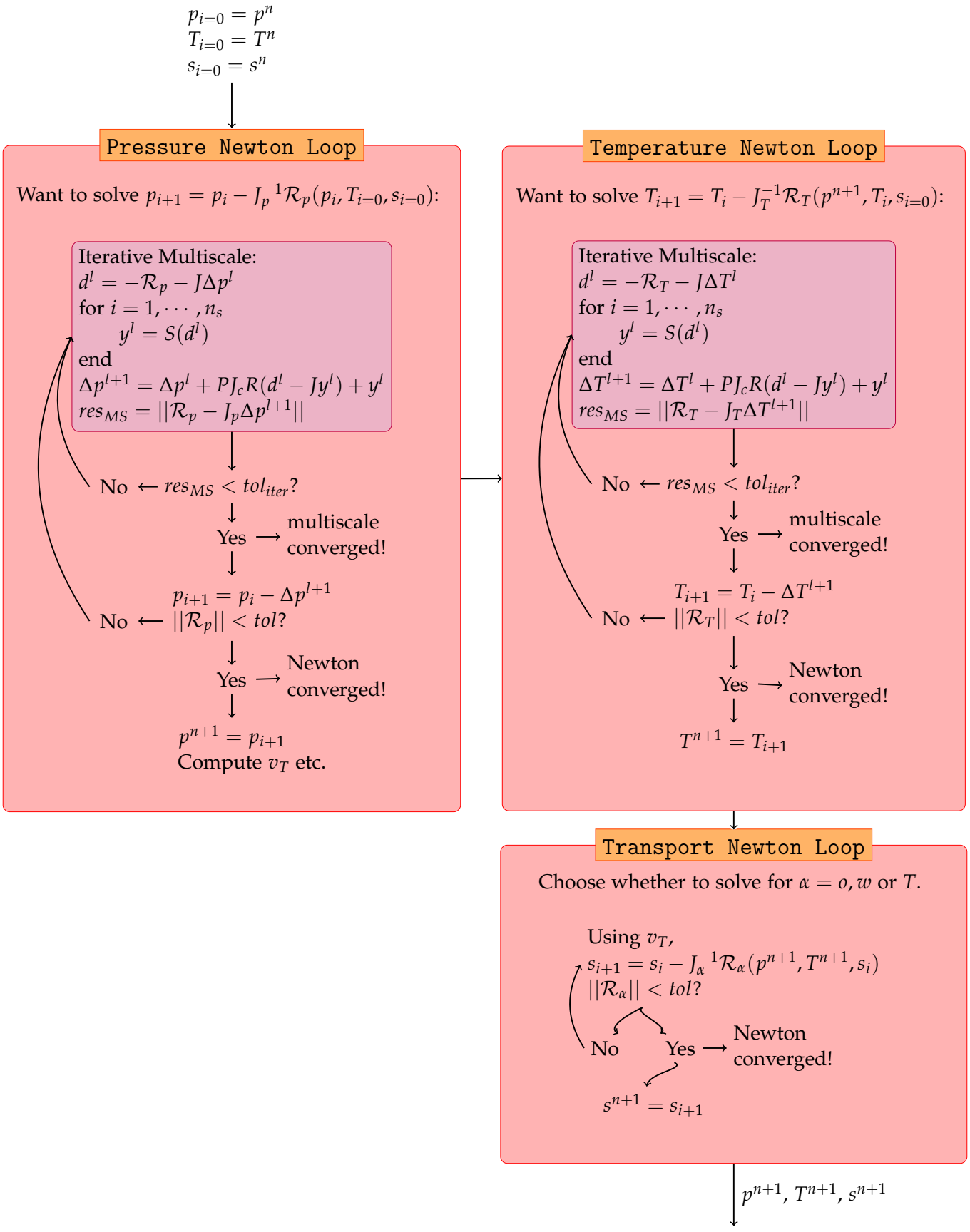
Even though A is sparse, there is no way to ensure that L and U are sparse. This can lead to large matrices, even though they are triangular, which in turn leads to increased memory usage. It is therefore possible to enforce the two matrices to be zero everywhere A is zero. This is the incomplete LU factorization with zero fill-ins, generally denoted ILU(0). The incomplete LU factorization will of course only be an approximation to the original matrix A, and as such $LUx = b$ will only give an approximation to the solution that would have been obtained from $Ax = b$. The approximation will be found quickly, however. As we only require a solution that will rapidly smooth the local error, but not an exact solution to the overall system, the incomplete LU factorization is a good method for us to use as our preconditioner.

MATLAB has developed a good function for the incomplete LU factorization, and this has been utilized in this project. We will therefore not go into detail about how to find the L and U matrices. The interested reader can consult various books and papers on the LU factorization, for instance [24].

5.3 OVERVIEW

There have so far been some twist and turns. To solve Equation (2.5), we need to use quite a few different methods. We refer to the flowchart on the next page to get a better overview of each time step.

For each time step, we need to go through three different Newton-Raphson loops, one to solve the pressure equation $\mathcal{R}_p$, one to solve the temperature equation $\mathcal{R}_T$ and one to solve the transport equation $\mathcal{R}_\alpha$, $\alpha = w, o$ or T. We start the given time step by using the pressure, temperature and saturation values from the last time step. The first thing we do is to go through the Newton loop to find the pressure. In each Newton iteration we want to find $\Delta p = -J_p^{-1}\mathcal{R}_p$. We solve this system by finding an approximation to the Newton update with the help of the iterative multiscale method. When the multiscale cycles have converged, we can proceed to the next Newton iteration. When the Newton-Raphson method has converged for pressure, we can continue with temperature, which is found in the same way as pressure. Finally, we use the total velocity to find the saturations through the means of the transport equation, solved through the normal Newton-Raphson method. Here, we do not apply the multiscale method. When this last Newton method has converged, we have the pressure, temperature and saturation solutions for the given time step, which we in turn can use to solve the next time step. All in all, the procedures needed to solve one time step are given in the flowchart on the next page.


$p_{i=0} = p^n$
$T_{i=0} = T^n$
$s_{i=0} = s^n$
Pressure Newton Loop
Want to solve $p_{i+1} = p_i - J_p^{-1}\mathcal{R}_p(p_i, T_{i=0}, s_{i=0})$:
Iterative Multiscale:
$d^l = -\mathcal{R}_p - J\Delta p^l$
for $i = 1, \cdots, n_s$
$y^l = S(d^l)$
end
$\Delta p^{l+1} = \Delta p^l + PJ_cR(d^l - Jy^l) + y^l$
$res_{MS} = ||\mathcal{R}_p - J_p\Delta p^{l+1}||$
No ← $res_{MS} < tol_{iter}$?
Yes → multiscale converged!
$p_{i+1} = p_i - \Delta p^{l+1}$
No ← $||\mathcal{R}_p|| < tol$?
Yes → Newton converged!
$p^{n+1} = p_{i+1}$
Compute v_T etc.
Temperature Newton Loop
Want to solve $T_{i+1} = T_i - J_T^{-1}\mathcal{R}_T(p^{n+1}, T_i, s_{i=0})$:
Iterative Multiscale:
$d^l = -\mathcal{R}_T - J\Delta T^l$
for $i = 1, \cdots, n_s$
$y^l = S(d^l)$
end
$\Delta T^{l+1} = \Delta T^l + PJ_cR(d^l - Jy^l) + y^l$
$res_{MS} = ||\mathcal{R}_T - J_T\Delta T^{l+1}||$
No ← $res_{MS} < tol_{iter}$?
Yes → multiscale converged!
$T_{i+1} = T_i - \Delta T^{l+1}$
No ← $||\mathcal{R}_T|| < tol$?
Yes → Newton converged!
$T^{n+1} = T_{i+1}$
Transport Newton Loop
Choose whether to solve for $\alpha = o, w$ or T.
Using v_T,
$s_{i+1} = s_i - J_\alpha^{-1}\mathcal{R}_\alpha(p^{n+1}, T^{n+1}, s_i)$
$||\mathcal{R}_\alpha|| < tol$?
No
Yes → Newton converged!
$s^{n+1} = s_{i+1}$
p^{n+1}, T^{n+1}, s^{n+1}

5.4 EXAMPLES

We are now going to present some examples that vigorously test the multiscale method. We are going to compare the solutions resulting from the multiscale method with the solutions obtained by the sequential splitting method. Unless otherwise specified, the enthalpy, internal energy, internal energy of the rock and initial values are all defined as in the beginning of Section 4.5. The same machinery as was used in Section 4.5 is used for these new tests. To test the accuracy of the multiscale method, we use the L^2 and L^∞ norms defined in Equation (4.1) except that we now use the sequential method as the reference solution and the multiscale method as the tester,

$$e^2 = \frac{||x_S - x_{MS}||_2}{||x_S||_2} = \frac{\left(\sum_{i=1}^{n} |x_{S,i} - x_{MS,i}|^2 |\Omega_i|\right)^{1/2}}{\left(\sum_{i=1}^{n} |x_{S,i}|^2 |\Omega_i|\right)^{1/2}},$$

$$e^\infty = \frac{||x_S - x_{MS}||_\infty}{||x_S||_\infty} = \frac{\max_{i=1,\cdots,n} |x_{S,i} - x_{MS,i}|}{\max_{i=1,\cdots,n} |x_{S,i}|}.$$

We have that x_S is the sequential solution, x_{MS} is the multiscale solution and $|\Omega_i|$ gives the volume of cell i.

The basis functions will be kept fixed for all the examples given below. That is, we will only find the prolongation operator once per example, and use this operator every time step. The improvement of the accuracy when the functions were updated throughout depended on the example, but it always came at a cost of noticeable difference in run time. As the accuracy was good with the fixed basis functions, we accept the loss of precision in order to have a more efficient code.

To get a thorough test of the multiscale method, we will start by testing the method on a case with only one phase, before we expand to multiphase flow. When looking at multiphase flow, the two phases will always be oil and water, while the single-phase is oil.

5.4.1 Single-Phase

The single-phase example that will be presented is taken from my Specialization Project [30]. Data from SPE10 will be used, and as Layer 5 was used in Section 4.5, it will be used here as well. We place a horizontal well in the cells $i = 2, j = 2, \cdots, 219$, that is, along the boundary in the y-direction. The initial temperature is 300 K, while the initial pressure is 200 bar. The enthalpy, internal energy and internal energy of the rock are the single-phase restrictions of the multiphase functions defined in Section 4.5, where we have used

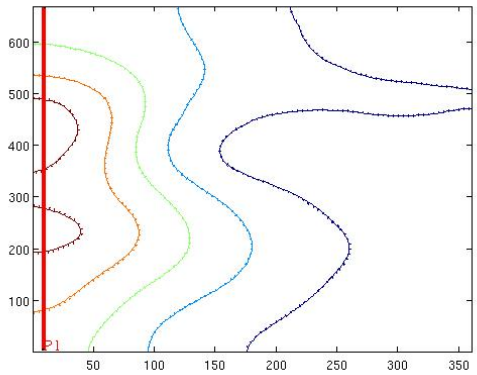
$\rho_W^s = 750$ km/m^3, $c_r = 0.5 \cdot 10^3$ and all other variables are as in Section 4.5 We let the simulation run for 550 days, and a time step of $\Delta t = 7.02$ days is used. The Newton-Raphson tolerance is 10^{-4}. We let the iterative multiscale run for 20 iterations each step, and 5 iterations of the Jacobi method have been used as a smoother. A coarse grid of $6 \times 22 \times 1$ is used. This constitutes a coarsening-factor of 100.

Anyone who has read my specialization project knows that I encountered a problem with my implementation of the iterative multiscale solver. I encountered 'spikes' in the first time steps in my temperature solution. The mistake occurred because I, instead of applying the iterative multiscale method on the equation $x_{i+1} = x_i - J_i^{-1}\mathcal{R}_i$ inside the Newton-Raphson method, applied the multiscale method with zero multiscale cycles, and then after the Newton-Raphson method had converged, applied the iterative multiscale on the converged update. This lead to an overestimation of the first time steps.

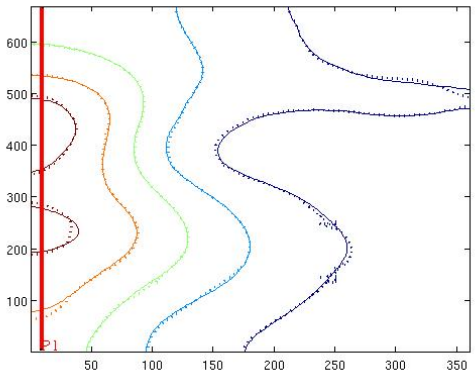
The pressure and temperature discrepancy between the correctly implemented iterative multiscale method and the sequential, fine scale, method is given in Table 5.1. Contour lines for both the sequential and multiscale temperature solution for $t = 550$ days are shown in Figure 5.4b. The sequential lines are given by the continuous lines, while the multiscale contours are dotted. Here you can also see the well, and you can clearly see how the heat spreads from the well throughout the reservoir. As one can see from the figure, the multiscale solution is not able to fully recreate the sequential solution near the well, and there are some noise on the purple contour line. The difference near the well might come from the actual well. The well will cause a lot of change, and the multiscale method is not able to pick up all the nuances. The noise is a bit unexpected, but as it occurs on the first heat wave from the well, it might indicate that the multiscale method needs a little time to adjust. Other than these two occurrences, we can be see from the table and figure that the multiscale method manages to reproduce the sequential reference solution to a large degree.

Table 5.1: The pressure and temperature L^2 and L^∞ discrepancy between the sequential method and the correctly implemented iterative multiscale method for the single-phase case.

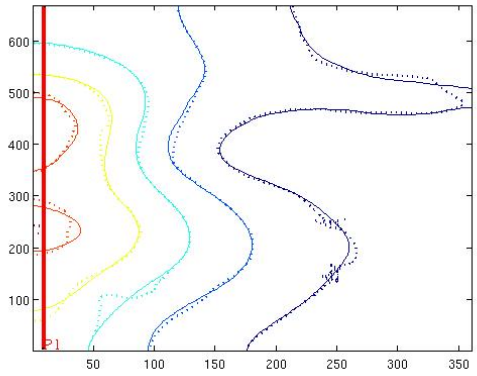
	e_T		e_p	
Time (days)	L^2	L^∞	L^2	L^∞
$t \approx 21$	$1.160 \cdot 10^{-5}$	$2.156 \cdot 10^{-4}$	$1.894 \cdot 10^{-6}$	$1.059 \cdot 10^{-5}$
$t \approx 250$	$1.989 \cdot 10^{-4}$	$2.000 \cdot 10^{-3}$	$8.242 \cdot 10^{-5}$	$1.267 \cdot 10^{-4}$
$t \approx 550$	$6.091 \cdot 10^{-5}$	$6.408 \cdot 10^{-4}$	$2.729 \cdot 10^{-5}$	$8.491 \cdot 10^{-5}$



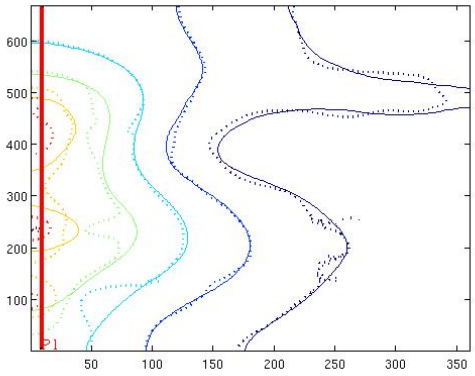
(a) Maximum iteration = 100



(b) Maximum iteration = 20



(c) Maximum iteration = 10



(d) Maximum iteration = 2

Figure 5.4: Contour lines of the multiscale (dotted lines) and sequential (continuous lines) temperature solutions for $t = 550$ days when the maximum number of iteration in the iterative multiscale method is 100, 20, 10 and 2.

It is important to check that the spikes we encountered in the specialization project actually have gone. Figure 5.5 shows the maximum and minimum temperature for each day for the sequential method and the correct multiscale method. As one can see, the two solutions closely resemble each other and the spikes have completely disappeared.

To check that the method works as expected, we check what happens when the maximum number of iterations change, and what happens when we change the dimensions on the coarse grid. The accuracy between the sequential method and the multiscale method tends to improve as the maximum number of iterations increases, see Figure 5.4. When the opposite is true, the discrepancy increase. Table 5.2 shows the temperature and pressure discrepancy between the sequential and multiscale method when the maximum number of iterations was set to 10. Comparing with the results in Table 5.1, where the maximum

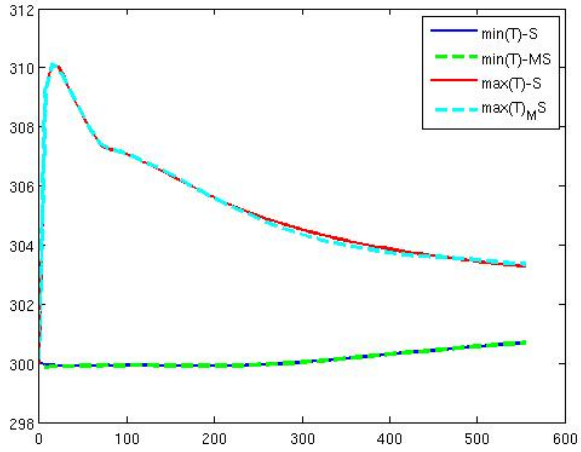


Figure 5.5: The maximum and minimum temperature per day generated by the sequential (S) method and the multiscale (MS) method.

iteration number was 20, we see that we now have a higher error between the two methods.

