Ph.D. Thesis

Production Optimization of Oil Reservoirs

Carsten Völcker

Department of Informatics and Mathematical Modelling
Technical University of Denmark
Kongens Lyngby

IMM-PHD-2011-265

Preface

This thesis was prepared at the Department of Informatics and Mathematical Modelling and the Center for Energy Resources Engineering, Technical University of Denmark (DTU), in partial fulfillment of the requirements for receiving the Ph.D. degree. The work presented in this thesis was carried out from January 2008 to March 2011.
During the course of the work presented in this thesis, a number of people have provided their help and support, for which I am very grateful. First of all I would like to thank my supervisors at DTU, John Bagterp Jørgensen, Per Grove Thomsen and Erling Halfdan Stenby. I would also like to thank Morten Rode Kristensen, Hans Bruun Nielsen, Jan Frydendall, Allan Peter Engsig-Karup, Bernd Dammann and Fabrice Delbary.
Finally, I wish to thank my family, girlfriend and friends for continuous support.

Kongens Lyngby, October 2011

Carsten Völcker

Summary

With an increasing demand for oil and difficulties in finding new major oil fields, research on methods to improve oil recovery from existing fields is more necessary now than ever. The subject of this thesis is to construct efficient numerical methods for simulation and optimization of oil recovery with emphasis on optimal control of water flooding with the use of smart-well technology.

We have implemented immiscible flow of water and oil in isothermal reservoirs with isotropic heterogenous permeability fields. We use the method of lines for solution of the partial differential equation (PDE) system that governs the fluid flow. We discretize the two-phase flow model spatially using the finite volume method (FVM), and we use the two point flux approximation (TPFA) and the single-point upstream (SPU) scheme for computing the fluxes.

We propose a new formulation of the differential equation system that arises as a consequence of the spatial discretization of the two-phase flow model. Upon discretization in time, the proposed equation system ensures the mass conserving property of the two-phase flow model. For the solution of the spatially discretized two-phase flow model, we develop mass conserving explicit singly diagonally implicit Runge-Kutta (ESDIRK) methods with embedded error estimators for adaptive step size control. We demonstrate that high order ESDIRK methods are more efficient than the low-order methods most commonly used in reservoir simulators. Most commercial reservoir simulation tools use step size control, which is based on heuristics. These can neither deliver solutions with predetermined accuracy or guarantee the

convergence in the modified Newton iterations. We have established predictive step size control based on error estimates, which can be calculated from the embedded ESDIRK methods. We change the step size control in order to minimize the computational cost per simulation. To our knowledge, there have been no previous attempts in applying ESDIRK integration or error based step size control for computation of the water flooding process, neither for commercial purposes nor in simulators developed for research purposes, with exception of the work that we present in this thesis.
We implement a numerical method for nonlinear model predictive control (NMPC) along with smart-well technology to maximize the net present value (NPV) of an oil reservoir. The optimization is based on quasi-Newton sequential quadratic programming (SQP) with line-search and BFGS approximations of the Hessian, and the adjoint method for efficient computation of the gradients. We demonstrate that the application of NMPC for optimal control of smart-wells has the potential to increase the economic value of an oil reservoir.

This thesis consists of a summary report and five research papers submitted, reviewed and published in proceedings in the period 2009 - 2011.

Resumé

Med en stigende efterspørgsel efter olie og vanskeligheder med at finde nye store oliefelter, er forskning i metoder til at forbedre olieudvinding fra de eksisterende felter mere nødvendig nu end nogensinde. Emnet for denne afhandling er at konstruere effektive numeriske metoder til simulering og optimering af olieudvinding med særlig vægt på optimal kontrol af vandinjektion med brug af smart-well teknologi.
Vi har implementeret ikke-blandbar strømning af vand og olie i isoterme reservoirer med isotrope heterogene permeabilitets felter. Vi bruger "method of lines" til løsning af det partielle differentialligningssystem (PDE), der modellerer væskestrømmen. Vi diskretiserer to-fase strømningsmodellen spatialt ved hjælp af finite volume metoden (FVM), og vi bruger to-punkts flux tilnærmeslen (TPFA) og et-punkts opstrøms metoden (SPU) til at beregne fluxen af væskerne.
Vi foreslår en ny formulering af det differentialligningssystem, der opstår som følge af den spatiale diskretisering af to-fase strømningsmodellen. Det ny differentialligningssystem sikrer to-fase strømningsmodellens massebevarende egenskab under temporal diskretisering. Til løsning af den spatialt diskretiserede to-fase strømningsmodel, udvikler vi massebevarende "explicit singly diagonally implicit Runge-Kutta" (ESDIRK) metoder med ind-byggede fejl estimatorer til adaptiv skridtlængdekontrol. Vi viser, at højere ordens ESDIRK metoder er mere effektive end de lav-ordens metoder, der oftest anvendes i reservoir simulatorer. De fleste kommercielle reservoir simuleringsværktøjer bruger skridtlængde kontrol, som er baseret på heuristikker. Disse kan hverken levere løsninger med forudbestemt nøj-

agtighed eller garantere konvergens i de modificerede Newton iterationer. Vi har etableret prædiktiv skridtlængde kontrol baseret på fejlestimater, der kan beregnes ved hjælp af de indlejrede ESDIRK metoder. Vi ændrer skridtlængdekontrollen med henblik på at minimere den beregningsmæssige omkostning per simulering. Så vidt vi ved, er der ikke tidligere gjort forsøg på at anvende ESDIRK integration eller skridtlængde kontrol baseret på fejlestimater til beregning af vandinjektionsprocessen, hverken til kommercielle formål eller i simulatorer udviklet til forskningsformål, med undtagelse af det arbejde, som vi præsenterer i denne afhandling.
Vi implementerer en numerisk metode til ikke-lineær modelbaseret kontrol (NMPC) sammen med smart-well teknologi til at maksimere nutidsværdien (NPV) af et oliereservoir. Optimeringen er baseret på quasi-Newton sekventiel kvadratisk programmering (SQP) med linie-søgning og BFGS approksimationer af Hessian matricen, og den adjungerede metode til effektiv beregning af gradienterne. Vi viser, at anvendelsen af NMPC til optimal kontrol af smart-wells har potentiale til at øge den økonomiske værdi af et oliereservoir.

Denne afhandling består af en sammenfattende rapport samt fem forskningsartikler indsendt, revideret og offentliggjort i perioden 2009 - 2011.

Contents

CHAPTER 1

Introduction

The motivation and background for the work presented in this thesis is given in Section 1.1. Section 1.2 is an overview of relevant literature on reservoir optimization. In Section 1.3 we summarize the objectives and key contributions of this work, and in Section 1.4 we outline the remainder of the thesis.

1.1 Motivation

The global oil consumption is increasing and the most available resources are being exhausted. On the same time it gets increasingly more difficult to find new major oil fields. Consequently, future oil production may become highly demanding financially as well as technologically. Either by producing from complicated reservoirs (e.g. deep sea reservoirs, Arctic environment, extra heavy oil or oil sands) or by producing oil remaing in reservoirs after current conventional production. Currently, it is expected that the world's oil fields have an average recovery below 50%. With an increasing demand for oil and difficulties in finding new major oil fields, the research in efficient exploration of existing fields is becoming increasingly important. In particular, this

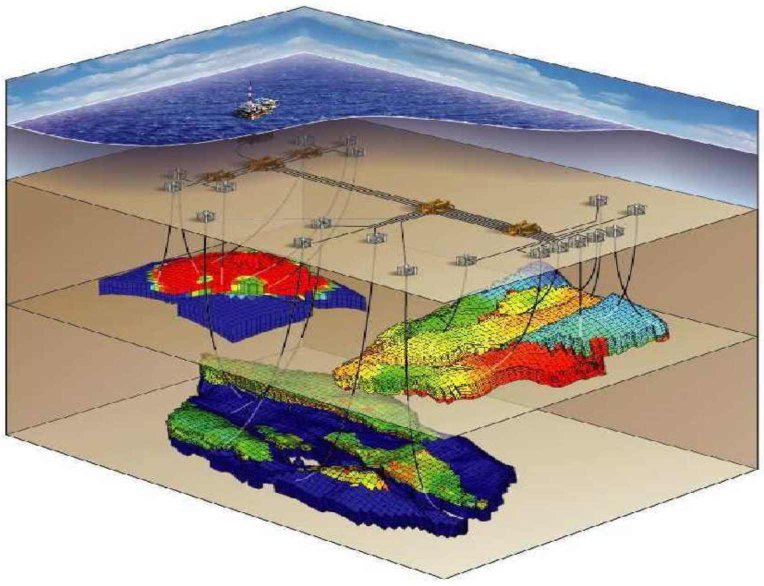


Figure 1.1: An off-shore oil field with multiple horizontal wells.

is highly relevant for the Danish North Sea oil fields where the expected average oil recovery is less than 30%.
Oil is produced from subsurface reservoirs, which are formations of porous rock, enclosed by impermeable layers. The reservoir fluids, mainly oil, gas and water, are contained inside the microscopic pores of the rock under high temperature and pressure. The reservoir rock is not only porous but also permeable, i.e. the pores are interconnected, and we may induce fluid flow by adding a pressure gradient in the reservoir. In particular, we will consider two-phase flow, which describes combined flow of water and oil in reservoirs, exploited as a mechanism for oil production optimization.
The development of an oil field essentially consists of drilling wells into the reservoir rock and connecting them to surface facilities from which the oil can be transported to refineries for processing, see Figure 1.1. In general, the depletion process of a reservoir consists of two production phases,

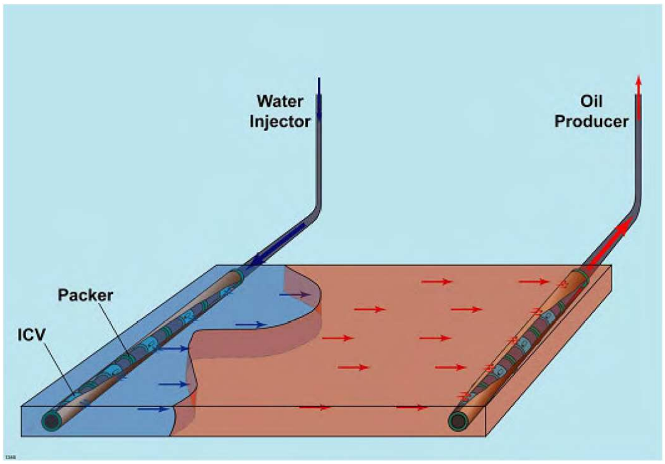


Figure 1.2: Two horizontal smart-wells in the water flooding problem [1].

however, a third phase using enhanced oil recovery (EOR) techniques may be used if economy permits. In the primary production phase, the initial reservoir pressure will be the driving mechanism for the production. After drilling the wells, oil will start flowing to the surface by itself. During this phase, the pressure gradually decreases, and ultimately it will not be high enough to push the reservoir fluids to the production facilities. Most of the oil will remain in the reservoir by the use of this passive approach alone. In the secondary recovery phase, liquid or gas is injected into the reservoir. The most commonly used secondary recovery mechanism involves injection of water, referred to as water flooding. The purpose is to sustain reservoir pressure and to displace the oil from the injection wells toward the production wells, see Figure 1.2. Another similar recovery method is the injection of CO_2. However, CO_2 injection is far more complicated, mainly because of the involved logistics, and the method is not usually classified as a secondery recovery method. Even such techniques will leave the majority of the oil trapped in the pores of the reservoir rock, and the oil recovery factor will in most cases stay somewhere between 10% to 50%. Sometimes a

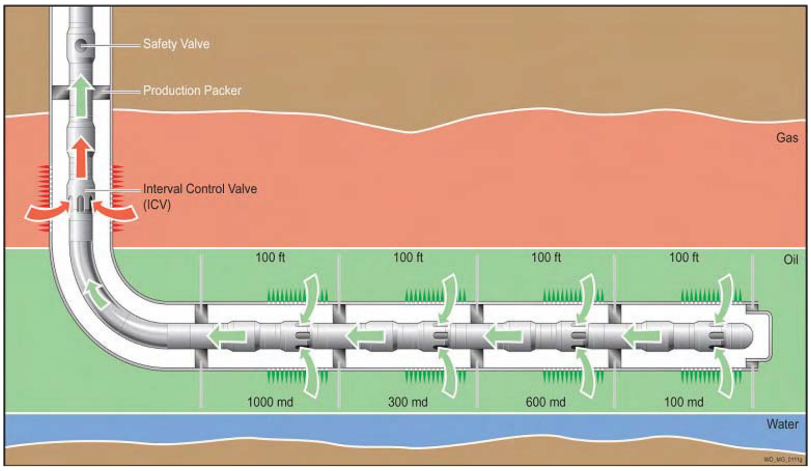


Figure 1.3: A smart-well with individually controllable segments [2].

further increase in the recovery factor is possible during a tertiary recovery phase using enhanced oil recovery (EOR) methods: chemical flooding by injection of e.g. surfactants or polymers, microbial EOR, steam injection, and in-situ combustion. However, these techniques are relatively expensive and not economically feasible even at the current high level oil prices.
Alternatively, careful planning of the water flooding process may increase the oil recovery factor without doing excessive investments. The most commonly used strategy is a reactive flooding control, based on measuring the oil-water ratio in each well. In case of water breakthrough, that is, in case that injected water reaches a production well, a shut-off threshold is applied. This reaction based strategy suffers from two major drawbacks: the long-term time horizon from initial water injection until water breakthrough makes proper feedback control unfeasible, and the irreversible development of an oil field makes learning based algorithms for control unfeasible too. Together with the fact that water flooding behaves in a non-linear fashion, such a simplified strategy often leads to poor sweep efficency. That is, certain regions of the reservoir is not drained sufficiently, because injected

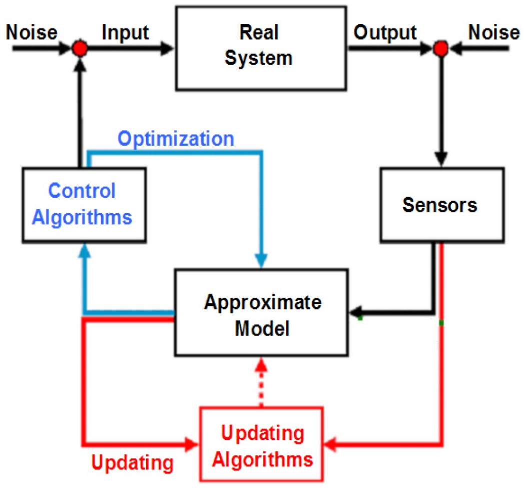


Figure 1.4: Block diagram af closed-loop optimization [3].

water may pass by the oil in those regions and penetrate a production well instead.
The use of modern reservoir simulation tools based on physical reservoir models may to some extend resolve this problem. In particular, a combination of reservoir modeling and nonlinear model predictive control (NMPC) has the potential to provide long-term predictions of fluid flow through the reservoir, which can be used to plan future development strategies. In model based reservoir management, the water flooding process is optimized by adjustment of smart-wells containing down-hole measurement and control equipment as well as individually controllable segments, see Figure 1.3. In the sense of closed-loop optimization, these measurements together with seismic interpretations of the subsoil may be used for frequent updating of the reservoir model, whereas an optimal control strategy can be computed based on the regularly updated model, see Figure 1.4.