Table 5.2: Pressure and temperature discrepancy between the sequential and multiscale method for $t \approx 21, 250, 550$ days, when the maximum number of iterations are 10. The temperature discrepancies are given by e_T, while the pressure discrepancies are given by e_p.

	e_T		e_p	
Time (days)	L^2	L^∞	L^2	L^∞
$t \approx 21$	$2.243 \cdot 10^{-5}$	$3.666 \cdot 10^{-4}$	$6.638 \cdot 10^{-6}$	$4.207 \cdot 10^{-5}$
$t \approx 250$	$2.707 \cdot 10^{-4}$	$3.313 \cdot 10^{-3}$	$1.760 \cdot 10^{-4}$	$2.769 \cdot 10^{-4}$
$t \approx 550$	$1.364 \cdot 10^{-4}$	$1.497 \cdot 10^{-3}$	$8.022 \cdot 10^{-5}$	$3.695 \cdot 10^{-4}$

This can also be seen when comparing the sequential temperature solutions for $t = 550$ days with the multiscale temperature solution for the same time found by using different maximum iteration numbers. This can be done by studying Figure 5.4, which depicts four contour plots, one where the maximum number of iterations is 100, one where it is 20, one where it is 10 and one where it is 2. The sequential solutions are given by the continuous lines, while the multiscale solutions are dotted. Though the temperature solution found by the multiscale method when the maximum number of iterations is 10 still resembles the sequential solution, the results are in no way as good as before. It is clear that the two contour lines do not align. By reducing the maximum number of iterations further, the two solutions are even more spread apart, as can be seen in the bottom right plot in Figure 5.4, where the maximum number of iterations is 2. We also have that the noise and error by the well we encountered when 20 iterations were used has increased. So, all in all, we have that by decreasing the maximum number of iterative multiscale iterations, the accuracy decreases,

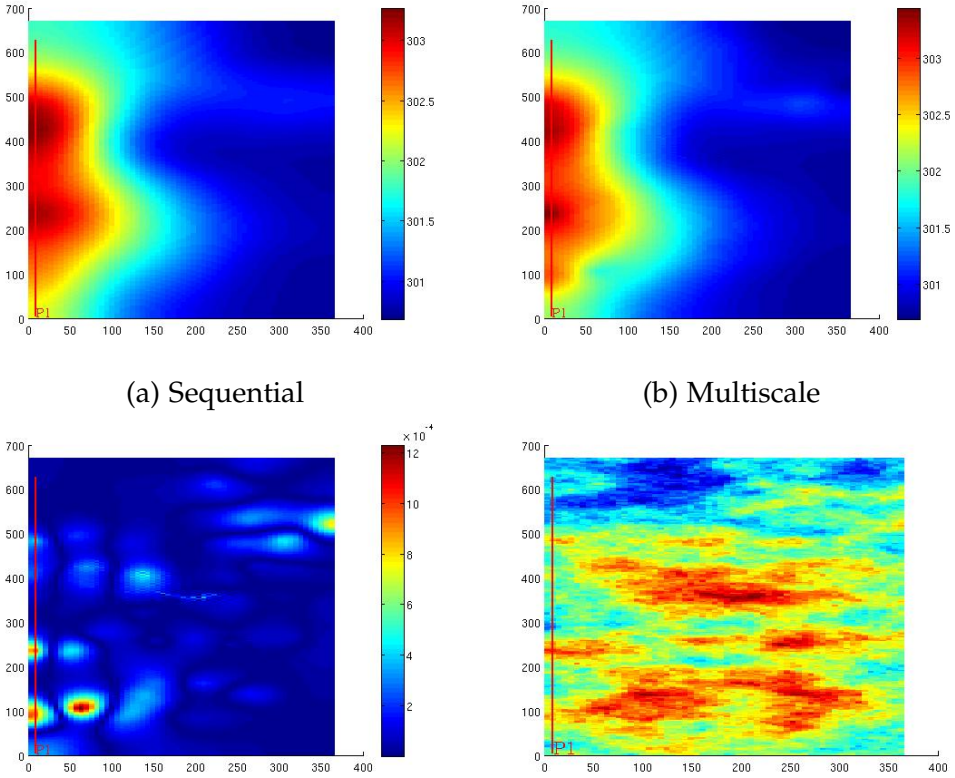
just as one would expect. If we increase the maximum number of iterations the accuracy improves, as can be seen by studying the upper left plot in Figure 5.4. Here we can see that when the 100 iterations are used, the sequential method and multiscale method perfectly align.

We go back to the standard maximum iteration number of 20. When we change the dimensions of the coarse grid, it is reasonable to assume that the accuracy will change as well, at least if we keep the maximum number of iterations fixed. Looking at Table 5.3, where we have used a coarse dimension of $5 \times 10 \times 1$ we see that the assumption is, indeed, correct, but mostly for the last time step. The coarse grid used to generate the two columns to the right in Table 5.3 is coarser than the grid used to generate the two columns to the left, and this last one gives the same discrepancies as in Table 5.1. As all other parameters remain unchanged, we can see that making the coarse grid coarser in this example, gives a very small increase in the discrepancy between the sequential method and the multiscale method. If we did not have a fixed value for the iterations, and instead let the method run until it reaches a preset tolerance, the method's discrepancy would not be much affected by a coarser grid. The method would converge even for very coarse grids, but it would need more iterations to do so. This will be discussed in a later example.

The minimal change in the discrepancy for this example indicates that the multiscale method's performance is good even when quite a coarse grid is used and the number of iterations is fixed. Though it is important to remember that we have used a relatively high value for the number of iterations here which enforce good results. The sequential and multiscale temperature solution, as well as the discrepancy between them, for $t = 550$ days is shown in Figure 5.6. This also shows that the method remains good when coarser grids are applied. Even though there is a discernible difference between the multiscale solution and the sequential reference solution, the difference is small. Most of the difference between the two methods seems to arise because of the well. We can also see that there are some correlation between the error and the permeability field. When we make the coarse scale finer, the accuracy improves, the discrepancy between the two methods decreases.

Table 5.3: The sequential and multiscale temperature discrepancy on two different coarse grids.

	$6 \times 22 \times 1$		$5 \times 10 \times 1$	
Time (days)	L^2	L^∞	L^2	L^∞
$t \approx 21$	$1.160 \cdot 10^{-5}$	$2.156 \cdot 10^{-4}$	$1.305 \cdot 10^{-5}$	$2.321 \cdot 10^{-4}$
$t \approx 250$	$1.989 \cdot 10^{-4}$	$2.000 \cdot 10^{-3}$	$4.407 \cdot 10^{-4}$	$5.173 \cdot 10^{-3}$
$t \approx 550$	$6.091 \cdot 10^{-5}$	$6.408 \cdot 10^{-4}$	$1.421 \cdot 10^{-4}$	$1.227 \cdot 10^{-3}$



(a) Sequential

(b) Multiscale

(c) Error

(d) Permeability

Figure 5.6: Sequential and multiscale temperature solution, as well as the difference between them, when $t = 550$ days, generated with a coarse grid with dimensions $5 \times 10 \times 1$. The permeability in logarithmic scale of the layer is also displayed.

5.4.2 Homogeneous Permeability

When extending the multiscale method to multiphase flow, we are first going to consider the simple test case from Section 4.5, namely the reservoir with the homogeneous permeability. The dimensions, well placements, initial values and all other parameters are as in Section 4.5.

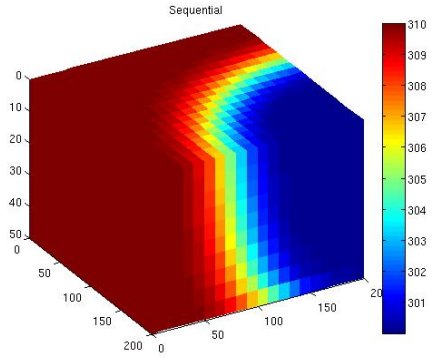
The discrepancy for $t = 20, 100, 200$ days between the sequential and multiscale pressure and temperature solution is given in Table 5.4. A coarse grid with dimensions $5 \times 5 \times 5$ is used. This constitutes a coarse grid with 125 grid blocks, making it 32 times as coarse as the fine grid. The time step used was $\Delta t = 20$ days, the simulation ran for 200 days, and the sequential tolerance was 10^{-6}. The multiscale method uses a Newton-Raphson method which has a tolerance of 10^{-6}. For each Newton iteration, it applies the iterative multiscale method, which runs until an *iterative tolerance* is reached. The first three rows in Table 5.4 are the result of an iterative tolerance of 10^{-1}. The table shows that the discrepancy between the two methods is quite small for both the pressure and

Table 5.4: The discrepancy between the sequential solution and the multiscale solution for both temperature and pressure for the homogeneous permeability. Two different values for the iterative tolerance tol_{iter} were used, as well as a time step of $\Delta t = 20$.

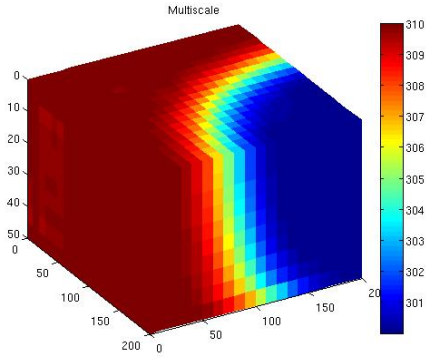
		e_T		e_p	
tol_{iter}	Time (days)	L^2	L^∞	L^2	L^∞
10^{-1}	t = 20	$9.045 \cdot 10^{-5}$	$5.071 \cdot 10^{-4}$	$1.112 \cdot 10^{-5}$	$3.122 \cdot 10^{-5}$
	t = 100	$1.381 \cdot 10^{-4}$	$8.272 \cdot 10^{-4}$	$1.331 \cdot 10^{-5}$	$6.774 \cdot 10^{-5}$
	t = 200	$2.753 \cdot 10^{-4}$	$2.519 \cdot 10^{-3}$	$3.749 \cdot 10^{-6}$	$1.351 \cdot 10^{-5}$
10^{-6}	t = 20	$6.127 \cdot 10^{-11}$	$4.745 \cdot 10^{-10}$	$4.259 \cdot 10^{-12}$	$3.607 \cdot 10^{-12}$
	t = 100	$3.625 \cdot 10^{-10}$	$1.465 \cdot 10^{-9}$	$3.796 \cdot 10^{-10}$	$1.115 \cdot 10^{-9}$
	t = 200	$2.520 \cdot 10^{-10}$	$1.067 \cdot 10^{-9}$	$1.429 \cdot 10^{-10}$	$3.661 \cdot 10^{-10}$

temperature solutions. However, when studying the second plot in Figure 5.7 we see that the multiscale method has not completely managed to converge to the sequential solution. The figure gives the temperature solution when $t = 200$ days for the sequential and multiscale method, and, as can be seen, the multiscale solution differs slightly from the sequential solution when the iterative tolerance is 10^{-1}. This can be corrected for, however, by decreasing the iterative tolerance, see the last three rows in Table 5.4 and the third plot in Figure 5.7. An iterative tolerance of 10^{-6} has now been used, resulting in a drastically decreased discrepancy, and a multiscale and sequential temperature solution that are identical to the naked eye. The figure also display the discrepancy between the two methods for the last time for the two different iterative tolerances. Most of the error seems to occur on the two sides that make the corner where the production well is placed. Note the smaller order on the error when we use the lower iterative tolerance. The multiscale method is thus, as can be seen, a good method when the iterative tolerance is 10^{-1}, but it is even better when we use $tol_{iter} = 10^{-6}$. It is important to remember that a homogeneous permeability is the worst case scenario for the multiscale method, as there are no small-scale structures for the prolongation operator to customize to. So the fact that the method works well is reassuring.

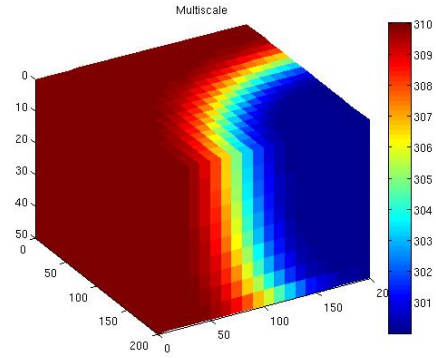
Remember from Section 4.4 that we can add outer iterations in order to improve the sequential method compared to the fully implicit method. It is therefore important to study the affect on outer iterations on the multiscale method as well, to see if this make the multiscale solutions converge towards the fully implicit solutions. There are, as previously stated, two places where we can add outer iterations. We can have n_1 iterations after the temperature system, and n_2 iterations after the transport equation.



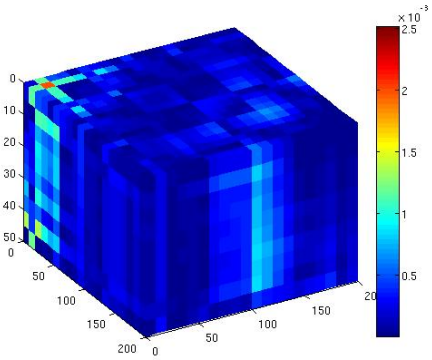
(a) Sequential



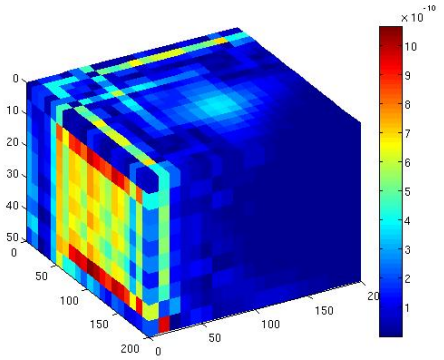
(b) Multiscale, $tol_{iter} = 10^{-1}$



(c) Multiscale, $tol_{iter} = 10^{-6}$



(d) Error, $tol_{iter} = 10^{-1}$



(e) Error, $tol_{iter} = 10^{-6}$

Figure 5.7: The temperature solution for the last time step, $t = 200$ days, for the sequential method and for the multiscale method, where two different iterative tolerances have been used. The first used a tolerance of 10^{-1}, while the other used a tolerance of 10^{-6}. The discrepancy between the sequential solution and the multiscale solution for the same time for the two different iterative tolerances are also depicted.

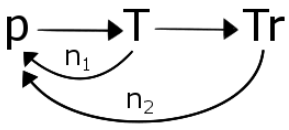
We will study the cases from Section 4.5,

CASE 1: $n_1 = 1, n_2 = 0$

CASE 2: $n_1 = 1, n_2 = 1$

CASE 3: $n_1 = 0, n_2 = 2$

CASE 4: $n_1 = 1, n_2 = 2$.



It turns out, that if we use a low iterative tolerance, then the outer iterations will improve the multiscale solutions compared to the sequential solutions and therefore also the fully implicit solutions. This is due to the fact that the iterative tolerance is high, and the discrepancy will therefore change when the systems are resolved. This does not really tell us much about the overall effect of the outer iterations though, it is just a side effect of the iterative tolerance, and the results are therefore omitted for brevity. If we use a lower iterative tolerance,

the solutions are affected, but not to the same degree, see Table 5.5. The table shows the pressure and temperature discrepancy between the sequential and multiscale methods for the different cases when a coarse grid of $[5, 5, 5]$ is used, the iterative tolerance is 10^{-3} and the Newton-Raphson tolerance is 10^{-6}. The discrepancies are shown for $t = 20, 100, 200$ days, a time step of $\Delta t = 20$ days is used, and the simulations ran for 200 days. The run time of both the sequential method and the multiscale method for the different cases are also displayed in the table. As can be seen, the discrepancies actually increase with the first case. As this case did not improve the sequential method compared to the fully implicit method to a large degree either, this is clearly not the case to use. In Section 4.4 we found that CASE 2 and 3 were the best cases for the sequential method considering the accuracy and efficiency. Looking at Table 5.5, we see that the discrepancies between the multiscale method and the sequential method are relatively unchanged compared to the original case for these same cases. Thus, if we wanted to improve the accuracy of the multiscale method compared to the fully implicit method, while still keeping the multiscale method efficient, these two cases should be considered.

Table 5.5: Temperature and pressure discrepancies between the sequential (S) and multiscale (MS) solvers, measured in the L_2 norm for $t = 20, 100, 200$ days, as well as the run time for the two methods for the all the different cases. The temperature discrepancy is given by e_T, while e_p denotes the pressure discrepancy.