1.2 Literature Review

Research on optimization of oil reservoirs using gradient based algorithms and the adjoint approach for gradient computation has been conducted by other authors. There are three major topics when it comes to optimization of oil reservoirs: optimal well placement, history matching and production optimization. Both optimal well placement and history matching are outside the scope of this work. We will therefore only briefly comment on these two subjects. Regarding production optimization, we will mainly focus on research that involves the adjoint method, when applied to the water flooding process.
Drilling of wells is extremely expensive. Thus, determination of the number, type and location of wells are among the most important decision factors in the early stages of reservoir development planning. Optimal well placement is therefore an issue of ongoing research [4, 5, 6]. Common for these groups is the use of gradient based optimization algorithms and the adjoint model for gradient computation.
The first applications of gradient based optimization using the adjoint approach in the oil production industry was for history matching. History matching used for reservoir optimization was pioonered by [7, 8].
Within the research area of production optimization, [9] was among the first to formulate the problem of production optimization in the context of an optimal control problem, using the adjoint method for gradient computation of the objective function. His work was mainly focused on tertiary recovery techniques, such as chemical flooding. Later, [10, 1] optimized the water flooding process, see Figure 1.2, and demonstrated that smart-well technology, see Figure 1.3, combined with optimal control has the potential to increase net present value (NPV) of a reservoir. Through their work, gradient based optimization using the adjoint model has received significant attention in the society of petroleum engineers. Smart-wells are either completed with bang-bang type valves, i.e. on-off valves, or completed with variable-setting valves. It has been shown that the shape of the optimal solutions will sometimes be of the bang-bang type [11, 12]. This type of solution was also to some extent demonstrated in [10, 1]. Solutions of the bang-bang

type may be implemented with simple on-off valves, which has the advantage of being less expensive than variable-setting valves. That is why the shape of optimal solutions is still an open issue in the reservoir commuity. Based on their work, several research groups have become interested in the use of smart-well technology combined with optimal control. Common for all groups is the use of gradient based optimization methods that include the adjoint model for gradient computation [13, 14, 15, 16, 17]. Numerous other issues related to production optimization of oil reservoirs are open to ongoing research. In particular, state constraint handling [18, 14, 15, 19, 20, 21], which is a very important topic in practical reservoir management, e.g. bounds on the reservoir pressure to avoid fracturing of the reservoir rock or limits on the amount of produced water. Another topic of relevance, is the calculation of the second derivative, the Hessian, of the objective function. The most common approach is to use BFGS approximations of the Hessian. However, to gain more accuracy and better convergence toward an optimal solution, second order adjoints may be an alternative [22].
Both history matching and production optimization are building blocks in a closed-loop optimizer, see Figure 1.4, in which the red loop represents the history matching, while the blue loop represents the production optimization. Production optimization is an open-loop optimizer, which is based on response from the reservoir model, whereas history matching updates the model based on production data and seismic interpretations. Long-term reservoir management requires a closed loop approach, where properties such as permeability and porosity of the reservoir rock are updated frequently and uncertainties on such properties are quantified. It is most common to close the loop using the ensemble Kalman filter (EnKF) for history matching in conjunction with a gradient based algorithm for production optimization. The EnKF provides model updates and quantifies model uncertainties based on an ensemble of different reservoir properties, e.g. ensembles of permeability and porosity. Some of the earliest attempts of linking the EnKF and gradient based optimization using the adjoint model for gradient computation can be found in [23, 10, 24, 25, 26]. There has also been attempts on combining an estimator and an optimizer, where both production data misfits and production optimization has been com-

puted using the adjoint approach [27, 28]. However, the methods described above for history matching and production optimization all have the potential to increase the economic value of an oil reservoir. This is demonstrated in a recent benchmark study of the water flooding process performed on a synthetic data set [29].

1.3 Objectives and Main Contributions

The primary objective of the work in this thesis is the implementation of a framework for oil production optimization by water flooding. We will use an explicit singly diagonally implicit Runge-Kutta (ESDIRK) method as a tool for efficient integration of the reservoir model. To achieve this we must understand the underlying processes of both reservoir simulation and methods for production optimization. Reservoir models may generate large-scale optimization problems. That is why single-shooting using adjoints for gradient computation is the most widely used method for production optimization of oil reservoirs, and it is also the reason why we have chosen this method. Good convergence in the single-shooting method requires an efficient and accurate numerical integration. ESDIRK methods with adaptive step size control has such properties.
Computation of adjoints involves the gradients of the model. These are computed by ESDIRK integration during simulations. Thus, to efficiently compute the adjoints, we need an implementation of both model and integrator that facilitates the reuse of gradient information. This suggests that we need a reservoir simulator, a method for numerical integration and an adjoint model, where we have full access to the source code.
For the simulation we implement a two-phase flow simulator. We spatially discretize the flow equations by the finite volume method (FVM), in which we use the two-point flux approximation (TPFA) and single-point upstream (SPU) weighting of the fluid terms. We use vertical injection and production wells of the Peaceman type.
For the numerical integration we develop and implement mass conserving ESDIRK methods with embedded error estimators for adaptive step size

control. The two-phase flow problem is based on conservation of mass. We propose a new differential equation model that upon temporal discretization maintains such a property. The spatially discretized two-phase flow problem can be directly formulated using this model.
For the optimization we use a gradient based algorithm, in which we include the adjoint model for computing the gradient of the objective function. We use BFGS updates for computing the Hessian of the objective function. We do not have access to any real production data, thus, we perform open-loop optimization (the blue loop in Figure 1.4). We focus on optimal control of injection rates and bottom hole pressures (BHPs) of injection wells and production wells, respectively. The objective is to increase oil production using water flooding and thereby maximize net present value (NPV) of the reservoir.
The main contributions of this work are:

- The formulation of a new differential equation model that upon temporal discretization maintains the mass conserving property of the spatially discretized two-phase flow problem.
- The development of mass conserving ESDIRK methods with embedded error estimators for adaptive step size control.
- The application of ESDIRK integration and error based step size control for computation of the water flooding process.

To our knowledge, there have been no previous attempts on applying ESDIRK methods and error based step size control for computation of the water flooding process. Neither for commercial purposes nor in simulators developed for research purposes, with exception of the work that we present in this thesis and in the papers A - E, included in this thesis.

1.4 Outline

This thesis is divided into 5 chapters and 8 appendices. The background and motivation behind the project is given in Chapter 1. The model of

the combined flow of water and oil in a reservoir is decribed in Chapter 2, together with a description of the well models. The reservoir model is essentially a system of partial differential equations. The methods for spatial and temporal discretization that we use to extract the solution from this system is given in Chapter 3 and 4, respectively. In Chapter 5 we present the water flooding process and the problem of maximizing net present value of a reservoir. Chapter 6 concludes the study.
In Appendix A we describe the derivation of selected Jacobian elements in details. Appendix B includes the coefficients for the numerical integration methods. Appendix C contains a list of the physical quantities that describes the reservoir model and the well models along with a list of commonly used abbreviations. In appendix D - H we include five conference papers A - E that have been reviewed, presented and published in proceedings. The material presented in the papers and in the thesis overlap to some extend. However, they are complementary, since both contain details that are not presented elsewhere.

CHAPTER 2

Reservoir Model

Oil reservoirs are characterized by complex geometry, spatially variable geological properties, e.g. porosity and permeability of a porous medium, and complex fluid mixtures of water and multiple oil and gas components. In reservoir simulation, we are interested in describing the transport of these different components through the porous medium. In general, a component can exist in any fluid phase, and as a result we must solve one equation per component times a set of phase equilibrium relations. Compositional reservoir models, which treats every component in every fluid phase individually, are computationally expensive to simulate, even with today's modern supercomputers. The complexity of the model can be reduced by lumping the hydrocarbon components into pseudo components. The black oil model, which is common in the petroleum industry, is a simplified isothermal compositional reservoir model. Besides water, it contains only two pseudo hydrocarbon components: oil and gas. In the black oil model it is assumed that water and oil are immiscible, and that water does not vaporize into the gas phase. Thus, the water phase consists of the water component alone. The gas can dissolve in both the water phase and the oil phase, and it is not uncommon that black oil models allow oil to vaporize into the gas

phase.
With respect to computation time it is less demanding to simulate the black oil model compared to compositional simulation. However, the physics are treated less realistically in the black oil formulation. Because of the reduced computational cost of the black oil model, this formulation is relevant in production optimization, which requires frequent updating/simulation of the reservoir model. In the present work we use a simplified version of the black oil formulation: a two-phase system containing only water and oil with complete immiscibility [30, 31, 32, 33, 34]. The simplification eliminates the phase equilibrium relation due to the solubility of gas in the oil phase. In this way we reduce the computational cost per simulation. The simplified description of the reservoir fluids is sufficient for demonstrating the production optimization technique utilized in Chapter 5.
This chapter is organized as follows. In the first two sections we present the equations that govern the two-phase flow problem together with the initial and boundary conditions. In Section 2.3 we describe the constutive models, divided into properties concerning the reservoir rock and properties concerning the reservoir fluids. Section 2.4 describes the transport model governed by Darcy's law. The well models [35] that we use are presented in Section 2.5. Finally, in Section 2.6 we discuss how to derive the primary variables of the system.

2.1 Governing Equations

Consider the spatial domain $\Omega \subset \mathbb{R}^3$ and the time domain $T = \{t \in \mathbb{R} : t \geq 0\}$. The boundary of the spatial domian is $\partial\Omega \subset \mathbb{R}^3$ and the boundary of the time domain is $\partial T = \{t \in \mathbb{R} : t = 0\}$. The water and oil phases are indexed as $\alpha \in \{w, o\}$. Let $C_\alpha = C_\alpha(t, \mathbf{s})$, be the mass concentrations of water and oil in the reservoir (mass per unit volume of reservoir) as functions of time $t \in T$ and position $\mathbf{s} \in \Omega$. The mass balances of the reservoir fluids are expressed by the following system of partial differential equations

$$\frac{\partial C_\alpha}{\partial t} = -\nabla \cdot \mathbf{F}_\alpha + Q_\alpha \quad t \in T \setminus \partial T \quad \mathbf{s} \in \Omega \setminus \partial\Omega \tag{2.1}$$

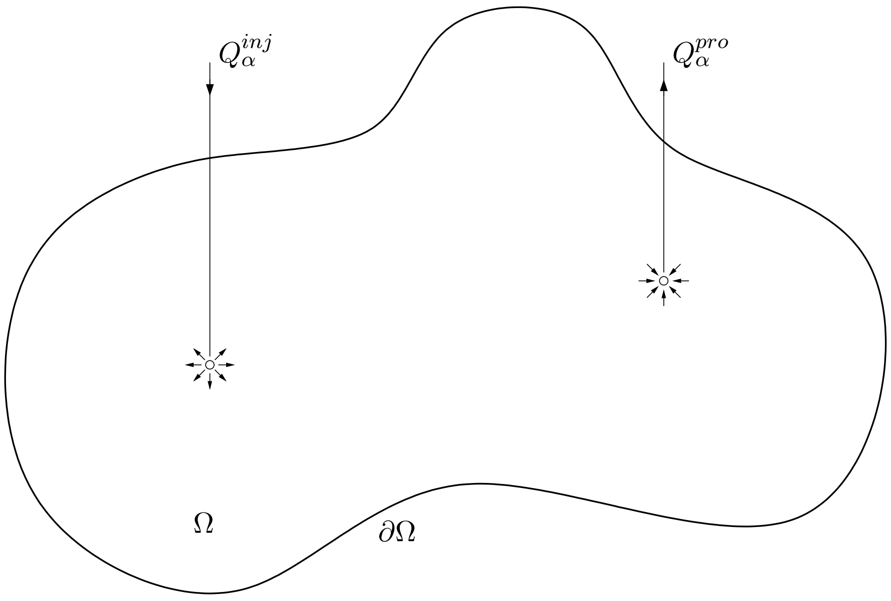


Figure 2.1: Outline of a reservoir with internal source/sink ($Q_\alpha^{inj}/Q_\alpha^{pro}$) terms.

in which $\mathbf{F}_\alpha = \mathbf{F}_\alpha(t, \mathbf{s})$ is the flux of water and oil through the porous medium. The flux is a 3-dimensional vector, $\mathbf{F}_\alpha \in \mathbb{R}^3$, describing the mass flow in each spatial coordinate direction. The source/sink terms of water and oil are denoted $Q_\alpha = Q_\alpha(t, \mathbf{s})$. They are used to describe the flow from injection wells and the flow to production wells inside the reservoir. The reservoir is outlined in Figure 2.1.

2.2 Initial and Boundary Conditions

The initial concentrations of the reservoir fluids are specified

$$C_\alpha(t, \mathbf{s}) = C_\alpha(\mathbf{s}) \quad t \in \partial T \quad \mathbf{s} \in \Omega \tag{2.2}$$

The boundary conditions of the model incorporate external and internal conditions. External boundary conditions define spatial limits of the reservoir. Internal boundary conditions define well placements. Both external and internal conditions are specified by defining flow rates across a boundary or pressure at a boundary, which corresponds to Neumann or Dirichlet type conditions. For external boundaries we will assume a Neumann condition corresponding to no flow across the reservoir boundaries

$$\mathbf{F}_\alpha(t, \mathbf{s}) = 0 \quad t \in T \quad \mathbf{s} \in \partial\Omega \tag{2.3}$$

Internal boundaries due to injection and production wells are treated seperately in Section 2.5.