	Time (days)	ORIGINAL	CASE 1	CASE 2	CASE 3	CASE 4
	$t = 20$	$1.234 \cdot 10^{-8}$	$6.450 \cdot 10^{-10}$	$5.222 \cdot 10^{-8}$	$1.497 \cdot 10^{-8}$	$6.515 \cdot 10^{-10}$
e_T	$t = 100$	$6.415 \cdot 10^{-8}$	$1.438 \cdot 10^{-6}$	$8.309 \cdot 10^{-8}$	$2.412 \cdot 10^{-7}$	$8.947 \cdot 10^{-9}$
	$t = 200$	$4.363 \cdot 10^{-8}$	$1.316 \cdot 10^{-6}$	$5.468 \cdot 10^{-8}$	$3.343 \cdot 10^{-6}$	$1.196 \cdot 10^{-7}$
	$t = 20$	$1.343 \cdot 10^{-9}$	$7.450 \cdot 10^{-11}$	$5.238 \cdot 10^{-9}$	$1.650 \cdot 10^{-9}$	$1.073 \cdot 10^{-10}$
e_p	$t = 100$	$1.091 \cdot 10^{-9}$	$2.338 \cdot 10^{-8}$	$2.781 \cdot 10^{-9}$	$6.457 \cdot 10^{-9}$	$3.418 \cdot 10^{-10}$
	$t = 200$	$2.301 \cdot 10^{-10}$	$6.008 \cdot 10^{-9}$	$4.793 \cdot 10^{-10}$	$4.415 \cdot 10^{-8}$	$2.720 \cdot 10^{-9}$
Run time S (sec)		11.2	16.4	28.7	24.6	40.4
Run time MS (sec)		10.8	16.6	29.4	24.6	41.4

Notice that the run time of the sequential and multiscale method is almost identical for all cases. In Section 4.5 we found the run time of the fully implicit method to be approximately 33 seconds. The table shows that both methods are quicker than the fully implicit method for the three first cases, as well as for the ORIGINAL case where no outer iterations are used. This is a relatively small system, and as both MATLAB and automatic differentiation have less of a focus on efficiency, is it not to be expected that the methods are particularly effective for this example. The fact that the sequential and multiscale methods still outperform the fully implicit method is therefore positive. This is the end

of the discussion of outer loops, and in the examples that follow, we will use $n_1 = n_2 = 0$ for all.

Applying the multiscale method on a model with a fully homogeneous permeability field holds to an extent no real value. The method uses the permeability field when the basis functions are formed. When we have a homogeneous permeability field, the basis functions will be uniform functions and applying the multiscale method will thus be the same as solving the solution on a coarse grid, and then just expand to a fine grid. That is all well and good, but there are many methods that does the same. It is therefore more interesting to see what happens when we apply the multiscale method on a model with a changing permeability field.

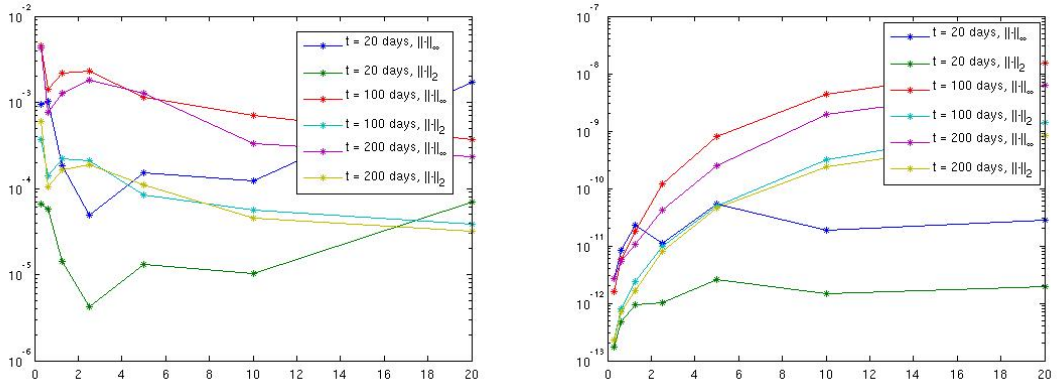
5.4.3 SPE 10, Layer 5

Again we are going to study the two phase model given by SPE 10, layer 5. Remember from before that the fine scale system (the sequential system) has $60 \times 220 \times 1$ cells, as we are only considering one layer. This means that the fine scale system has 13,200 fine cells. We are going to start with a coarse scale partition of $10 \times 20 \times 1$ blocks. We thus have a coarsening factor of 66. The discrepancies between the pressure and temperature solutions found with the multiscale method and the sequential, fine scale method are given in Table 5.6. The time step used was $\Delta t = 20$ days, and the discrepancies are shown for $t = 20, 100, 200$ days. The simulation ran for 200 days, the sequential tolerance was $tol = 10^{-6}$, as was the Newton tolerance for the multiscale method. The iterative multiscale tolerance was 10^{-1}. As can be seen from the table, the multiscale method mimics the sequential method. Especially the pressure solution is good, but the temperature solution is acceptable as well.

Table 5.6: Temperature and pressure discrepancy for Layer 5. The discrepancy in temperature is given by e_T, while the pressure discrepancy is given by e_p. The time step is $\Delta t = 20$ days, the Newton tolerance is $tol = 10^{-6}$ and the iterative tolerance is 10^{-1}.

	Time (days)	L^2	L^∞
e_T	t = 20	$6.962 \cdot 10^{-5}$	$1.727 \cdot 10^{-3}$
	t = 100	$3.795 \cdot 10^{-5}$	$3.659 \cdot 10^{-4}$
	t = 200	$3.222 \cdot 10^{-5}$	$2.366 \cdot 10^{-4}$
e_p	t = 20	$1.002 \cdot 10^{-5}$	$6.200 \cdot 10^{-5}$
	t = 100	$1.276 \cdot 10^{-6}$	$3.005 \cdot 10^{-6}$
	t = 200	$1.128 \cdot 10^{-6}$	$5.652 \cdot 10^{-6}$

Test of Δt: In Section 4.5, we found that the results were good even when relatively large time steps were used. We therefore chose to use $\Delta t = 20$ days to generate Table 5.6. It is interesting to study what happens when we use smaller time steps as well though, which we can do by studying Figure 5.8. The figure shows the temperature discrepancy between the sequential and multiscale method for $t = 20, 100, 200$ days when $\Delta t = 0.3125, 0.625, 1.25, 2.5, 5, 10, 20$ days, and as you can see, the discrepancies increase when Δt is very small and $tol_{iter} = 10^{-1}$. The error depends on how accurately we solve the linear system, and it is therefore natural that the error accumulate when we have a higher number of time steps. The smaller Δt will therefore lead to a higher discrepancy. When we use a stricter iterative tolerance, $tol_{iter} = 10^{-6}$, Figure 5.8 shows that the discrepancies mostly decrease as the time step decreases. The smaller tolerance is thus able to dispose of the accumulated error. The impact of the iterative tolerance will be further discussed below.

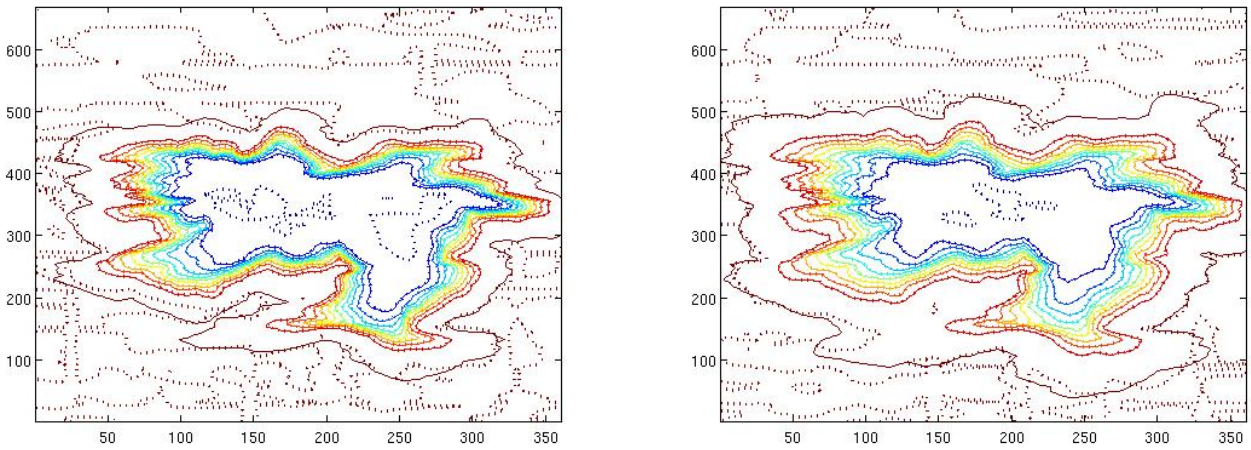


(a) $tol_{iter} = 10^{-1}$ (b) $tol_{iter} = 10^{-6}$

Figure 5.8: Discrepancies between the sequential temperature solution and the multiscale temperature solution as a function of different time step sizes for two different values for tol_{iter}. The L^2 and L^∞ errors for $t = 20, 100, 200$ days are shown for $\Delta t = 0.3125, 0.625, 1.25, 2.5, 5, 10, 20$ days.

Contour plots of the sequential and multiscale temperature solution for the last time step, $t = 200$ days, when $\Delta t = 5$ days and $\Delta t = 20$ days are given in Figure 5.9. The iterative tolerance is 10^{-1}. All other parameters are unchanged. The sequential contour lines are given as continuous lines, while the multiscale contour lines are dotted. The figure gives a lot of information. First, notice that the multiscale solution has contour lines that the sequential solutions does not have, both when $\Delta t = 5$ days and when $\Delta t = 20$ days. These lines are especially apparent outside the main temperature change region, the circular region where the contour lines are closest. But the additional multiscale lines are also present inside the belt. These lines are easy to get rid of though. We have used a fairly high tolerance for the iterative tolerance, a $tol_{iter} = 10^{-1}$ was used and by

having a smaller tolerance the excess contour lines will disappear. This will be discussed further below. The figure also shows that if we disregard the excess lines, the multiscale solution gives a good approximation to the sequential solution. It is easy to see that the two solutions match closely in the belt for both time steps. You can hardly see the difference between the sequential solution and the multiscale solution in the belt area. The last thing to note is the fact that it might seem like the multiscale method gives a better solution when $\Delta t = 20$ days than when $\Delta t = 5$ days. Looking again at Figure 5.8, we see that this is in line with previous findings. Figure 5.8 shows that at time $t = 200$ days, the multiscale solution is in fact a bit better when $\Delta t = 20$ days is used, than when $\Delta t = 5$ days.



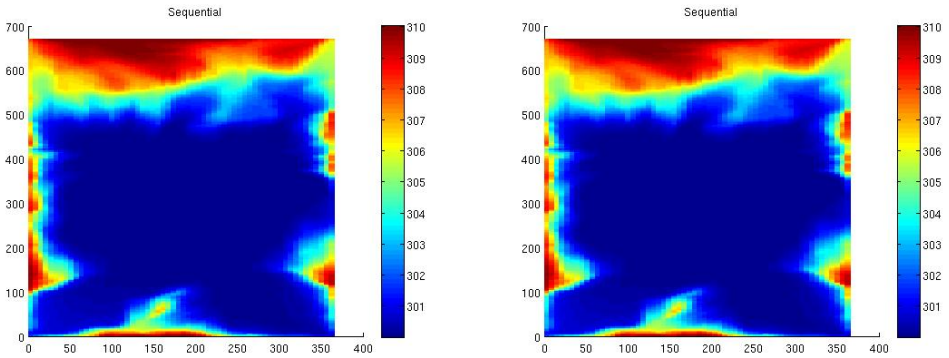
(a) $\Delta t = 5$ days

(b) $\Delta t = 20$ days

Figure 5.9: Contour plots of the multiscale and sequential temperature for $t = 200$ days. Two different step sizes have been used, $\Delta t = 5$ days and $\Delta t = 20$ days. The contour lines for the sequential solutions are given as continuous lines, while the contour lines for the multiscale solutions are dotted. The iterative tolerance is 10^{-1}.

Remember from Section 4.5 that while we found there to be a bound on the fully implicit step size Δt, the same did not hold true for the sequential step size. Similarly, we have not found an upper bound for the step size with the multiscale method. We can therefore look at how the multiscale method performs when $\Delta t = 200$ days. We let the simulation run for ten time steps, or until $t = 5$ years, 175 days. The other parameters have not changed. The temperature solutions for $t = 200, 1000, 2000$ days are given in Figure 5.10. The figure shows that the multiscale method gives a good approximation to the sequential solution, but the two methods are not identical. This is also shown in Table 5.7, where the temperature discrepancy is displayed for the same times. We can see from the table that the multiscale method is a good solver, especially considering the high iterative tolerance, which is 10^{-1}. The MsRSB method can

to a large degree be considered a more accurate upscaling method with this tolerance. The accuracy decreases as time goes on, but, as can be seen from the figure, the multiscale solution is still able to imitate the sequential solution for the last time step. By decreasing the iterative tolerance, the two solutions will converge. As a sidebar, it is interesting to note that it takes a long time for the heat from the injection well to spread throughout the reservoir. The temperature in the well is 300 K, and even after almost 5.5 years the well-temperature has not spread all the way through the reservoir.



(a) Sequential solution (b) Multiscale solution

Figure 5.10: Sequential and multiscale temperature solution for $t = 2000$ days when $\Delta t = 200$ days.

Table 5.7: Temperature discrepancy for Layer 5. A time step of $\Delta t = 200$ days has been used. The Newton tolerance was $tol = 10^{-6}$ and the iterative tolerance was 10^{-1}.

Time (days)	L^2	L^∞
t = 200	$7.815 \cdot 10^{-6}$	$8.980 \cdot 10^{-3}$
t = 1000	$4.639 \cdot 10^{-4}$	$3.982 \cdot 10^{-3}$
t = 2000	$3.323 \cdot 10^{-4}$	$2.593 \cdot 10^{-3}$

Change of tolerance: Up to now, the tolerance of the iterative multiscale has mostly been 10^{-1}. Changing the tolerance to 10^{-6} has a dramatic effect on the multiscale solution. Table 5.8 gives the temperature discrepancy between the sequential and multiscale method for $t = 20, 100, 200$ days, when $\Delta t = 20$ days and $tol_{iter} = 10^{-6}$. All other variables remain the same. The table shows a dramatic decrease in the error compared to Table 5.6. It is clear that the multiscale method manages to recreate the sequential solution to a large degree. If we also look at contour plots, Figure 5.11, which displays contour plots for the six iterative tolerances $tol_{iter} = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$, we see that when $tol_{iter} = 10^{-6}$ (Figure 5.11f) all excess contour lines from $tol_{iter} = 10^{-1}$ (Figure 5.11a) are gone. In fact, the sequential and multiscale solutions look

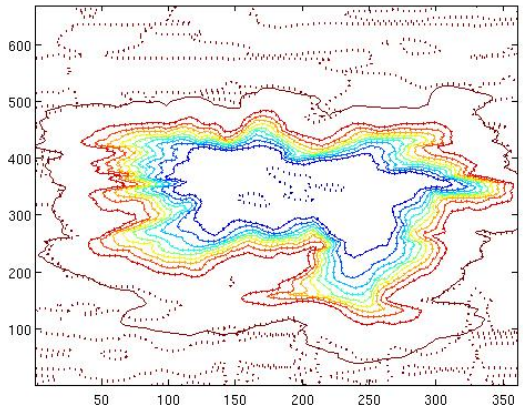
Table 5.8: Temperature discrepancy for Layer 5, when $t = 20, 100, 200$ days. A time step of $\Delta t = 20$ days has been used, while the iterative tolerance was 10^{-6}.

Time (days)	L^2	L^∞
t = 20	$1.674 \cdot 10^{-10}$	$2.778 \cdot 10^{-9}$
t = 100	$2.082 \cdot 10^{-10}$	$1.718 \cdot 10^{-9}$
t = 200	$1.440 \cdot 10^{-10}$	$1.234 \cdot 10^{-9}$

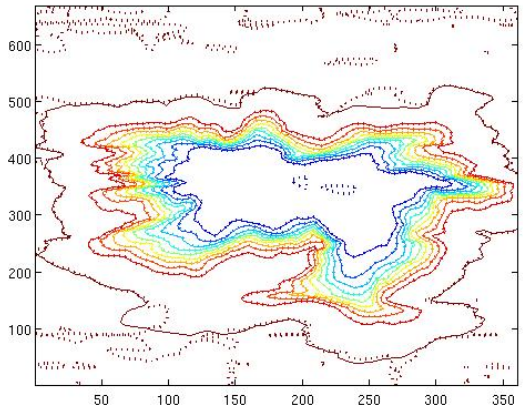
identical when $tol_{iter} = 10^{-6}$, as the lines for the two solutions align. When $\Delta t = 5$ days and $\Delta t = 200$ days the same occurred. An iterative tolerance of 10^{-6} is quite small, and as can be seen from the rest of the contour plots, the noise disappears when an iterative tolerance of 10^{-3} is used. In fact, the four last plots look identical. It is therefore not a necessity to use such a small iterative tolerance as 10^{-6}.

Choosing the iterative tolerance should be based on one's needs. If we are looking for a method that removes the noise from our multiscale solution, an iterative tolerance of 10^{-3} is sufficient, at least for this example. Changing the iterative tolerance in this example does not affect the run time to a large degree though, so if we want a method that most accurately mimics the sequential solution, a smaller tolerance should be used, see Figure 5.12. The figure shows the multiscale and sequential temperature discrepancy as a function of the iterative tolerance for different times. The time step was $\Delta t = 20$ days. Again we see that the discrepancy drastically decreases as the tolerance decreases. In fact, as the iterative tolerance decreases, the multiscale solution converges towards the sequential solution to computer precision for $t = 20$ days. Note also that the errors are close for all t. The figure also shows a steep descent between the 10^{-2} and 10^{-3} tolerance. This fits with what we found in Figure 5.11, where we saw that the noise in the multiscale solution disappears when we go from a tolerance of 10^{-2} to a tolerance of 10^{-3}.