2.3 Constitutive Models

The concentrations of water and oil in the reservoir may be expressed as

$$C_\alpha = \phi \rho_\alpha S_\alpha \tag{2.4}$$

ϕ is the porosity of the reservoir rock. $\rho_\alpha = \rho_\alpha(P_\alpha)$ is the density of each reservoir fluid in particular. The density depends on the pressure $P_\alpha = P_\alpha(t, \mathbf{s})$ in the fluid. $S_\alpha = S_\alpha(t, \mathbf{s})$ is the saturation of the fluid. The saturation represents the volumetric fraction of the fluid occupying the void space (pore space volume). As water and oil jointly fill the entire void space of the reservoir, the following saturation constraint holds

$$S_w + S_o = 1 \tag{2.5}$$

Water and oil are transported by convection through the porous medium. Therefore, the fluxes can be expressed as

$$\mathbf{F}_\alpha = \rho_\alpha \mathbf{u}_\alpha \tag{2.6}$$

$\mathbf{u}_\alpha = \mathbf{u}_\alpha(P_w, S_w)$ are the velocities at which the fluid phases individually flow through the porous medium. They are modelled in terms of Darcy flow in Section 2.4.

To complete the two-phase flow model we must introduce several other quantities, namely permeability, relative permeability, capillary pressure and viscosity. Porosity and permeability are quantities that characterize the porous medium in which the reservoir fluids reside. Density, relative permeability, capillary pressure and viscosity are all properties of the reservoir fluids. In the following we will describe these quantities and provide the models needed for completing the two-phase flow model.

2.3.1 Porosity

The reservoir fluids are trapped inside the pores of a porous medium, e.g. chalk or sandstone formations. The porosity is the volumetric fraction of the total bulk volume that is not occupied by solid matter (reservoir rock), ie. it is the fraction of the total bulk volume that may contain the reservoir fluids (water, oil). Let V be the total bulk volume of the reservoir and let V_p be the pore volume (PV), i.e. void space of the reservoir rock, then the porosity is defined as $\phi = V_p/V$. Since changes in rock characteristica are not included in our model, the porosity is constant over time but may vary at different locations in the reservoir, such that $\phi = \phi(\mathbf{s})$.

2.3.2 Permeability

The permeability is a measure of the capacity of a porous medium to conduct fluids through its interconnected pores. It is defined for single-phase flow and often referred to as *absolute permeability*. The concept of permeability is of importance in determining the connectivity and preferred flow direction in the reservoir. The permeability is defined as a tensor $\mathbf{k}$ of size 3×3. In theory $\mathbf{k}$ is a full tensor, however, in many practical situations it is possible to assume that $\mathbf{k}$ is a diagonal tensor given by

$$\mathbf{k} = \begin{bmatrix} k_{xx} & 0 & 0 \\ 0 & k_{yy} & 0 \\ 0 & 0 & k_{zz} \end{bmatrix} \tag{2.7}$$

Classification	Permeability [mD]
Poor to fair	1-15
Moderate	15-20
Good	20-250
Very good	250-1000
Excellent	>1000

Table 2.1: Typical values of absolute permeabilities [34].

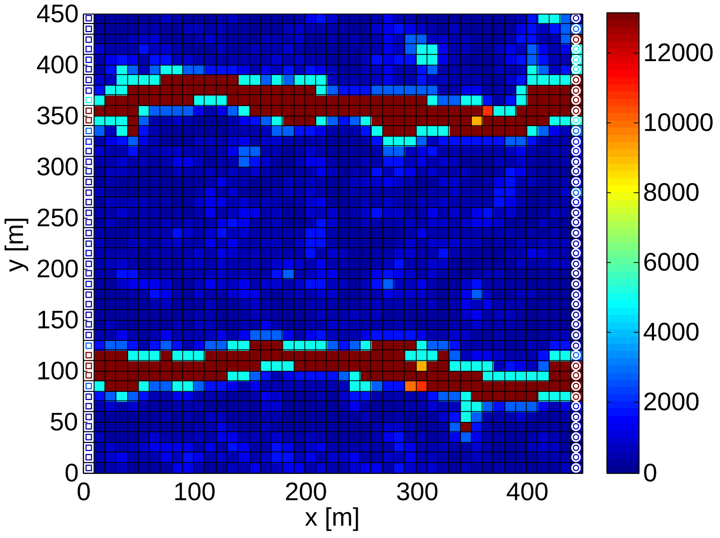


Figure 2.2: Permeability field [mD] with two high permeable streaks [1]. The white squares on the left-hand side illustrates a row of injectors, and the white circles on the right-hand side illustrates a row of producers.

in which all cross terms $k_{xy} = k_{xz} = k_{yx} = k_{yz} = k_{zx} = k_{zy} = 0$. Typically, in reservoirs formed by deposition, the directional trend in the horizontal plane is not very distinct or even not apparent, and we may assume $k_{xx} = k_{yy} = k_H$ to be the horizontal permeability. In such reservoirs the vertical permebaility $k_{zz} = k_V$ usually differs from k_H, and the horizontal permeability tends to be much larger than the vertical. In situations where $\mathbf{k} = k\mathbf{I}^{3\times 3}$ $(k_{xx} = k_{yy} = k_{zz} = k)$, the porous medium is called *isotropic*, otherwise it is called *anisotropic*. We limit ourselves to a porous medium with an isotropic heterogeneous permeability field. In our model we do not include temporal changes of properties related to the reservoir rock. Heterogeneity though implies a spatial dependency of the permeability, $\mathbf{k} = \mathbf{k}(\mathbf{s})$. The SI unit for permeability is [m^2]. However, in the petroleum literature it is common to use the unit *milliDarcy* [mD], where 1 mD $= 0.9869 \cdot 10^{-15}$ m^2 ($1.01325 \cdot 10^{12}$ mD $=$ 1 m^2). A reservoir is considered to be exploitable without stimulation (with e.g. surfactants), if the permeability of the reservoir rock is greater than approximately 100 mD. Rocks with permeability values significantly lower than this may form effecient seals, preventing the reservoir in exchanging fluids with the surrounding environment. Typical values of absolute permeabilities are given in Table 2.1. The permeability field that we use for production optimization in Chapter 5 is depicted in Figure 2.2, it is a synthetic field constructed by [1] for testing of optimization strategies. It has since been used as a benchmark problem in various articles [13, 24, 36, 19, 37] dealing with production optimization of oil reservoirs.