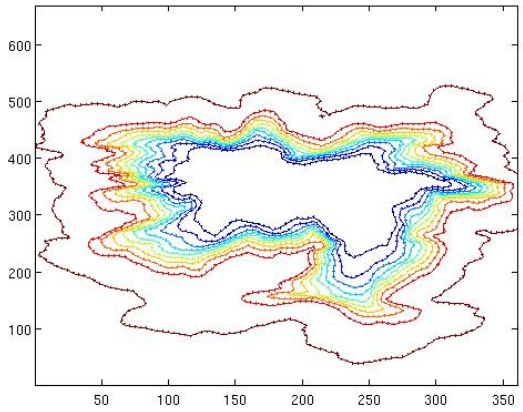
Multiscale iterations: Let us study the number of multiscale iterations per time step. Remember that for each time step we apply a Newton iteration that runs until convergence, and for each Newton iteration, we apply the iterative multiscale until convergence. So, for each time step, the multiscale method might use different number of iterations to converge, one for each Newton-Raphson iteration. We will therefore consider an average. We will look at the average number of iterations it takes for the multiscale method to converge for each time step. Figure 5.13 shows the average number of multiscale iteration needed to solve the pressure system $\mathcal{R}_p$ when the iterative tolerance is 10^{-6} and 10^{-1}, and the average number of iterations needed to solve the temperature system $\mathcal{R}_T$ when the iterative tolerance is 10^{-6}. A time step of $\Delta t = 20$ days was used, and the simulation ran for 10 time steps.



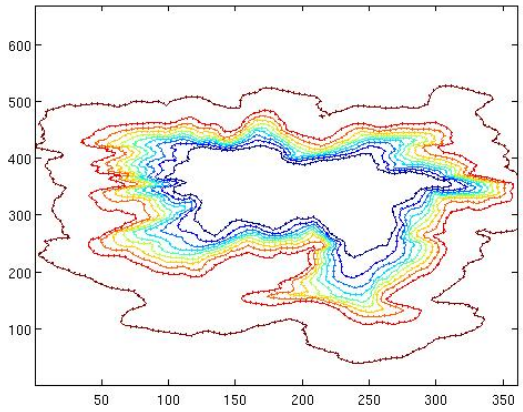
(a) $tol_{iter} = 10^{-1}$



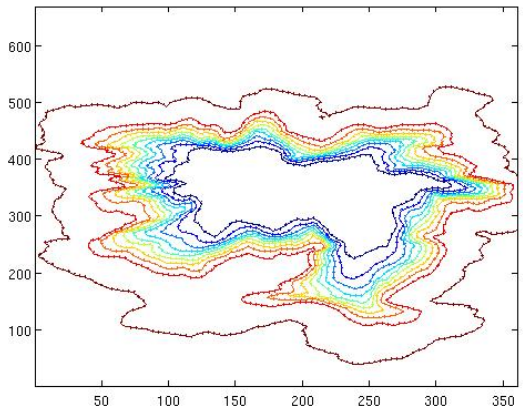
(b) $tol_{iter} = 10^{-2}$



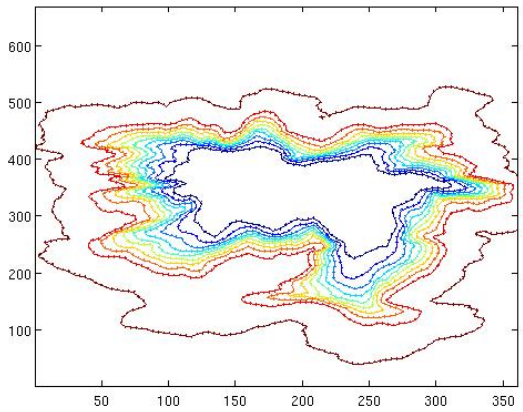
(c) $tol_{iter} = 10^{-3}$



(d) $tol_{iter} = 10^{-4}$



(e) $tol_{iter} = 10^{-5}$



(f) $tol_{iter} = 10^{-6}$

Figure 5.11: Contour plot of the multiscale temperature solution (dotted lines) and the sequential temperature solution (continuous lines) for $t = 200$ days, when $\Delta t = 20$ days for different iterative tolerances tol_{iter}.

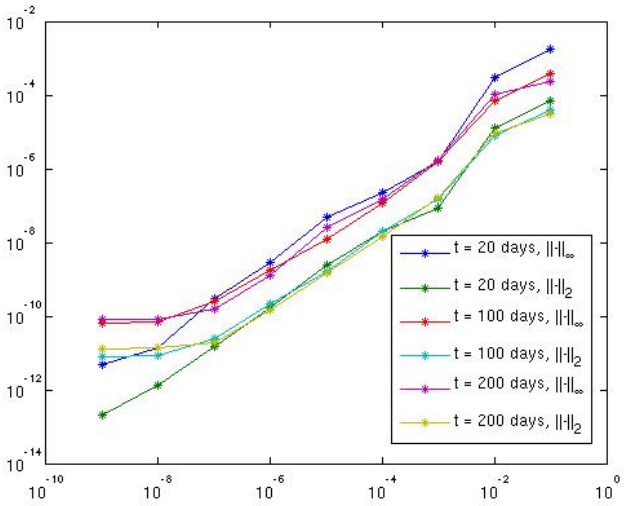
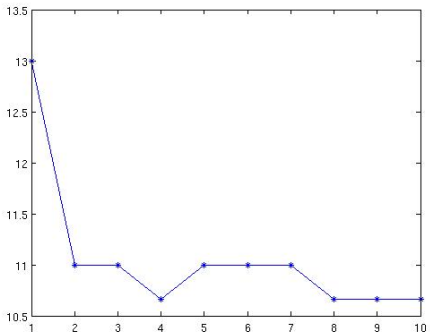
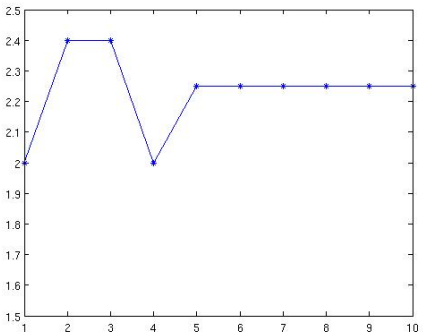


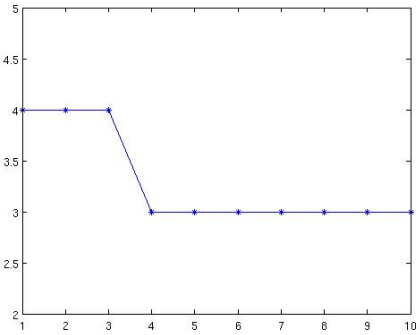
Figure 5.12: Temperature discrepancies between the sequential and multiscale temperature solutions as a function of the iterative tolerance. The time step used was $\Delta t = 20$ days.



(a) Pressure, $tol_{iter} = 10^{-6}$



(b) Pressure, $tol_{iter} = 10^{-1}$



(c) Temperature, $tol_{iter} = 10^{-6}$

Figure 5.13: Average number of multiscale iterations per time step. The pressure iterations shows the average iterations to solve the pressure system $\mathcal{R}_p$, while the temperature iterations shows the average iterations to solve the temperature system $\mathcal{R}_T$. A time step of $\Delta t = 20$ days was used, and the simulation ran for 200 days.

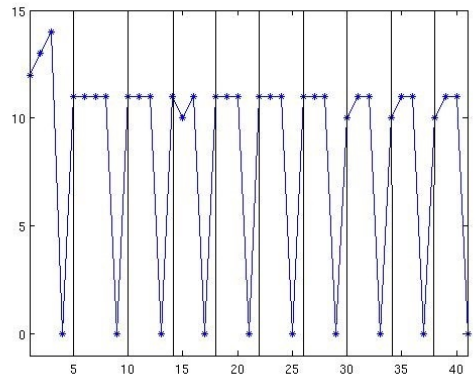
As can be seen, the multiscale method uses fewer iterations to find the temperature solution than it does to find the pressure solution. When the iterative tolerance is 10^{-1}, the method uses an average of 1 iteration per time step to find the temperature solution. This is the case both for a time step of $\Delta t = 20$ days and when $\Delta t = 5$ days. When a smaller iterative tolerance is used, the number of iterations increases slightly, but not by much. When the time step is $\Delta t = 5$ days, the method uses an average of 2 iterations per time step, while a time step of $\Delta t = 20$ days leads to 4 iterations in the beginning, before it changes to 3 iterations, see Figure 5.13c. The number of iterations is higher for the larger time step because when a smaller time step is used, the solution will be closer to the solution from the last time step, as the system has not had a lot time to vigorously change. The method will therefore find the solution to the next time step quicker when smaller time steps are used, and the number of iterations will be smaller.

When it comes to pressure, we see that the method uses more iterations. The basis functions are kept fixed for both the pressure and temperature system, so this should not affect the difference in the number of iterations. We especially have many pressure iterations when we use a small iterative tolerance, see Figure 5.13a, but the number of iterations is also higher than the temperature iterations when the tolerance is bigger, Figure 5.13b. We see the same tendencies between the big and small time steps as with temperature, although we have again skipped the plots. But, shorter time steps leads to fewer iterations, both for the small and bigger tolerance.

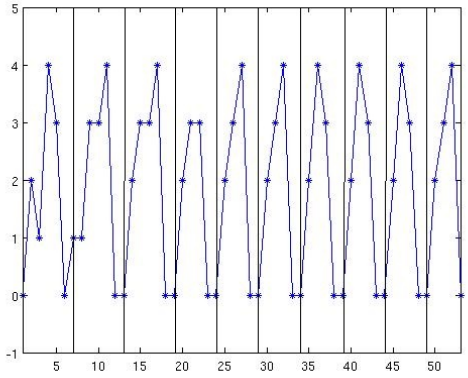
We see again that the smaller tolerance leads to a higher number of iterations per time step, which is natural, as the solver needs more iterations to reach a smaller tolerance. Notice the dip in pressure iteration at time step 4, i.e., after 80 days. This corresponds to the time step where the temperature iteration decreases as well. The multiscale method will use more iterations in the beginning, as it does not have previous time steps to build on, but will, generally, decrease and stabilize as time goes on. The higher iteration number in the beginning might also be due to initial transients in the solution that dissipate after a while. The temperature system clearly uses more iterations in the beginning, and the same tendencies are present for the pressure iterations as well, though it does have a dip at the forth time step. The fact that the multiscale method needs more iterations to solve the pressure system than it needs to solve the temperature system might come from the fact that the pressure system includes the wells, and there can be some difficulties with the link between the reservoir and wells. We must also remember that the temperature system uses the updated pressure, while the pressure system uses the temperature from the last time step. The solver will therefore need longer time to solve for pressure.

We have also included a figure that shows the multiscale iterations for each Newton iteration, Figure 5.14. The figure shows the iterations needed for both

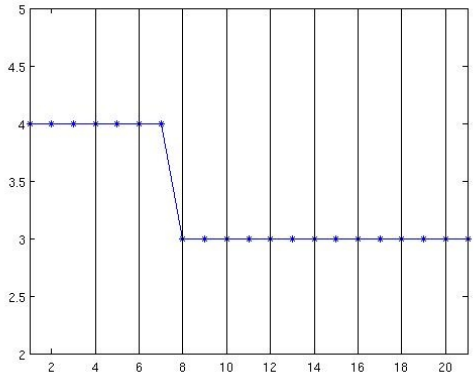
pressure and temperature, for both $tol_{iter} = 10^{-6}$ and $tol_{iter} = 10^{-1}$. A new time step is indicated by a solid black line. Again we see that we need a higher number of iterations when the iterative tolerance is strict. We also see that while the temperature iterations remain constant for each Newton iteration in a time step, the pressure system has more alteration. Though we do have the same tendencies for each time step for the pressure system. For the pressure system, we have a Newton iteration with zero multiscale iterations at the end of each time step. This is because we at the end of each time step go through a velocity reconstruction and this does not need any multiscale iterations to converge for this example. Other than this zero iteration, we see that when the stricter tolerance is used, $tol_{iter} = 10^{-6}$, the multiscale iterations remain relatively constant for each Newton iteration. The method must work more to reach the tolerance, and all the multiscale iterations will be relatively high. This is not the case when $tol_{iter} = 10^{-1}$ is used, and we see that there are more variety for each Newton iteration for this case. We can also note that both the pressure and temperature system use a higher number of Newton iterations in the beginning of the simulation than at the end.



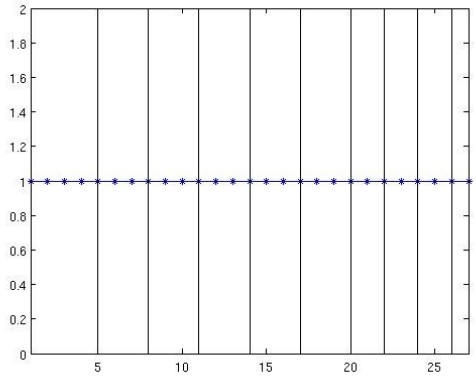
(a) Pressure, $tol_{iter} = 10^{-6}$



(b) Pressure, $tol_{iter} = 10^{-1}$



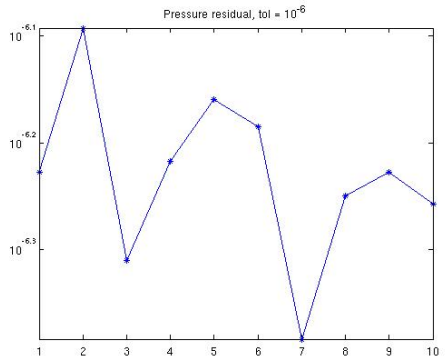
(c) Temperature, $tol_{iter} = 10^{-6}$



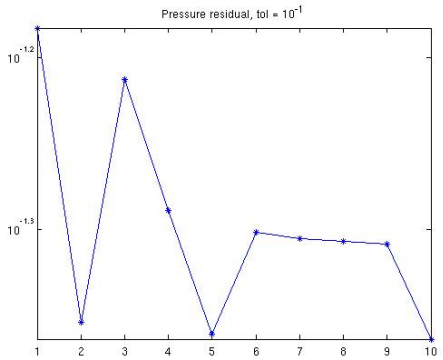
(d) Temperature, $tol_{iter} = 10^{-1}$

Figure 5.14: Multiscale iterations for each Newton iteration. A new time step is shown with the solid black lines.

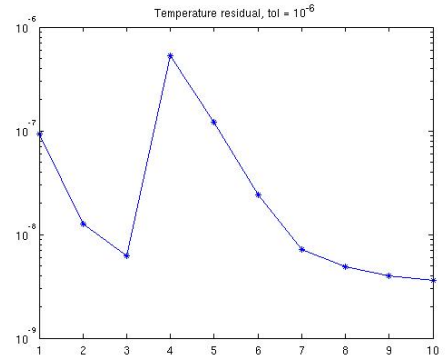
Residuals: The average pressure and temperature residual for the final multiscale iteration over the Newton iteration for each simulation step is given in Figure 5.15. The figures are generated using a time step of $\Delta t = 20$ days. When looking at Figure 5.15c, note the larger residual for time step number 4. Remember that time step 4 was the time step that led to a decrease in the temperature iteration in Figure 5.13c. The high residual explains why the solver needed fewer iterations for this time step. For each time step, the residual has decreased, until it, for time step 4, reached the tolerance with a smaller iteration number. This time step's residual will then be higher, because it has not had as many iterations to drive down the residual. Looking at the pressure residuals, Figures 5.15a and 5.15b, we see that the residuals for the fourth time step is high also here, coinciding with the dip in iterations from Figure 5.13, though they are not the highest residuals. The residual for the first time step in Figure 5.15b is quite high, resulting in the lower iteration number for the first time step in Figure 5.13b. Both pressure plots in Figure 5.15 have residuals that oscillate, which once again is explained by the fact that the residuals decrease until the solver needs fewer iterations to push the residual under the tolerance.



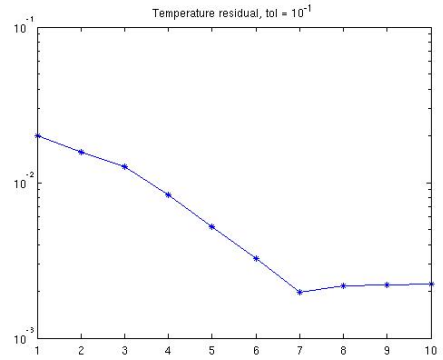
(a) Pressure, $tol_{iter} = 10^{-6}$



(b) Pressure, $tol_{iter} = 10^{-1}$



(c) Temperature, $tol_{iter} = 10^{-6}$



(d) Temperature, $tol_{iter} = 10^{-1}$

Figure 5.15: Average multiscale residuals for the last iteration per time step. A time step of $\Delta t = 20$ days was used, and the simulation ran for 200 days.

Test of different number of grid blocks: Changing the number of coarse grid blocks in the multiscale system will change the accuracy of the solver. Using a higher number of grid blocks will generally lead to a higher accuracy, as the number of blocks in the system now is closer to the number of cells in the sequential system. When the number of coarse grid blocks equals the number of fine cells, that is, when $m = n$, the multiscale method mimics the sequential method to computer precision. We have in Table 5.9 listed the discrepancy between the multiscale and sequential temperature solutions for different degrees of coarsening. We have looked at coarse grid with 8, 66 and 3300 grid blocks. For each coarsening, we looked at two scenarios. First we let the multiscale solver run until the set iterative tolerance was reached, just as we have done with all other simulations. Then, we looked at what happened if we set a maximum iteration number, $iter_{max}$. That is, if the iterative multiscale method has not reached convergence by $iter_{max}$, we use the value from $iter_{max}$. We let $iter_{max} = 5$. For all simulations, we let $\Delta t = 20$ days, the iterative tolerance was 10^{-3}, while the other tolerances were 10^{-6}. The simulation ran for 200 days.

The table shows that if we do not set a restriction on the maximum iterations, the multiscale method is accurate even for very few blocks. This comes from the fact that we iterate until the residual is small enough. Each iteration uses the fine scale system when it applies the smoother, so when we can use many it-

Table 5.9: Temperature discrepancy between the sequential and multiscale solutions for different coarse grid sizes. The iterative tolerance is 10^{-3}, while all the other tolerances are 10^{-6}. When we have 66 blocks, the coarse dimensions are $6 \times 11 \times 1$, when we have 8 blocks, the coarse dimensions are $2 \times 4 \times 1$, and when we have 3300 blocks the coarse dimensions are $30 \times 110 \times 1$. The first three rows, the ones corresponding to $iter_{max} = \infty$ let the iterative multiscale run until reached tolerance, while the last three rows, $iter_{max} = 5$ ran the iterative multiscale five times unless tolerance was reached before said five iterations.

		$iter_{max} = \infty$		$iter_{max} = 5$	
	Time (days)	L^2	L^∞	L^2	L^∞
66 blocks	t = 20	$8.356 \cdot 10^{-8}$	$1.559 \cdot 10^{-6}$	$4.130 \cdot 10^{-6}$	$7.252 \cdot 10^{-5}$
	t = 100	$1.522 \cdot 10^{-7}$	$1.574 \cdot 10^{-6}$	$2.062 \cdot 10^{-6}$	$1.800 \cdot 10^{-5}$
	t = 200	$1.652 \cdot 10^{-7}$	$1.658 \cdot 10^{-6}$	$1.788 \cdot 10^{-6}$	$2.107 \cdot 10^{-5}$
8 blocks	t = 20	$1.491 \cdot 10^{-7}$	$2.119 \cdot 10^{-6}$	$6.193 \cdot 10^{-5}$	$1.167 \cdot 10^{-3}$
	t = 100	$1.132 \cdot 10^{-6}$	$3.131 \cdot 10^{-5}$	$1.454 \cdot 10^{-4}$	$2.773 \cdot 10^{-3}$
	t = 200	$9.231 \cdot 10^{-7}$	$5.904 \cdot 10^{-6}$	$3.385 \cdot 10^{-4}$	$5.130 \cdot 10^{-3}$
3300 blocks	t = 20	$4.662 \cdot 10^{-8}$	$1.175 \cdot 10^{-6}$	$4.662 \cdot 10^{-8}$	$1.175 \cdot 10^{-6}$
	t = 100	$1.032 \cdot 10^{-6}$	$1.494 \cdot 10^{-5}$	$1.032 \cdot 10^{-6}$	$1.494 \cdot 10^{-5}$
	t = 200	$6.175 \cdot 10^{-7}$	$7.721 \cdot 10^{-6}$	$6.175 \cdot 10^{-7}$	$7.721 \cdot 10^{-6}$

erations the solutions will converge to the fine scale solutions. Even though the discrepancy may be small, the iteration number may be high. If we apply maximum iterations on the other hand, we see that having very few blocks results in poorer accuracy, though the results are still remarkably good considering the few blocks. The accuracy improves as the number of coarse blocks increases. Note that when 3300 blocks are used, the method uses less than 5 iterations to converge, the two scenarios have the same discrepancies. When we look at the results from 66 blocks, we see that we still get good results. Compared to the original 13,200 fine cells, 66 blocks are few. So all in all, the multiscale method seems to work really well.

Change of enthalpy and internal energies: We will again look at what happens if we change the enthalpy and internal energies. We will use the same cases as in Section 4.5. We will thus have three cases, where the first case changes the constant in the functions for the enthalpy and internal energy for the rocks, the second case changes the internal energy, and the third combines case 1 and 2, and thus

CASE 1: $c_r = 0.5 \cdot 10^3$,

CASE 2: $U_\alpha = C_U T$,

CASE 3: $c_r = 0.5 \cdot 10^3, U_\alpha = C_U T$.

The discrepancy between the sequential and multiscale temperature solution for the three cases is given in Table 5.10. The iterative tolerance is 10^{-1}, while the other tolerances are 10^{-6}. The time step used is $\Delta t = 20$ days, and we have again used a coarsening factor of 66. The table shows that the discrepancy between the two methods is small for all three cases. Considering the relatively high iterative tolerance, these values are very good. Though the errors for the three cases are similar, we see that the multiscale method works best on CASE 3. Remember from Section 4.5 that CASE 3 is the case that has the strongest diffusion term. The heat equation's character is determined by the advection and diffusion term. The more dependent on the diffusion term, the more elliptic. This means that the heat equation is more elliptic in character for CASE 3 than for the other cases. The multiscale method usually works best on elliptic equations, which we can see here by looking at Table 5.10, where the discrepancy is smallest for the third case. If we compare the values from Table 5.10 to the values in Table 5.6, giving the discrepancy for the first example where everything except the enthalpies and internal energies are the same as now, we see that all the discrepancies in Table 5.10 are smaller than the ones found in Table 5.6. Though many of the discrepancies are close. This again shows that the multiscale method works better on elliptic equations, as all three cases lead to a heat equation that is more elliptic than the heat equation used to find the values in Table 5.6.

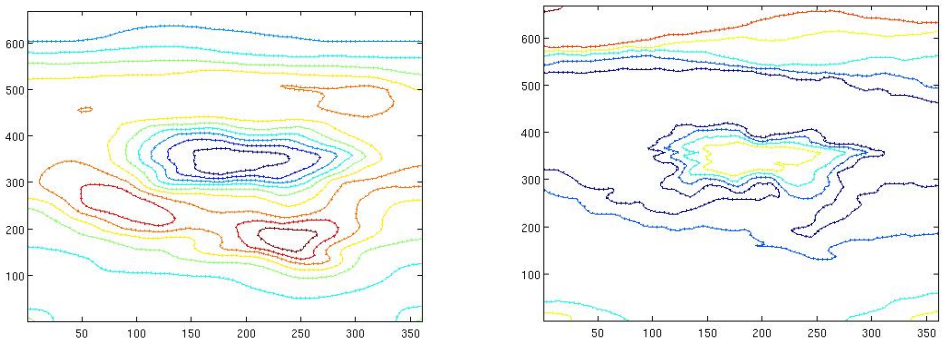
Table 5.10: Temperature discrepancy between the sequential and multiscale solution for $t = 20, 100, 200$ days, when the enthalpy and internal energy are changed. We have used $\Delta t = 20$ days, the iterative tolerance is 10^{-1}, while the other tolerances are 10^{-6}.

	Case 1		Case 2		Case 3	
Time (days)	L^2	L^∞	L^2	L^∞	L^2	L^∞
t = 20	$5.397 \cdot 10^{-5}$	$7.505 \cdot 10^{-4}$	$4.746 \cdot 10^{-5}$	$1.280 \cdot 10^{-3}$	$3.218 \cdot 10^{-6}$	$2.873 \cdot 10^{-5}$
t = 100	$2.213 \cdot 10^{-5}$	$1.071 \cdot 10^{-4}$	$2.459 \cdot 10^{-5}$	$1.942 \cdot 10^{-4}$	$4.898 \cdot 10^{-7}$	$1.999 \cdot 10^{-6}$
t = 200	$2.180 \cdot 10^{-5}$	$8.515 \cdot 10^{-5}$	$2.104 \cdot 10^{-5}$	$1.524 \cdot 10^{-4}$	$6.405 \cdot 10^{-7}$	$3.990 \cdot 10^{-6}$

To change the character of the heat equation, we could also change the thermal conductivity coefficient κ. We have as previously stated, considered a simplified case and used a homogeneous value for the conductivity instead of a heterogeneous one. However, the use of a heterogeneous conductivity would make the problem more realistic, and it would also be interesting to study its impact on the multiscale solution. This will therefore be studied in a later example, and should be explored more in future works.

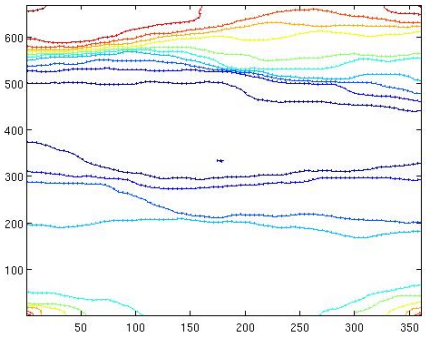
Figure 5.16 gives the contour plots of the sequential and multiscale temperature solution for $t = 200$ days for the three cases. The same parameters as above are used. We can again compare to the contour plots generated with the original enthalpy and internal energy values, Figure 5.9. The iterative tolerance is 10^{-1} for both figures. For Figure 5.9 this relatively high tolerance leads to noise in the multiscale solution in the form of contour lines that should not be there. Comparing with the contour lines in Figure 5.16, we see that this is not the case for our three cases. This time, we get very good results even with the high tolerance, the excess contour lines have completely disappeared. The multiscale solution mimics the sequential solution to a large degree and the two solutions are identical to the naked eye for all three cases. This is different than Figure 5.9, where the sequential solution was visible in some, but few, areas. All in all, we can conclude that the multiscale method works even better for these new cases.

If we study the average number of iterations per time step in the simulation for the three cases, a lot is similar to the number of iterations for the original case. The multiscale method uses fewer iterations to find the temperature solution than the pressure solution, but the number of iterations increases for both solutions when the iterative tolerance decreases. Figure 5.17 shows the average number of iterations used to find the temperature when the iterative tolerance is 10^{-1}. Figures 5.17b and 5.17c display a lot of what we have seen before. Case 2, i.e., Figure 5.17b, uses an average of one iteration per time step, just as the original case, while Case 3, Figure 5.17c, uses less. The multiscale method goes through the multiscale iteration one time before it applies the Newton-Raphson method, to see if we already have the answer, and for case 3, the multiscale method uses zero iterations to reach the tolerance in this first try. It uses an



(a) Changed enthalpy

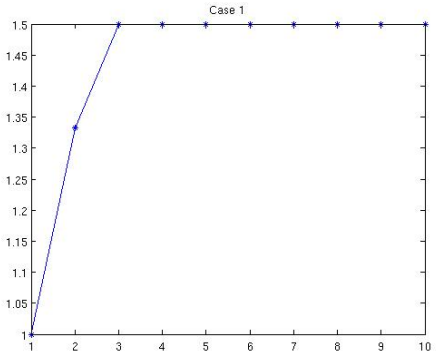
(b) Changed internal energy



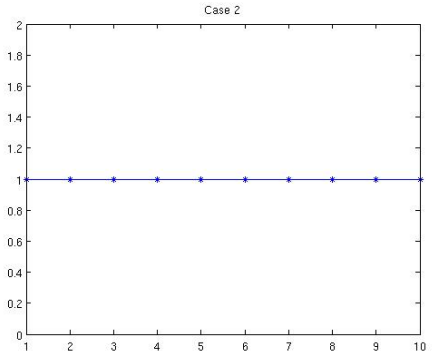
(c) Changed internal energy and enthalpy

Figure 5.16: Contour plots for the sequential and multiscale temperature solution when $t = 200$ days for CASE 1, 2 and 3. The sequential contour lines are given in continuous lines, while the multiscale contour lines are dotted. A time step of $\Delta t = 20$ days is used, and $tol_{iter} = 10^{-1}$.

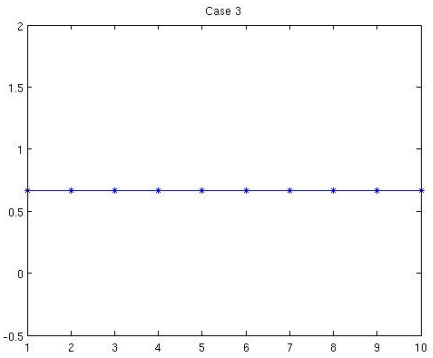
average of one iteration in the Newton-Raphson method, so it still follows the normal pattern from above. When we look at CASE 1 and Figure 5.17a, we see that though the case starts out as usual, with one iteration per simulation step, it quickly rises to almost 2. Looking at the discrepancy for the case, Table 5.10, we see that this rise fits with the discrepancy. The discrepancy is biggest when $t = 20$, i.e. the first simulation step, before it decreases as time goes on. This is reasonable considering the fact that more iterations are used for the other simulation steps. If we were to use a smaller iterative tolerance, the average number of iterations would increase for all three cases. The figures have been omitted for brevity, though it should be said that for CASES 2 and 3, the increase is small and we need fewer iterations than the original case. This fits with the fact that the multiscale method is better at finding the solutions for elliptic equations.



(a) Changed enthalpy



(b) Changed internal energy



(c) Changed internal energy and enthalpy

Figure 5.17: The average number of temperature iterations in the iterative multiscale method as a function of time step for the three different cases when the iterative tolerance is 10^{-1}. The simulation ran for 10 time steps for all the cases, and $\Delta t = 20$ days was used.

The average number of iterations used to find the pressure solution when the iterative tolerance is 10^{-1} is identical for all three cases, see Figure 5.18. the multiscale method uses overall fewer iterations now compared to the number of iterations used to find the pressure solution in the original case, Figure 5.13b. The shape is similar to before, however, with a lower number of iterations for the first time step and a dip for time step four. This is not unexpected, as we have only changed the parameters in the heat equation. This will affect the pressure equation as well, as it is dependent on the temperature, but the change will be of a lower degree, as we can see from Figure 5.18. If we use a smaller iterative tolerance, the iterations will increase.

Layer 80: As a quick test, we are also going to look at a layer from the Upper Ness formation, which is considered to be a difficult model for multiscale methods. We are going to consider the 80th layer, which has the permeability field given in Figure 5.19. We are going to keep the well placements and restrictions from Layer 5. For the enthalpies and internal energies we are going to look at

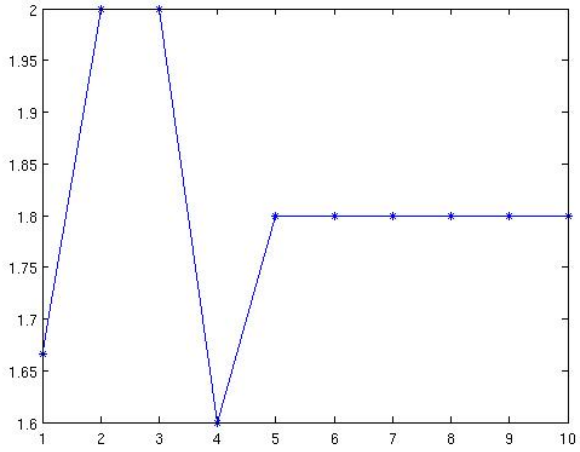


Figure 5.18: The average number of pressure iterations in the iterative multiscale method as a function of the time step for CASE 1, CASE 2 and CASE 3, when the iterative tolerance is 10^{-1}. The simulation ran for 10 time steps for all the cases, and $\Delta t = 20$ days was used. The plot was equal for all three cases.

the original values from above, and compare this to the values given by CASE 3 from above. All other values, parameters and functions remain unchanged.

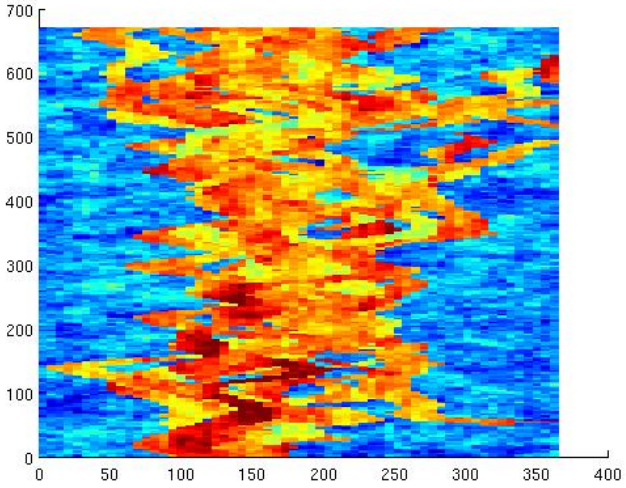


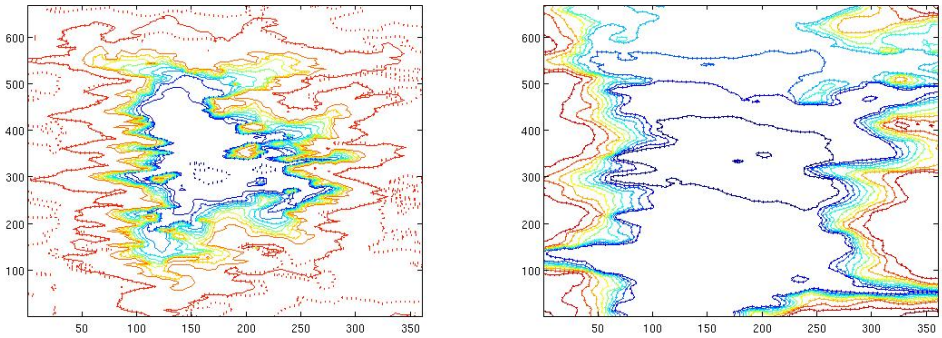
Figure 5.19: The permeability given in logarithmic scale of the 80th layer of the SPE10 data set.