2.3.3 Relative Permeability

The relative permeability is a measure of the capacity of a porous medium to conduct a fluid through its interconnected pores in the presence of other fluids. It is defined as the ratio of the absolute permeability assigned to a particular fluid. The relative permeability $k_{r\alpha}$ is a dimensionless term. $k_{r\alpha} \leq 1$, assuming that the flow of each phase is inhibited by the presence of other phases. Thus, the effective permeability $\mathbf{k}_\alpha = \mathbf{k}k_{r\alpha}$ of a particular fluid is less than or equal to the single-phase permeability (absolute permeability) of the porous medium. We accept, as an empirical fact, that relative permeabilities are dependent on saturations alone, $k_{r\alpha} = k_{r\alpha}(S_\alpha)$. Figure 2.3 shows typical relative permeabilities for an oil-water system with water displacing oil. The value of S_w at which water starts to flow is called the critical water saturation, S_{wc}. The value of S_o at which oil (the displaced

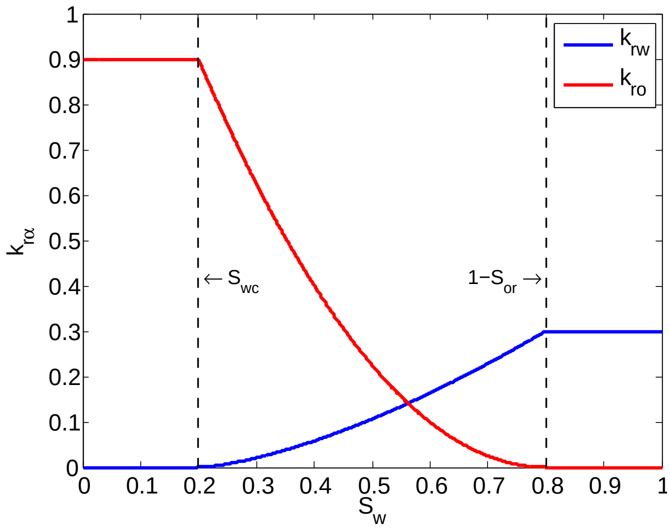


Figure 2.3: Corey's two-phase relative permeability model (2.8) for the values in Table 2.2.

phase) ceases to flow is called the residual oil saturation, S_{or}. S_{wc} and S_{or} are irreducible saturations that define lower bounds for the saturation values of water and oil, respectively. Consequently, the maximum oil saturation is $1 - S_{wc}$, and the maximum water saturation is $1 - S_{or}$. We use Corey's two-phase relative permeability model [38, 31, 34]

$$k_{rw} = \begin{cases} 0 & 0 \leq S_w \leq S_{wc} \\ k_{rw}^{\circ} s_w^{n_w} & S_{wc} < S_w < 1 - S_{or} \\ k_{rw}^{\circ} & 1 - S_{or} \leq S_w \leq 1 \end{cases} \tag{2.8a}$$

$$k_{ro} = \begin{cases} 0 & 0 \leq S_o \leq S_{or} \\ k_{ro}^{\circ} s_o^{n_o} & S_{or} < S_o < 1 - S_{wc} \\ k_{ro}^{\circ} & 1 - S_{wc} \leq S_o \leq 1 \end{cases} \tag{2.8b}$$

in which k_{rw}° and k_{ro}° are the relative permeability end points, n_w and n_o

Symbol	Description	Value	Unit
S_{wc}	Critical water saturation	0.2	-
S_{or}	Residual oil saturation	0.2	-
k°_{rw}	Water end point relative permeability	0.3	-
k°_{ro}	Oil end point relative permeability	0.9	-
n_w	Water Corey exponent	1.5	-
n_o	Oil Corey exponent	2.0	-

Table 2.2: The values used for illustration of the Corey's two-phase relative permeability model (2.8) in Figure 2.3.

are the Corey exponents and s_w and s_o are the normalized saturations

$$s_w = \frac{S_w - S_{wc}}{1 - S_{wc} - S_{or}} \tag{2.9a}$$
$$s_o = \frac{S_o - S_{or}}{1 - S_{wc} - S_{or}} \tag{2.9b}$$

of the water phase and the oil phase. Figure 2.3 is constructed using the parameters in Table 2.2.

2.3.4 Capillary Pressure

In immiscible oil-water systems, water is most often wetting the surface of the reservoir rock, meaning that water tends to maintain contact with the rock, thus displacing the oil. Due to interfacial tension between the non-wetting and the wetting phase fluids, the pressure in the non-wetting fluid is higher than the pressure in the wetting fluid. The difference between these two pessures is the capillary pressure

$$P_{cow} = P_o - P_w \tag{2.10}$$

Empirically, it is accepted, that the capillary pressure depends on the saturation of the wetting fluid, such that $P_{cow} = P_{cow}(S_w)$. Effects due to capillarity becomes less significant in highly permeable and highly porous media. In dense formations with very small pores, capillarity introduces a diffusive term into (2.1) [39]. However, we assume zero capillary pressure

$$P_{cow} = 0 \tag{2.11}$$

since no additional insight for the optimization problem is gained by including capillarity.

2.3.5 Density

Depending upon how fluids respond to pressure, they can be classified as compressible, slightly compressible or incompressible. Constant water density is normally a valid assumption. Some oil components may exhibit significant density changes with pressure, especially if the oil phase contains large quantities of dissolved gas. For a gas phase compressibility is very important. However, we consider a water/oil system, in which we assume both the water phase and the oil phase to behave like slightly compressible fluids at reservoir conditions. For isothermal conditions the compressibility of a fluid is defined by

$$c_\alpha = \frac{1}{\rho_\alpha} \frac{\partial \rho_\alpha}{\partial P_\alpha}\bigg|_T \tag{2.12}$$

where $\rho_\alpha = \rho_\alpha(P_\alpha)$ is the fluid density (at constant temperature T). Assuming the fluid compressibility to be constant over the pressure range of interest, integration of (2.12) yields the following equation of state

$$\rho_\alpha = \rho_\alpha^\circ e^{c_\alpha (P_\alpha - P_\alpha^\circ)} \tag{2.13}$$

where ρ_α° is the density at reference pressure P_α°. The relation (2.13) between density and pressure is depicted in Figure 2.4 using the reference values in Table 2.3. Using a Taylor series expansion, we see that

$$\rho_\alpha = \rho_\alpha^\circ \left[1 + c_\alpha (P_\alpha - P_\alpha^\circ) + \frac{1}{2!} c_\alpha^2 (P_\alpha - P_\alpha^\circ)^2 + \cdots \right] \tag{2.14}$$

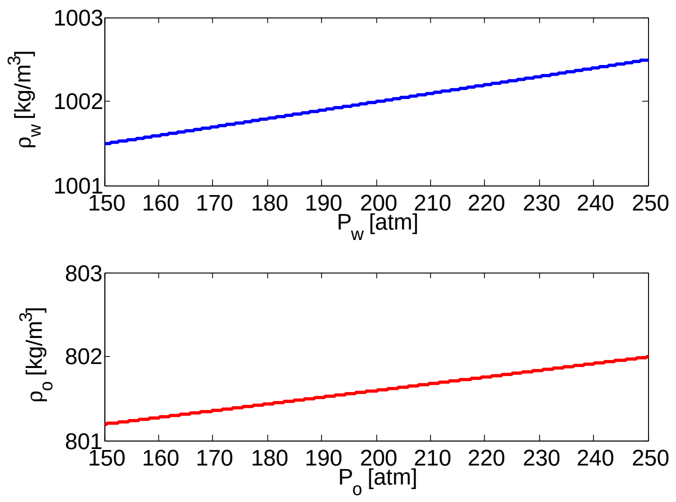


Figure 2.4: The fluid density (2.13) for the values in Table 2.3.

Symbol	Description	Value	Unit
ρ_w°	Water density (at 1 atm)	1000	kg·m^{-3}
ρ_o°	Oil density (at 1 atm)	800	kg·m^{-3}
c_w	Water compressibility	10^{-5}	atm^{-1}
c_o	Oil compressibility	10^{-5}	atm^{-1}

Table 2.3: The values used for illustration of the fluid density (2.13) in Figure 2.4.

and since we assume the fluids to be only slightly compressible, i.e. the compressibility c_α is small (typically in the range from 10^{-10} to 10^{-9} Pa^{-1} [34]), we can neglect all high order terms and approximate (2.13) with the linear relationship

$$\rho_\alpha \approx \rho_\alpha^\circ[1 + c_\alpha(P_\alpha - P_\alpha^\circ)] \tag{2.15}$$

using a reference density and pressure at reservoir conditions.

Classification	Viscosity [cP]
Light oil	0.3-1
Medium oil	1-6
Moderate oil	6-50
Heavy oil	50-1000
Heavy oil and oil sands	>1000

Table 2.4: Typical values of oil viscosities at reservoir conditions (28-41 MPa and 93° C) [34].

Symbol	Description	Value	Unit
μ_w	Water viscosity	1.0	cP
μ_o	Oil viscosity	1.0	cP

Table 2.5: Constant water and oil viscosities that we use in our model.

2.3.6 Viscosity

The viscosity (thickness) of a fluid is a measure of the internal friction that arises due to shear and tensile stress, when it is in motion. In general, fluid viscosity depends on phase composition (multi component phases), pressure (at very high pressures) and temperature. But, since the two-phase flow model is not compositional, and the pressure does not vary significantly throughout the reservoir, and the model is isothermal, we do not provide any of these dependencies. Instead, the viscosities μ_α of the water phase and the oil phase are held constant. The SI unit for visosity is [Pa·s], however, in the petroleum literature it is common to use the unit *centipoise* [cP], where 1 cP $= 10^{-3}$ Pa·s. The viscosity of water at standard conditions (1013.25 hPa at 20° C) is 1 cP. Typical values of oil viscosities at reservoir conditions (28-41 MPa at 93° C) are given in Table 2.4. In our model we use the values in Table 2.5.

2.4 Transport Model

The flow of the reservoir fluids is driven by spatial differences in pressure (pressure gradient) and gravity. Fluid flow in a porous medium at low to moderate velocities is governed by Darcy's law [40, 31]

$$\mathbf{u}_\alpha = -\lambda_\alpha \mathbf{k} \nabla \Phi_\alpha \tag{2.16}$$

in which $\lambda_\alpha = \frac{k_{r\alpha}}{\mu_\alpha}$ is the phase mobility. Darcy's law is a linear relationship between the phase velocity $\mathbf{u}_\alpha = \mathbf{u}_\alpha(P_\alpha, S_\alpha)$ and the flow potential field

$$\nabla \Phi_\alpha = \nabla P_\alpha - \rho_\alpha g \nabla z \tag{2.17}$$

where ∇P_α is the pressure gradient, g is the gravitational acceleration and z is the depth (downward positive). We do not include gravitational effects in the model, that is $g = 0$ in (2.17).

2.5 Well Models

Although we have not presented any method for spatial discretization yet, we will use the concept of grid blocks in this section. A grid block is a finite control volume of the total bulk volume of the reservoir. We will provide a detailed description of the finite volume approach in Chapter 3. Injection wells and production wells (injectors and producers) are located and perforated in a single grid block, e.g. as illustrated in Figure 2.5. Injectors are used for injection of the phase, which in our case is water, displacing the reservoir fluids. Producers are used for prodution of the displaced reservoir fluids. Let $\mathcal{N}$ be the set of grid blocks, $\mathcal{I} \subset \mathcal{N}$ be the set of grid blocks containing an injector, and $\mathcal{P} \subset \mathcal{N}$ be the set of grid blocks containing a producer. Thus, if $i \in \mathcal{I}$ then the i^{th} grid block is penetrated by an injection well, and if $i \in \mathcal{P}$ then it is penetrated by a production well. Injectors are operated at variable (volumetric) injection rates, whereas producers are operated at variable bottom hole pressures (BHPs). The BHP is the pressure inside the well at reservoir depth.

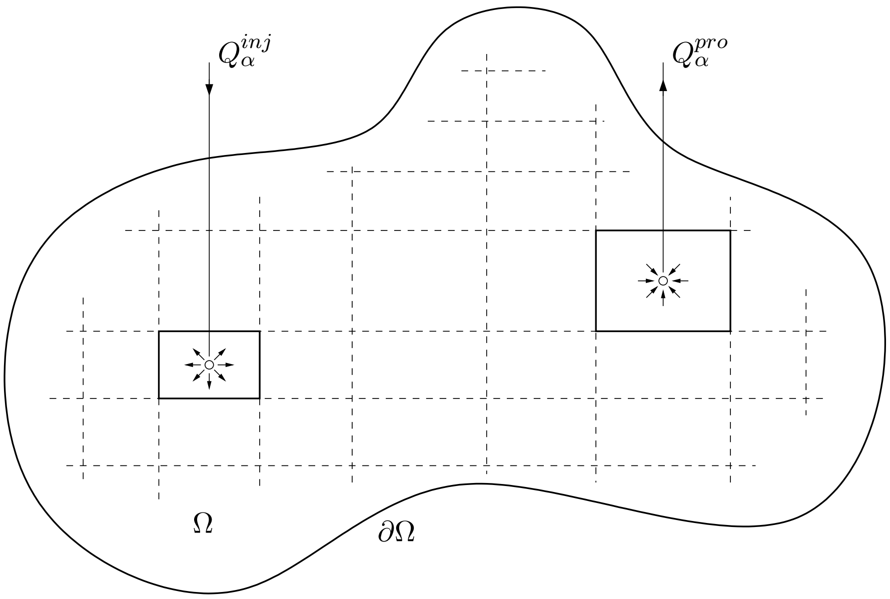


Figure 2.5: Example of an injector and a producer located in two different grid blocks.

Under the assumption of single-phase flow in the vicinity of the wells [31, 41], we consider wells that are vertical, and fully completed and centered in a single grid block. Furthermore, we assume that injectors and producers can not coexist in the same grid block, $i \neq j$, $i \in \mathcal{I}$, $j \in \mathcal{P}$. Other models could be implemented to simulate more specialized wells, e.g. horizontal or deviated wells, completion in multiple grid blocks, or wells in unstructured grids. However, such well models require a much more involved approach [42, 43, 44, 45, 46, 47, 48], which is outside the scope of this work.

2.5.1 Injection wells

If an injector penetrates grid block $i \in \mathcal{I}$, then we directly control the source terms $Q_{\alpha,i}^{inj} = Q_{\alpha,i}^{inj}(P_{\alpha,i})$. We define the source terms

$$Q^{inj}_{\alpha,i} = \left(\frac{q^{inj}_{\alpha}}{V}\right)_i \quad i \in \mathcal{I} \tag{2.18a}$$

as the rate of injected mass $q^{inj}_{\alpha,i}$ of each phase averaged over the grid block volume V_i (the rate of injected mass per unit volume of reservoir). Usually, only water is injected to keep the pressure in the reservoir above a certain level, implying that

$$q^{inj}_{w,i} = (\rho_w q^{inj})_i \quad i \in \mathcal{I} \tag{2.19a}$$

$$q^{inj}_{o,i} = 0 \quad i \in \mathcal{I} \tag{2.19b}$$

in which q^{inj}_i is the volumetric injection rate of water into grid block $i \in \mathcal{I}$.