The pressure and temperature discrepancies between the multiscale and sequential method are given in Table 5.11. The table shows the values for both the original setting of the enthalpies and internal energies, and the discrepancies obtained using CASE 3, resulting in the more elliptic version of the heat equation. As can be seen from the table, the multiscale method works a lot better on the more elliptic case. This is also evident when we look at contour lines

of the two methods for each case, given in Figure 5.20. The multiscale contour lines are dashed, while the sequential ones are given as continuous lines. While the contour lines match closely for the more elliptic case, Figure 5.20b, there are unmatched lines in the original case, Figure 5.20a. The iterative tolerance is 10^{-1} though, so the results are relatively good for the original case as well. All in all, we see that the multiscale method works well on this layer.

Table 5.11: Pressure and temperature discrepancies for SPE10, layer 80. The discrepancies are shown for two different sets of enthalpies and internal energies. The one denoted Original has values given in Section 4.5, while the one denoted CASE 3 has values given above. The temperature discrepancy is given by e_T, while the pressure discrepancy is given by e_p.

		Original		CASE 3	
	Time (days)	L^2	L^∞	L^2	L^∞
e_T	t = 20	$5.642 \cdot 10^{-5}$	$3.571 \cdot 10^{-3}$	$2.485 \cdot 10^{-6}$	$1.779 \cdot 10^{-5}$
	t = 100	$4.835 \cdot 10^{-5}$	$1.322 \cdot 10^{-3}$	$1.452 \cdot 10^{-6}$	$1.919 \cdot 10^{-5}$
	t = 200	$1.689 \cdot 10^{-4}$	$1.562 \cdot 10^{-2}$	$3.203 \cdot 10^{-6}$	$5.486 \cdot 10^{-5}$
e_p	t = 20	$7.517 \cdot 10^{-6}$	$2.737 \cdot 10^{-5}$	$7.517 \cdot 10^{-6}$	$2.737 \cdot 10^{-5}$
	t = 100	$2.784 \cdot 10^{-6}$	$2.395 \cdot 10^{-5}$	$3.436 \cdot 10^{-6}$	$3.941 \cdot 10^{-5}$
	t = 200	$6.220 \cdot 10^{-6}$	$5.824 \cdot 10^{-5}$	$6.590 \cdot 10^{-6}$	$1.010 \cdot 10^{-4}$



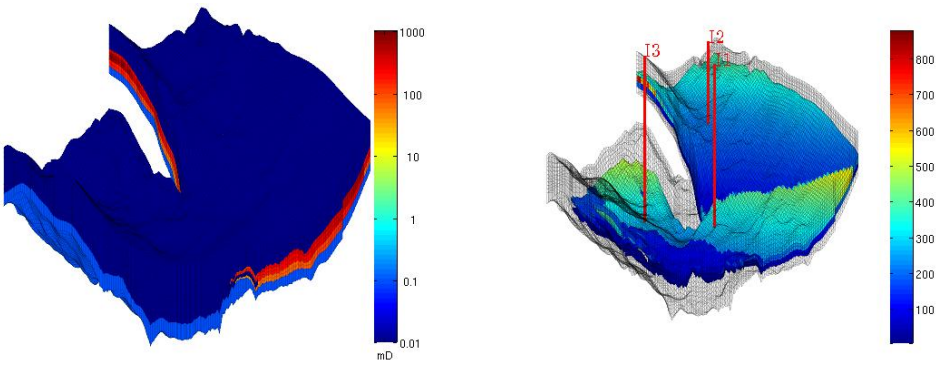
(a) Original (b) CASE 3

Figure 5.20: Contour lines for the multiscale (dotted lines) and sequential (continuous lines) temperature solution for SPE10, layer 80 with the original setting and the more elliptic heat equation given with the values presented in CASE 3. The solutions are shown for $t = 200$ days, a time step of $\Delta t = 20$ days is used, and $tol_{iter} = 10^{-1}$.

5.4.4 The Johansen Formation

We are now going to look at the Johansen formation, which is a real life formation situated in the North Sea. The data pertaining the formation can be downloaded from SINTEF's website, `https://www.sintef.no/projectweb/matmora/downloads/johansen/`. It has a big and complex formation, containing faults, non-smooth surfaces and different types of sand. The formation has been suggested as a location for CO_2 storage. This is not going to be the focus here, however. We are interested in seeing how the multiscale solver handles complicated grids in three dimensions, with realistic permeability and porosity values, rather than seeing how well the reservoir is suited for CO_2 storage. We are therefore going to inject the reservoir with water, because that has been the norm in this thesis. We are not going to include any production wells, however, partly because the field is thought of as a storage area, and partly because the formation has very low permeability values, which enforces changes to happen slowly and this will make it difficult for the phases to reach the wells.

The Johansen formation is represented through a $100 \times 100 \times 11$ grid, and consists of several areas, made up of different types of rocks and thus different permeabilities. We are most interested in the layers with a little higher permeability. We are therefore going to strip away the cells with very low permeabilities when showing the solutions, to be able to properly study the change in pressure and temperature. The permeability field of the whole model, as well as the permeability field for the layers with higher permeability is given in Figure 5.21. The figure also show the placement of the wells. We have placed three injection wells in the system, each controlled by the surface rate. The wells penetrate the 10th layer of the model, which is part of the area with the higher permeability.



(a) Permeability - Whole reservoir (b) Permeability - Higher permeability

Figure 5.21: The permeability of the Johansen formation, as well as the permeability in the cells that have not been stripped away because of very low permeability.

We have used a coarse grid with $20 \times 20 \times 3$ grid blocks, and let the simulation run until $t = 5$ years and 175 days has been reached. The end time is arbitrarily chosen, but we needed a longer time in order to see some proper change in the reservoir. This will be our end time for the rest of the examples. The pressure and temperature discrepancies for different times t are shown in Table 5.12. The Newton tolerance is 10^{-6}, while the iterative tolerance is 10^{-1}. As can be seen, the multiscale method is able to recreate the sequential reference solution to a large degree. This is especially impressive considering the complex grid.

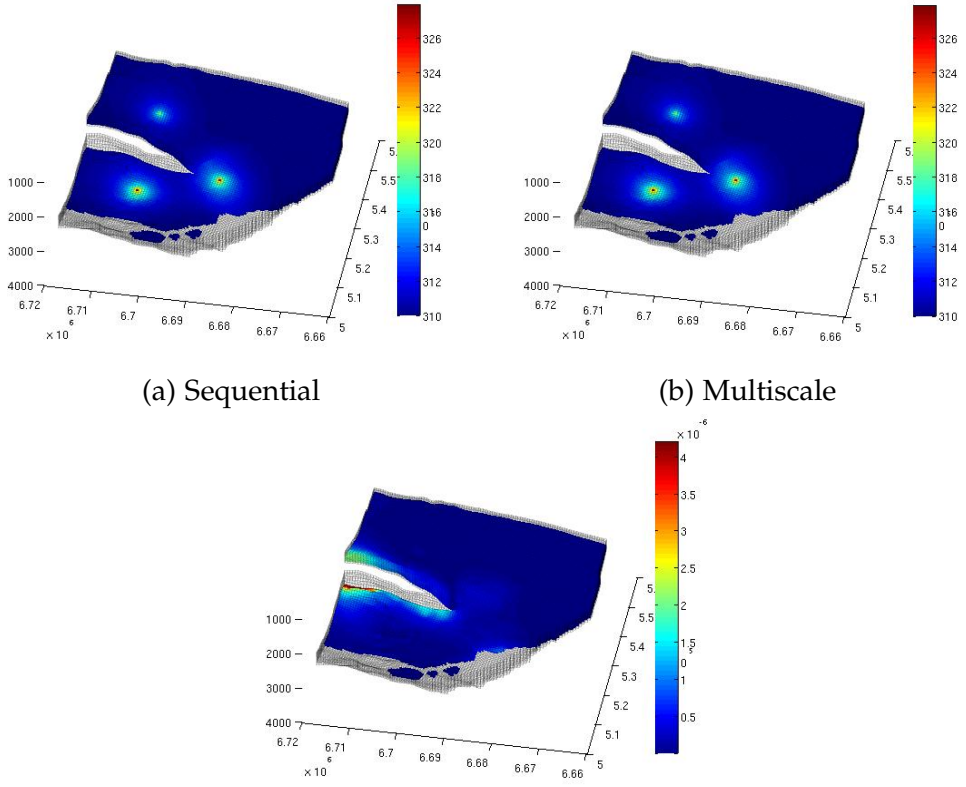
Table 5.12: The temperature e_T and pressure e_T discrepancy for the Johansen formation when the simulation ran for $t = 5$ years and 175 days. A time step of $\Delta t = 20$ days is used, and $tol_{iter} = 10^{-1}$.

	Time (days)	L^2	L^∞
e_T	t = 20	$1.594 \cdot 10^{-9}$	$8.442 \cdot 10^{-8}$
	t = 1000	$3.403 \cdot 10^{-7}$	$8.387 \cdot 10^{-6}$
	t = 2000	$5.951 \cdot 10^{-7}$	$1.232 \cdot 10^{-6}$
e_p	t = 20	$5.112 \cdot 10^{-9}$	$2.661 \cdot 10^{-7}$
	t = 1000	$1.100 \cdot 10^{-6}$	$2.756 \cdot 10^{-5}$
	t = 2000	$1.932 \cdot 10^{-6}$	$4.161 \cdot 10^{-6}$

It might be a little surprising that the discrepancy is so small as it is. This is, after all, an example that should be difficult to simulate. Figure 5.22 might give an explanation. The figure shows the sequential and multiscale temperature solution as well as the difference between them when $t = 5$ years and 175 days. Because of the large dimensions of the reservoir, as well as the low permeability, the temperature has not spread through the reservoir with these high times. Similar results are found for pressure and saturation. And, as there have been relatively little change in the system, the multiscale method does not have too much problem mastering the simulation. Note that most of the error occur by the well closest to the fault.

5.4.5 SAIGUP

As another realistic example of a petroleum reservoir, we are going to look at a model from the SAIGUP project [16]. The project generated a broad spectre of shallow-marine reservoir models, and one of these can be accessed through MRST. The model in question constitutes a complex three dimensional geometry which contains faults and a realistic permeability field. The model is made up of $40 \times 120 \times 20$ fine cells, which we are going to partition into $10 \times 20 \times 5$ coarse grid blocks. We are again going let the reservoir be filled with mostly oil initially, which is then displaced by water throughout the simulation. Again we will let the simulation run until $t = 5$ years and 175 days is reached. This is done in order to see proper change in the data, and also because oil recovery



(a) Sequential

(b) Multiscale

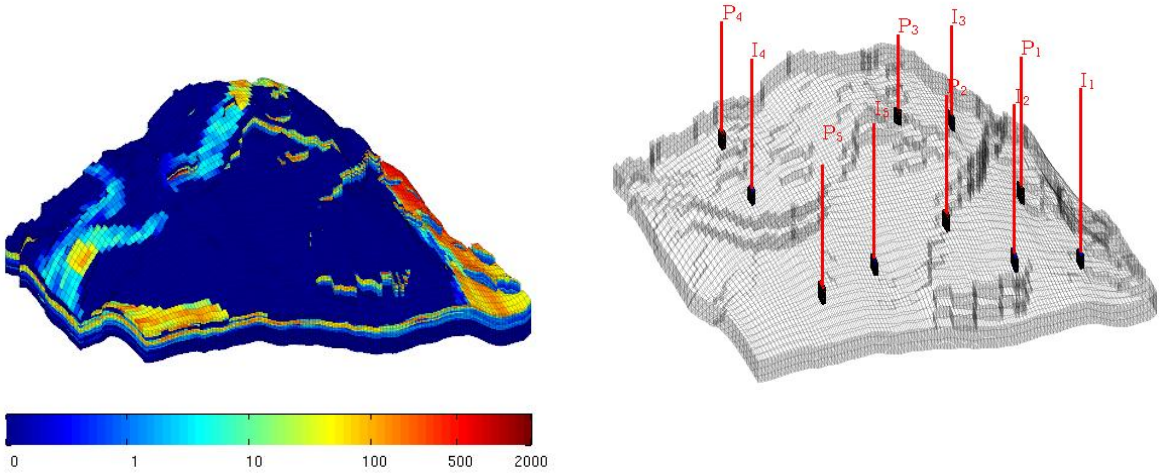
(c) Error

Figure 5.22: Sequential and multiscale temperature solution, as well as the difference between them for the Johansen formation when $t = 5$ years and 175 days. A time step of $\Delta t = 20$ days is used, and $tol_{iter} = 10^{-1}$.

goes over a long time. A time step of $\Delta t = 20$ days will be used throughout.

The horizontal permeability throughout the reservoir given in logarithmic scale is shown in Figure 5.23a. There is, as can be seen, a big variance between the values. We have placed ten wells in the system, five injection wells controlled by the surface rate, and five producer wells controlled by the bottom hole pressure. The wells can be seen in Figure 5.23b. The injection wells are placed in the bottom 12 layers, while the production wells are placed in the topmost 14 layers.

The temperature and pressure discrepancies between the multiscale and sequential method when $t = 20, 1000, 2000$ days are given in Table 5.13. The Newton-Raphson tolerance is 10^{-6} for both methods, and the iterative tolerance is 10^{-1}. The table shows two sets of discrepancies. One generated using the original enthalpy and internal energy values used in the examples above, and the other using the values from the third case in the SPE10-Layer 5 example. As can be seen from the table, the multiscale method gives a good approximation for both pressure and temperature for both cases. The results are especially good con-



(a) Permeability (b) Geometry with wells

Figure 5.23: The horizontal permeability of the SAIGUP model given in logarithmic scale, as well as the geometry of the model with the placement of the wells.

sidering the high iterative tolerance. This model is complex, and the fact that the multiscale method works so well with the given tolerance is satisfactory. Though both cases give good results, we see that the results are best for the most elliptic case, CASE 3, at least for the temperature solution. This once again indicates that the multiscale method works best on elliptic equations. The fact that the pressure discrepancy does not change when the enthalpy and internal energy change is not that surprising, as the pressure equation does not depend on said functions.

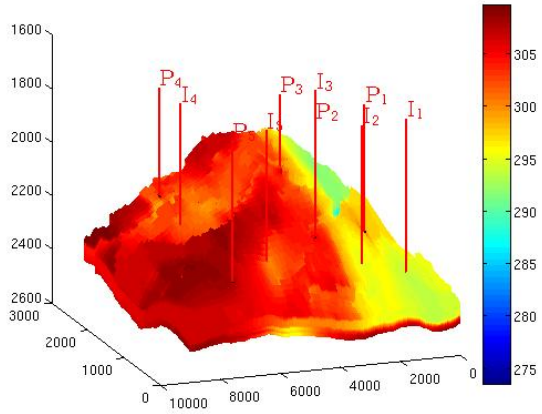
Table 5.13: The temperature and pressure discrepancies for two different values of enthalpy and internal energy. The original case has functions that are used in the examples above, while CASE 3 was defined in the example concerning SPE10-Layer 5. The temperature discrepancy is given by e_T, while the pressure discrepancy is given by e_p. We have $tol_{iter} = 10^{-1}$.

		Original		CASE 3	
	Time (days)	L^2	L^∞	L^2	L^∞
e_T	t = 20	$6.234 \cdot 10^{-8}$	$2.331 \cdot 10^{-6}$	$1.667 \cdot 10^{-8}$	$3.067 \cdot 10^{-7}$
	t = 1000	$8.832 \cdot 10^{-6}$	$5.106 \cdot 10^{-4}$	$7.471 \cdot 10^{-7}$	$2.780 \cdot 10^{-5}$
	t = 2000	$8.180 \cdot 10^{-6}$	$4.456 \cdot 10^{-4}$	$4.298 \cdot 10^{-7}$	$1.521 \cdot 10^{-5}$
e_p	t = 20	$5.391 \cdot 10^{-8}$	$6.686 \cdot 10^{-7}$	$5.391 \cdot 10^{-8}$	$6.686 \cdot 10^{-7}$
	t = 1000	$2.312 \cdot 10^{-6}$	$6.875 \cdot 10^{-5}$	$2.398 \cdot 10^{-6}$	$7.076 \cdot 10^{-5}$
	t = 2000	$1.481 \cdot 10^{-6}$	$4.662 \cdot 10^{-5}$	$1.470 \cdot 10^{-6}$	$4.590 \cdot 10^{-5}$

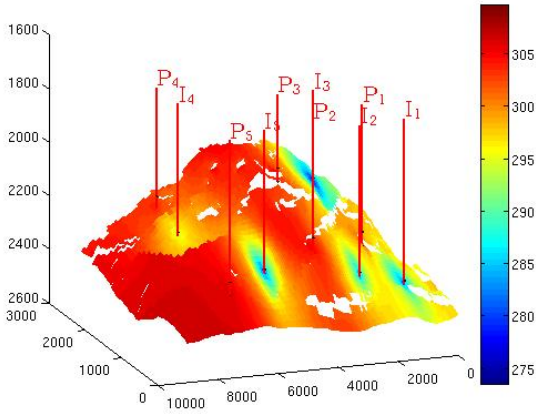
Figure 5.24 gives the sequential temperature solution for $t = 2000$ days (that is, $t = 5$ years and 175 days), for the whole reservoir, as well as the same solution for the twelfth layer in the model. The more elliptic version (CASE 3) has been used. The twelfth layer was chosen because this is a layer that is in between the completion of the injection and projection wells, so it should notice the effects from both types of wells and it is therefore interesting to see how the solutions are affected. The multiscale temperature solution for the same areas and time are also given, as well as the discrepancy between the two solutions. The discrepancy is, as can be seen, small throughout. The biggest error occurs around the biggest cluster of wells. This is consistent with findings from the other examples, where we found the difference between the two methods to be biggest in the areas containing a production well. This is where there will be the biggest change in the solution, and discrepancies will therefore occur. The fact that the discrepancy is smaller for the twelfth layer than for the whole model is also not that surprising, as we enforce the well values and the wells penetrate this layer. Note the fractures in the twelfth layer in Figure 5.24. The multiscale method is still able to give good results despite the faults.

If we look at the run time of the two solvers, we see that the multiscale method outperforms the sequential method. While the sequential method used 3144.5 seconds, or 52.4 minutes, to generate the figures used in Figure 5.24, the multiscale method used only 853.3 seconds, or 14.2 minutes. The multiscale method is thus nearly 4 times as fast as the sequential method for this complex example. This illustrates one of the strengths of the multiscale method, because it uses a relatively short run time and still gives a good approximation of the fine scale solution.

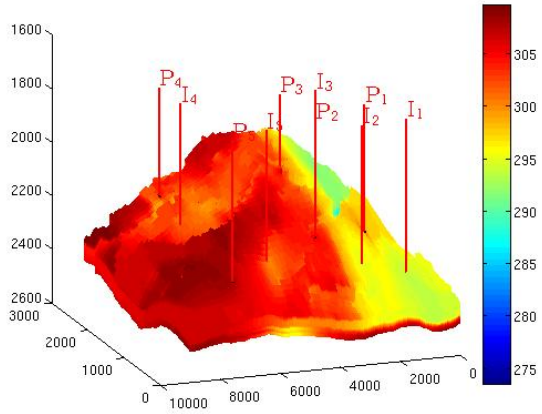
Heterogeneous κ: We have, up to now, used a homogeneous value for the thermal conductivity coefficient κ, where we have let $\kappa = 4$ W/(mK), the conductivity of granite. This is not realistic however. It would therefore be interesting to see what happens if we use a heterogeneous κ. Here, we are going to use the one given in Figure 5.25. The coefficient is found with the help of the porosity. The reasoning is that the porosity gives the volume of the reservoir that can contain fluid, and so where the porosity is zero or close to zero, there is most likely a solid rock, and where the porosity is bigger, there might be a material that is able to flow, like water or oil. The thermal conductivity of solid rocks ranges from 2-7 W/(mK), depending on what kind of rock one are considering, while the thermal conductivity of oil lies around 0.15 W/(mK) and the thermal conductivity of water is 0.58 [27]. The porosity for the SAIGUP model ranges from 0.0093-0.2911. We have therefore let the thermal conductivity equal 7 W/(mK) in the cells where the porosity is less than 0.07, because we figure this is pretty much only solid rock, we have let the conductivity equal 2 W/(mK) in the cells where the porosity is between 0.07 and 0.15, and we have let the thermal conductivity equal 0.15 everywhere else. This gives the thermal conductivity given in Figure 5.25. The figure also gives the porosity of the model. To get a more



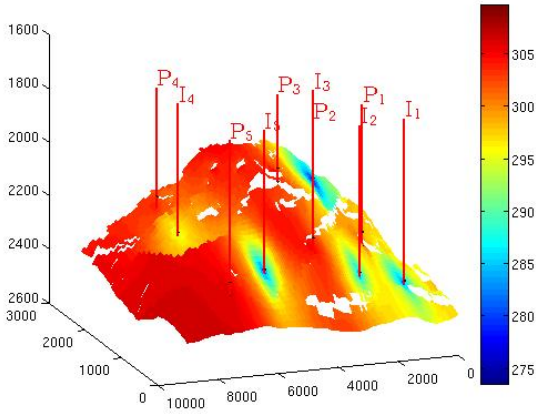
(a) Sequential, whole model



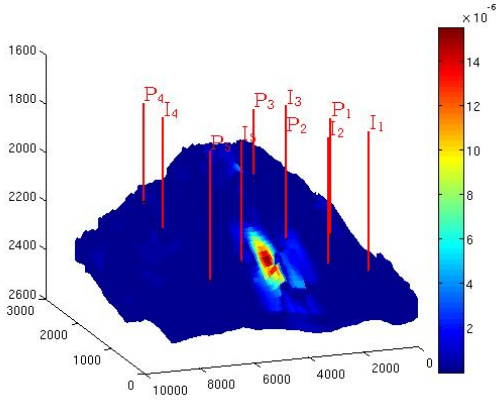
(b) Sequential, Layer 12



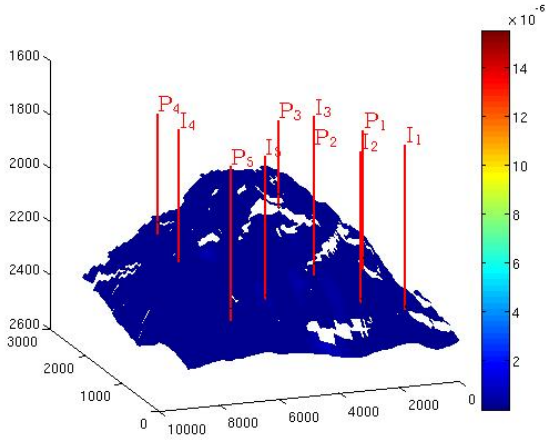
(c) Multiscale, whole model



(d) Multiscale, Layer 12

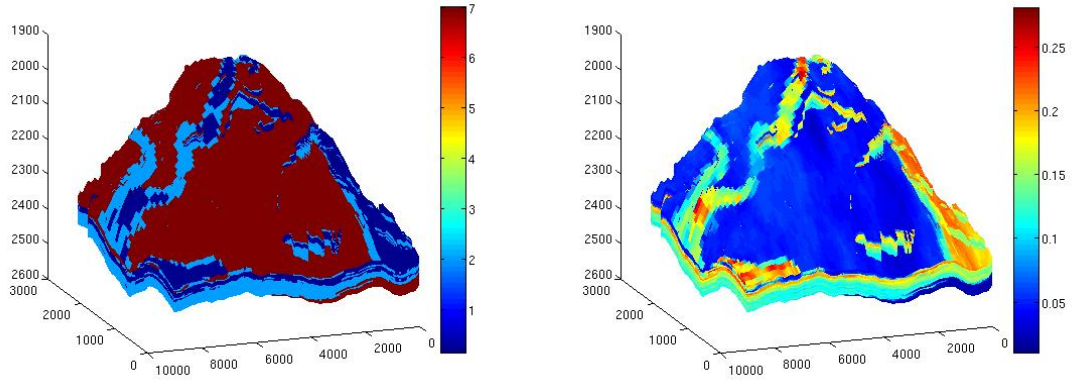


(e) Error, whole model



(f) Error, Layer 12

Figure 5.24: Sequential and multiscale temperature solution, as well as the difference between them, when $t = 2000$ days. The solutions are shown for the whole reservoir and for a single layer, namely the twelfth. A time step of $\Delta t = 20$ days is used, and $tol_{iter} = 10^{-1}$.



(a) Heat conductivity κ

(b) Porosity

Figure 5.25: The heterogeneous heat conductivity κ used in the second part of the SAIGUP example, as well as the porosity field used to generate the conductivity.

realistic model, one should probably update κ throughout the simulation, as the oil will move, and be replaced by water, but this is something we have ignored.

It turns out that the given κ does not affect the multiscale solution to a large degree. Table 5.14 gives the temperature discrepancy between the multiscale method and the sequential method when the homogeneous and the heterogeneous κ is used. We have used the more elliptic variant for the enthalpies and internal energies (CASE 3) so the two first columns in Table 5.14 is equal to the two last columns in Table 5.13. All other parameters are as used in Table 5.13. As can be seen from Table 5.14, the accuracy of the multiscale method does not really change with the new κ. We have a very slight decrease in discrepancy, but there is no significant change.

Table 5.14: Multiscale and sequential temperature discrepancy when a homogeneous and heterogeneous value is used for the thermal conductivity coefficient κ.

	Homogeneous κ		Heterogeneous κ	
Time (days)	L^2	L^∞	L^2	L^∞
t = 20	$1.667 \cdot 10^{-8}$	$3.067 \cdot 10^{-7}$	$1.666 \cdot 10^{-8}$	$3.058 \cdot 10^{-7}$
t = 1000	$7.471 \cdot 10^{-7}$	$2.780 \cdot 10^{-5}$	$7.456 \cdot 10^{-7}$	$2.651 \cdot 10^{-5}$
t = 2000	$4.298 \cdot 10^{-7}$	$1.521 \cdot 10^{-5}$	$4.459 \cdot 10^{-7}$	$1.458 \cdot 10^{-5}$

As can be seen from the depiction of κ, Figure 5.25, there is still quite a big part of the variable that is non-changing, even though the value is now heterogeneous. There is for instance a big part of the reservoir that has a thermal conductivity of $\kappa = 7$ W/(mK). This might be the reason why the discrepancy change is so small. To truly get realistic examples, one should use actual κ values from real reservoirs, or values that have been specifically designed to fit

petroleum reservoirs. This is not something that we have access to, so we were content with the given values, which let us study the impact on the multiscale method. Having more realistic κ's can impact the method in a way unbeknowns to us, however, and it is therefore something that should be explored in future works.

5.4.6 Norne Field

As a last example, we are going to look at the Norne Field. The reservoir is situated in the Norwegian Sea, and has been extracting oil through water injection since 1997. Statoil and its partners have in collaboration with the Norwegian university of science and technology (NTNU) released data from the field for educational purposes [10]. This enables us to study the impact of the multiscale method on a petroleum reservoir with a real life grid representation and petrophysical data. We are, in accordance with real life, going to let the oil be displaced through water.

The reservoir is modeled through a grid with $46 \times 112 \times 22$ fine cells, which we partition into a coarse grid of $15 \times 20 \times 5$ coarse grid blocks. As the Norne field is an active reservoir, it of course has wells placed in strategic places. We are going to disregard these wells, however, and arbitrarily place three production and three injection wells in the reservoir. We have let the injection wells be controlled by the surface rate, while the production wells are controlled by the bottom hole pressure. All wells are vertical, the injection wells are completed in layer 10-20, while the production wells are completed in the first ten layers. The permeability field in logarithmic scale is given in Figure 5.26. The figure also gives the placement of the wells. Note the irregularities of the grid, containing

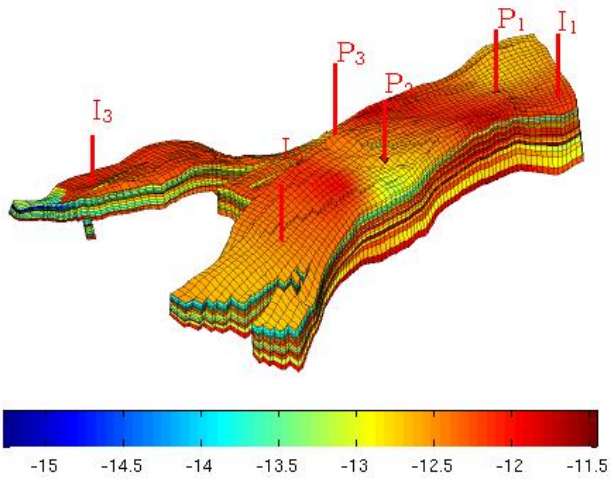


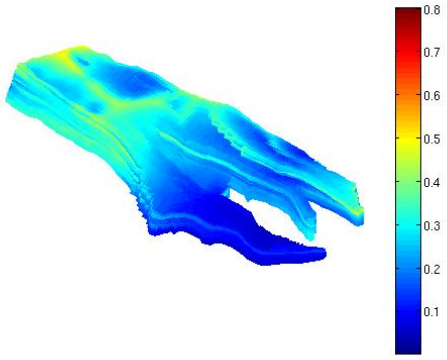
Figure 5.26: The permeability given in logarithmic scale of the Norne field.

faults and cracks. The reservoir also has a full layer of inactive cells, where the wells supply communication between the separated layers.

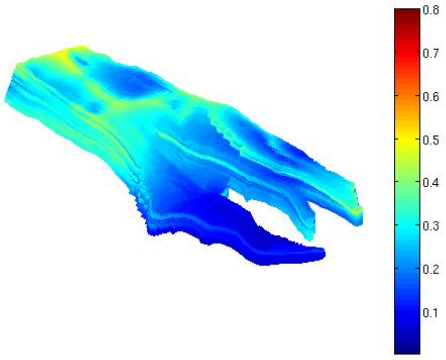
We simulate the flow process for 5 years and 175 days, to get a more realistic feel of the reservoir. As a time step, we use $\Delta t = 20$ days, and the Newton-Raphson tolerance is 10^{-6}. This time around, the multiscale method breaks down when we use an iterative tolerance of 10^{-1}. The method produces a pivot that is zero, which leads to break-down in the ILU(0) solver. This can be fixed either by updating the basis functions more frequently, or decreasing the iterative tolerance. As the updates of basis functions leads to a less efficient code, we are going to keep them fixed and instead use an iterative tolerance of 10^{-2}. This leads to good results, the temperature discrepancy in the L^2 norm lies around 10^{-5}, while the pressure discrepancy in the L^2 norms lies around 10^{-6}. The oil saturation for $t = 5$ years, 175 days found by both the sequential and multiscale method is given in Figure 5.27. The figure also show the difference between the two methods. Again we can see that the multiscale method produces good results. Note also that the oil saturation is greatly reduced, the reservoir initially had an oil saturation of $s_o = 0.8$ throughout the reservoir.

To properly see that the multiscale method is able to give a good approximation of the oil saturation we refer to Figure 5.28, which gives the oil saturation discrepancy as a function of time. The figure first gives the discrepancy between the multiscale method and the sequential method, which again shows that the multiscale method gives a good approximation to the sequential oil saturation. But, as it is really the fully implicit oil saturation we want to mimic, we have also given the discrepancy between the fully implicit method and the multiscale method as well as the discrepancy between the fully implicit method and the sequential method, Figure 5.28b. To the naked eye, the sequential method and the multiscale method give the same fully implicit discrepancy. Note that the discrepancy here is not as good as the one between the multiscale and sequential method. But, as the discrepancy is so small between the multiscale method and the sequential method, we can conclude that all error between the multiscale method and the fully implicit method is the result of differences in the sequential and fully implicit method. This difference can be reduced by applying outer iterations, and we can therefore conclude that the multiscale method is able to mimic the fully implicit solution accurately.

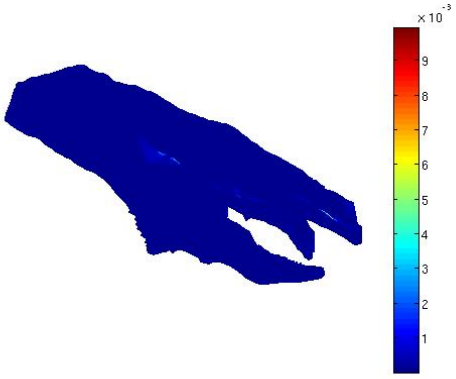
Having verified that the multiscale method accurately solves the complex problem given by the Norne field, it would be interesting to study how the method reacts when we adjust the advection and diffusion terms. This has also been studied in the SPE10 example, as well as a small degree in the Joahnsen and



(a) Sequential



(b) Multiscale



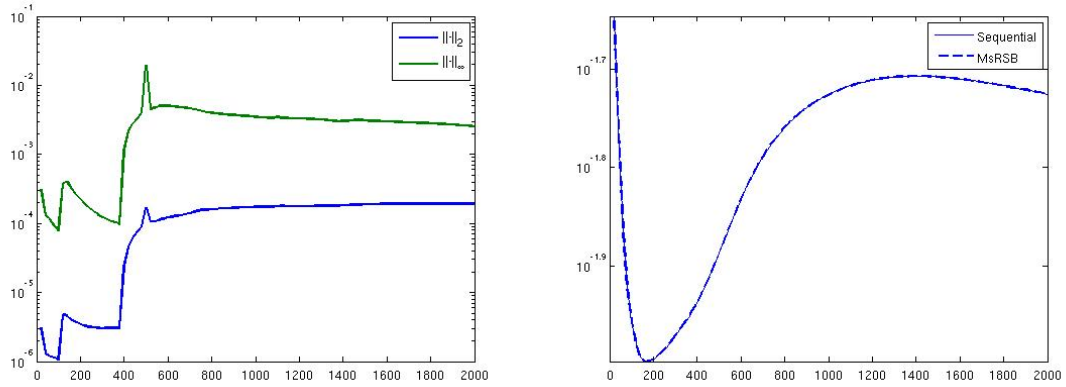
(c) Error

Figure 5.27: Sequential and multiscale oil saturation, as well as the difference between the two methods, for the Norne field when $t = 5$ years, 175 days. A time step of $\Delta t = 20$ days is used, and $tol_{iter} = 10^{-1}$.

SAIGUP examples, but we will try to study it more systematically here. If we go back to Chapter 2, we know that our heat equation is of the form

$$\frac{\partial}{\partial t}(T) + \underbrace{\nabla \cdot (A)}_{a} + \underbrace{\nabla \cdot (\overline{D}\nabla D)}_{d} = C,$$

where we have called the $\nabla \cdot (A)$ term a for advection, and $\nabla \cdot (\overline{D}\nabla D) = d$ for diffusion. The question is what happens with the multiscale solver if we adjust the advection term compared to the diffusion term. That is, will the method be affected if the advection term is bigger or smaller than the diffusion term. It turns out that, at least for this example, a smaller advection term compared to the diffusion term does not really impact the accuracy of the method. Having the advection term be bigger than the diffusion term does have an impact, however. See Table 5.15 for the different discrepancies. The table shows the temperature discrepancy between the multiscale and the sequential method when the two terms are of the same order, when the order of the advection term



(a) Sequential-MsRSB

(b) Fully implicit - Sequential, MsRSB

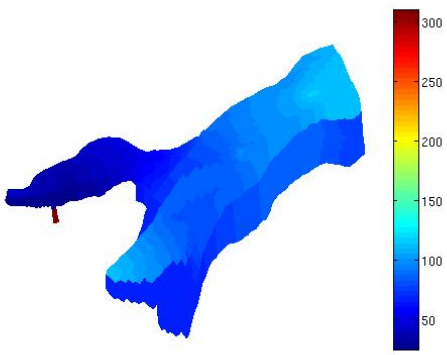
Figure 5.28: Discrepancy in oil saturation between the sequential and multiscale method given in both L^2 and L^∞ norms, as well as the oil saturation discrepancy between the fully implicit method and the sequential method and the fully implicit method and the multiscale method.

is 1000 times that of the diffusion term, and when the order of the advection term is 1000 times smaller than the diffusion term. That is, when $a = \mathcal{O}(d)$, $a = \mathcal{O}(10^3 d)$ and when $a = \mathcal{O}(10^{-3} d)$. As can be seen from the table, having $a = \mathcal{O}(10^3 d)$ leads to an increase in the discrepancy. The multiscale method is thus not as accurate when the advection term is bigger than the diffusion term. This is in line with our findings from the SPE10, Layer 5 example, where we found that the multiscale method works best when the equation is more elliptic and hence has a diffusion term that is more pronounced. Having $a = \mathcal{O}(10^{-3} d)$ does not really lead to any discrepancy change, however. This is a bit surprising, but might just be a result from our problem. By studying the temperature solution for time $t = 5$ years, 175 days for the three different cases of advection, Figure 5.29, we see that while the temperature solution changes when we use $a = \mathcal{O}(10^3 d)$ compared to $a = \mathcal{O}(d)$, the solution of $a = \mathcal{O}(10^{-3} d)$ is very similar to that of $a = \mathcal{O}(d)$. It is therefore not too strange that the error is similar for $a = \mathcal{O}(d)$ and $a = \mathcal{O}(10^{-3} d)$, because the two cases produce nearly the same solution. This is further proven by setting $a = 0$. This does not give a noteworthy difference either, and it is clear that the advection term does not give too much of a contribution when $a = \mathcal{O}(d)$ or less, and will only give a contribution by raising the term, which leads to poorer results.

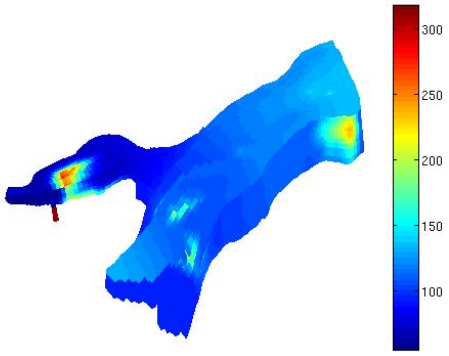
As a last discussion, we study the run time. Let us go back to the parameters used to generate Figure 5.27. The Norne example has also been used in [18]. The paper used the same multiscale method on the two-phase black-oil equations in an isothermal system. The difference between that example and this one, is thus that the temperature was not taken into account in the paper. The paper also used other variables and parameters, as well as a different computer, but it is still interesting to see the similarities of computational cost for that example and

Table 5.15: The temperature discrepancy between the multiscale and sequential method when the advection term and diffusion term are of the same order, $a = \mathcal{O}(d)$, when the order of the advection term is 1000 times bigger than that of the diffusion term, $a = \mathcal{O}(10^3 d)$, and when the order is 1000 times smaller than the diffusion term, $a = \mathcal{O}(10^{-3} d)$. A time step of $\Delta t = 20$ days is used, and $tol_{iter} = 10^{-1}$.

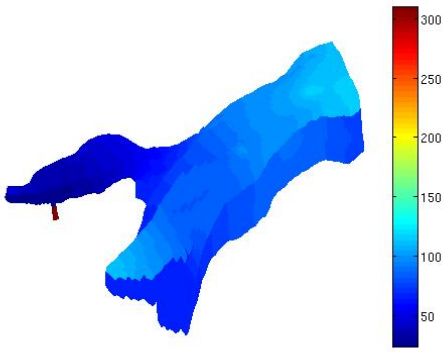
	$a = \mathcal{O}(d)$		$a = \mathcal{O}(10^3 d)$		$a = \mathcal{O}(10^{-3} d)$	
Time (days)	L^2	L^∞	L^2	L^∞	L^2	L^∞
t = 20	$6.435 \cdot 10^{-7}$	$1.657 \cdot 10^{-5}$	$1.669 \cdot 10^{-6}$	$1.318 \cdot 10^{-4}$	$6.432 \cdot 10^{-7}$	$1.654 \cdot 10^{-5}$
t = 1000	$2.019 \cdot 10^{-5}$	$4.570 \cdot 10^{-4}$	$3.833 \cdot 10^{-4}$	$3.352 \cdot 10^{-2}$	$2.016 \cdot 10^{-5}$	$4.567 \cdot 10^{-4}$
t = 2000	$9.970 \cdot 10^{-5}$	$1.517 \cdot 10^{-3}$	$1.472 \cdot 10^{-4}$	$9.472 \cdot 10^{-3}$	$9.985 \cdot 10^{-5}$	$1.519 \cdot 10^{-3}$



(a) $a = \mathcal{O}(d)$



(b) $a = \mathcal{O}(10^3 d)$



(c) $a = \mathcal{O}(10^{-3} d)$

Figure 5.29: The Norne temperature solution when $t = 5$ years, 175 days for the three different cases of advection.

this one. The paper found that the multiscale solver was 2-3 times as fast as the sequential solver. We have that the sequential solver uses 16.9 minutes to solve our problem, while the multiscale solver uses 8.5 minutes to solve the same problem. Our multiscale method is thus about 2 times faster than the sequential

method. So even though we have added temperature to the system, the run time of two multiscale methods seems to remain somewhat constant compared to each other. This is good, because it indicates that the multiscale method does not loose any of its already proven efficiency by adding temperature to the system.

6

CONCLUSION

In this thesis we have looked at how two methods, the sequential method and the multiscale restriction-smoothed basis method, solve equations that describe flow and heat transfer in porous media. Here, we will give some concluding remarks on both methods, and propose some ideas for further works.

SEQUENTIAL METHOD

We have seen that the sequential method gives solutions that manage to mimic the fully implicit solutions fairly accurately. This is true both for problems with fewer cells and homogeneous permeabilities, as well as for more complex problems with a higher number of cells and heterogeneous permeabilities. The method is more efficient than the fully implicit method, both when it comes to the easy test case, and the more complex one. In addition, we have with the sequential method a more flexible solver than the fully implicit method, and by adjusting properties such as step size, tolerances and outer loops, the sequential method is able to systematically trade accuracy for efficiency. The sequential method is also more stable than the fully implicit method, as we found no restriction on the step size with the former method. This is an advantage in reservoir simulation, where we often want to simulate processes over several years, maybe even decades. A restriction we did find was that we have to solve for pressure before solving for temperature, but this is not a big hindrance in our opinion.

MULTISCALE METHOD

By applying the MsRSB method on the systems obtained from the sequential method, we get a solver that more efficiently solves problems concerning non-isothermal flow in porous media. The method produces accurate temperature and pressure solutions for both single-phase and multiphase problems. Furthermore, it works well on both two dimensional and three dimensional grids, as well as both structured Cartesian grids and complex unstructured grids with

fractures, cracks and uneven surfaces. Just as with the sequential method, the multiscale method is a stable and flexible method, where you can, by adjusting different parameters, sharpen or lessen the accuracy to the sequential method. Adjusting the iterative tolerance is especially effective on the accuracy. Furthermore, we have that the multiscale method produces accurate solutions when quite coarse grids are used. We thus have a method that manages to mimic the sequential method to a large degree, and by adjusting the sequential method's parameters, the multiscale method's solutions will converge towards the fully implicit solution. We have to be a little careful though. Adjusting the time step Δt will make the sequential method's solutions more accurate compared to the fully implicit's ones, but it will make the multiscale's solutions less accurate compared to the sequential method, and thus also the fully implicit. At least when high iterative tolerances are used. So we have to find the right balance.

When it comes to efficiency, the multiscale method is best on complex grids and complex permeabilities. When we have simpler problems with homogeneous permeabilities, the multiscale method will not produce the solutions more efficiently than for instance the sequential method, and there is not much of a point of using the multiscale method. When we have more complex problems, however, the multiscale method is clearly more efficient than the sequential method. We also have that the multiscale method works best on elliptical problems. By using a more elliptic heat equation in the SPE10, Layer 5 example, we saw that the noise previously encountered when the iterative tolerance was 10^{-1} disappeared. Moreover, by using a more elliptic equation we generally needed fewer multiscale iterations for the solutions to converge, and by using a more hyperbolic heat equation, the accuracy decreased. The multiscale's accuracy does not seem to be much effected by a change of the heat conductivity κ. We did not find much change in the accuracy by switching to a heterogeneous κ, but this might come from the fact that κ still had large areas with constant values.

There does not seem to be any loss of accuracy or efficiency by going from isothermal flow to thermal flow. Both the pressure and temperature discrepancies are small, and the efficiency seems to be somewhat constant. In fact, a previous paper [18] found that the multiscale method was two to three times as fast as the sequential method on the Norne field when they studied isothermal flow, and we found the multiscale solver to be about two times as fast when we added temperature to the model. We also have that the temperature system uses fewer multiscale iterations to converge than the pressure system.

All in all, we have a method that for problems with complex grids and permeabilities is flexible, stable and more efficient than the sequential and fully implicit method, and which produces solutions that mimic the sequential solutions accurately. The method has previously been proven to work well on isothermal flow, but we can now conclude that it also works well when temperature is

added to the model. By adjusting the parameters of the sequential method, the multiscale method will further converge towards fully implicit method.

FURTHER WORK

When it comes to further work, we suggest studying the effects of a heterogeneous heat conductivity κ more thoroughly. Most of this thesis has employed a homogeneous heat conductivity. Real reservoirs has a heterogeneous κ however, and it is therefore important to know how the multiscale method is able to handle this. As previously described, when we tested a heterogeneous κ it did not lead to big changes in the accuracy. This was a bit surprising, as we would think that the multiscale method would handle a heterogeneous κ differently than it would handle a homogeneous one. We believe the small change is a result of the relatively small change in the conductivity, though, especially considering the fact that the permeability can vary over many orders. It would therefore be interesting to study even more realistic cases of the heat conductivity. As κ describes a materials ability to transfer heat, it is an important factor in heat transfer in real life, and it is therefore important that our model is as accurate as possible when a realistic value is used.

Another element that should be pursued is to study what happens when we add gas to our model. Gas is a natural element of petroleum reservoirs, and the methods should therefore be tested and verified on the full three-phase model derived in Section 2.2.2. In addition to this comes the fact that adding a third phase will lead to a slightly more complex model, and it would be intriguing to see whether the methods are able to handle the added complexity or not.

BIBLIOGRAPHY

[1] Z. Chen and T.Y. Hou. A mixed multiscale finite element method for elliptic problems with oscillating coefficients. *Mathematics of Computation*, 72(242):541–576, 2003.

[2] Z. Chen, G. Huan, and B. Li. An improved impes method for two-phase flow in porous media. *Transport in porous media*, 54(3):361–376, 2004.

[3] Z. Chen, G. Huan, and Y. Ma. *Computational methods for multiphase flows in porous media*, volume 2. Siam, 2006.

[4] M.A. Christie, M.J. Blunt, et al. Tenth spe comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Evaluation & Engineering*, 4(04):308–317, 2001.

[5] R.D. Falgout. An introduction to algebraic multigrid. *Computing in science & engineering*, 8:24, 2006.

[6] Y. Gautier, M. J. Blunt, and M. A. Christie. Nested gridding and streamline-based simulation for fast reservoir performance prediction. *Computational Geosciences*, 3(3-4):295–320, 1999.

[7] H. Hajibeygi. *Iterative multiscale finite volume method for multiphase flow in porous media with complex physics*. PhD thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 19872, 2011, 2011.

[8] S. T. Hilden, O. Møyner, K.-A. Lie, and K. Bao. Multiscale simulation of polymer flooding with shear effects. 2015.

[9] T. Y. Hou and X.-H. Wu. A multiscale finite element method for elliptic problems in composite materials and porous media. *Journal of computational physics*, 134(1):169–189, 1997.

[10] NTNU IO center. Norne benchmark case. `http://www.ipt.ntnu.no/~norne/wiki/doku.php`, 2016. [Online; accessed 21-January-2016].

[11] P. Jenny, S.H. Lee, and H.A. Tchelepi. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *Journal of Computational Physics*, 187(1):47–67, 2003.

[12] P. Jenny, S.H. Lee, and H.A. Tchelepi. Adaptive multiscale finite-volume method for multiphase flow and transport in porous media. *Multiscale Modeling & Simulation*, 3(1):50–64, 2005.

[13] S. Krogstad, K.-A. Lie, O. Moyner, H. M. Nilsen, X. Raynaud, and B. Skaflestad. Mrst-ad - an open-source framework for rapid prototyping and evaluation of reservoir simulation problems. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 2015.

[14] L.W. Lake. *Enhanced oil recovery*. Old Tappan, NJ; Prentice Hall Inc., 1989.

[15] K.-A. Lie. *An Introduction to Reservoir Simulation Using MATLAB User Guide for the MATLAB Reservoir Simulation Toolbox (MRST)*. SINTEF ICT, Department of Applied Mathematics, 2014. `http://www.sintef.no/Projectweb/MRST/Publications`.

[16] T. Manzocchi, J.N. Carter, A. Skorstad, B. Fjellvoll, K.D. Stephen, J.A. Howell, J.D. Matthews, J.J. Walsh, M. Nepveu, C. Bos, et al. Sensitivity of the impact of geological uncertainty on production from faulted and unfaulted shallow-marine oil reservoirs: objectives and methods. *Petroleum Geoscience*, 14(1):3–15, 2008.

[17] O. Møyner and K.-A. Lie. A multiscale two-point flux-approximation method. *Journal of Computational Physics*, 275:273–293, 2014.

[18] O. Møyner and K.-A. Lie. A multiscale restriction-smoothed basis method for compressible black-oil models. 2015.

[19] O. Møyner and K.-A. Lie. A multiscale restriction-smoothed basis method for high contrast porous media represented on unstructured grids. *Journal of Computational Physics*, 304:46–71, 2016.

[20] O. Møyner, K.-A. Lie, et al. A multiscale method based on restriction-smoothed basis functions suitable for general grids in high contrast media. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers, 2015.

[21] K.C. Park. Stabilization of partitioned solution procedure for pore fluid-soil interaction analysis. *International Journal for Numerical Methods in Engineering*, 19(11):1669–1673, 1983.

[22] D. W. Peaceman. *Fundamentals of numerical reservoir simulation*. Elsevier, 2000.

[23] D. W. Peaceman et al. Interpretation of well-block pressures in numerical reservoir simulation (includes associated paper 6988). *Society of Petroleum Engineers Journal*, 18(03):183–194, 1978.

[24] Y. Saad. *Iterative methods for sparse linear systems*. Siam, 2003.

[25] J.W. Sheldon, W.T. Cardwell Jr, et al. One-dimensional, incompressible, noncapillary, two-phase fluid flow in a porous medium. 1959.

[26] SINTEF. MRST - MATLAB reservoir simulation toolbox. http://www.sintef.no/projectweb/mrst/, 2015. [Online; accessed 4-January-2016].

[27] The Enginering ToolBox. Thermal conductivity of materials and gases. http://www.engineeringtoolbox.com/thermal-conductivity-d_429.html, 2016. [Online; accessed 20-January-2016].

[28] J. A. Trangenstein and J. B. Bell. Mathematical structure of compositional reservoir simulation. *SIAM journal on scientific and statistical computing*, 10(5):817–845, 1989.

[29] J. A. Trangenstein and J. B. Bell. Mathematical structure of the black-oil model for petroleum reservoir simulation. *SIAM Journal on Applied Mathematics*, 49(3):749–783, 1989.

[30] S.B. Vennemo. Application of the multiscale restricted-smoothed basis method on temperature equations.

[31] C. Wolfsteiner, S. H. Lee, and H. A. Tchelepi. Well modeling in the multiscale finite volume method for subsurface flow simulation. *Multiscale Modeling & Simulation*, 5(3):900–917, 2006.

[32] O.C. Zienkiewicz, D.K. Paul, and A.H.C. Chan. Unconditionally stable staggered solution procedure for soil-pore fluid interaction problems. *International Journal for Numerical Methods in Engineering*, 26(5):1039–1055, 1988.