2.5.2 Production wells

If a producer penetrates grid block $i \in \mathcal{P}$, then we can only indirectly control the sink terms $Q^{pro}_{\alpha,i} = Q^{pro}_{\alpha,i}(P_\alpha, S_\alpha)_i$, since the produced liquid is a composition of oil and water. We define the sink terms

$$Q^{pro}_{\alpha,i} = \left(\frac{q^{pro}_{\alpha}}{V}\right)_i \quad i \in \mathcal{P} \tag{2.20a}$$

as the rate of produced mass $q^{pro}_{\alpha,i}$ of each phase averaged over the grid block volume V_i (the rate of produced mass per unit volume of reservoir). The set of producers are modelled as

$$q^{pro}_{\alpha,i} = -(\mathrm{WI}\rho_\alpha\lambda_\alpha)_i(P_\alpha - P^{bh})_i \quad i \in \mathcal{P} \tag{2.21}$$

where WI_i denotes the well index, and P^{bh}_i denotes the BHP of the production well in grid block $i \in \mathcal{P}$. Each producer enters the reservoir model through a well index. The well index captures the interaction between the well and the reservoir, i.e. making the well model account for both geometric characteristics of the well and properties of the surrounding reservoir rock. Specifically, for each grid block $i \in \mathcal{P}$ containing a well, the quantity WI_i relates the flow rate of the well to the local BHP and the pressure in the grid block. For vertical wells in non-square Cartesian grids (cuboid grids) with anisotropic permeability fields (assuming diagonal tensors), Peaceman [35] derived an analytical expression for WI_i. The Peaceman well index is given as

$$\mathrm{WI}_i = \left(\frac{\theta \sqrt{k_{xx} k_{yy}} h}{\ln(r_e/r_w) + s} \right)_i \qquad i \in \mathcal{P} \tag{2.22}$$

θ is the angle open to flow (e.g. 2π for a well in the interior and $\frac{\pi}{2}$ for a corner well), k_{xx} and k_{yy} are permeability components in the x- and y-directions as defined in (2.7), Δx and Δy are the grid block sizes, h is the height of the well (grid block height in the z-direction), r_w is the wellbore radius, and

$$r_{e,i} = 0.28 \left(\frac{\sqrt{\sqrt{k_{yy}/k_{xx}} \Delta x^2 + \sqrt{k_{xx}/k_{yy}} \Delta y^2}}{\sqrt[4]{k_{yy}/k_{xx}} + \sqrt[4]{k_{xx}/k_{yy}}} \right)_i \qquad i \in \mathcal{P} \tag{2.23}$$

is the equivalent radius. r_e is the radial position (centered around the well) at which the pressure in the well block, computed by the simulator, is equal to the pressure obtained by the analytical model. The skin s in (2.22) is a dimensionless factor included to match the theoretical productivity of the well with actual conditions. Thus, the skin factor accounts for damage or influences that are impairing the well productivity, or stimulation (fracturing, acidization, etc.) that enhances productivity. We assume ideal conditions, that is $s = 0$.

2.6 State Transformation

Considering the algebraic constraints (2.5) and (2.11), we define

$$S = S_w = 1 - S_o \tag{2.24a}$$

$$P = P_o = P_w \tag{2.24b}$$

where $S = S(t, \mathbf{s})$ is the saturation of water, and $P = P(t, \mathbf{s})$ is the phase pressure (reservoir pressure, since $P_o = P_w$). Throughout the rest of this thesis we will refer to S and P as saturation and pressure respectively. In the two-phase flow model (2.1) - (2.17), we may use (S, P) as state variables instead of (C_w, C_o). (2.4), (2.5), (2.10) and (2.11) may be used to compute S and P given C_w and C_o. Implying that we may state the initial conditions (2.2) as initial saturation and pressure

$$S(t, \mathbf{s}) = S(\mathbf{s}) \quad t \in \delta T \quad \mathbf{s} \in \Omega \tag{2.25a}$$

$$P(t, \mathbf{s}) = P(\mathbf{s}) \quad t \in \delta T \quad \mathbf{s} \in \Omega \tag{2.25b}$$

2.7 Summary

We have described the two-phase flow problem, the well models, and the primary variables of the model. The two phases are oil and water. The model consists of two partial differential equations representing conservation of mass. Mass is transported by convection at a velocity determined by Darcy's law. Relative permeabilites are determined by a Corey expression and we assume zero capillary pressure. The fluid densities are described by an equation of state relating the densities to pressure. The porosity is homogeneous and we assume a heterogenous isotropic permeability field. We use vertical injection and production wells of the Peaceman type. The primary variables of the model are reservoir pressure and water saturation.

CHAPTER 3

Spatial Discretization

In this chapter, we describe how we obtain the discrete representations for the spatial derivative operators in the two-phase flow equations (2.1). For the purpose of spatial discretization we have a variety of methods to choose from. Most of the methods are variations and combinations of three well known methods: the finite difference method (FDM), the finite volume method (FVM), and the finite element method (FEM).
The FDM is considered the oldest of the three methods, and because of its simplicity and intuitive approach it is still widely used [49, 34]. The method represents the spatial derivatives in discrete grid points. FD schemes are sufficiently robust and efficient for a large number of problems, and extensions to higher order approximations of the solution is relatively straightforward to obtain. However, higher order finite difference stencils are often constructed locally for each spatial dimension. This limits the geometric flexibility of the method, making it less suitable for handling domains with complex geometry, e.g. in terms of local grid refinements for dealing with abrupt changes in absolute permeability.
The FVM is a control volume formulation that uses an element based discretization. Due to the control volume formulation, FVMs maintain local

conservation of mass, energy and momentum, which cannot in general be guaranteed for FDMs. The FVM represents the physical domain by a collection of small control volumes that jointly fill the entire domain. The elements may have different sizes and may be organized in an unstructured manner. The solution is approximated on the element by the cell average at the center of the element. The classic finite volume scheme is purely local, thus, no limitations are imposed on the grid structure, and the method generalizes easily to unstructured grids in higher dimensions. This ensures geometric flexibility of the method. The FVM reduces the flux term to a surface integral by the use of Gauss' divergence theorem, and therefore we must evaluate the fluxes at the boundaries. The interface flux between neighbouring elements is most often computed by the two point flux approximation (TPFA) [32, 65, 50]. The TPFA is a low order approximation. High order approximations are not straightforward to obtain in unstructured grids, which is one of the major drawbacks of the FVM. High order reconstructions of interface values by the use of multi point flux approximations (MPFA) involve information from more than two cells. This introduces the need for particular grid structures, which jeopardizes the geometric flexibility of the FVM in higher dimensions. In particular, for porous media flow in an anisotropic permebaility field on non-**k**-orthogonal grids, TPFA gives an error in the solution [51, 52]. However, to be able to solve the flow equations on general grids, variations of both TPFA [53] and MPFA [54, 55, 56, 57, 58] methods have been suggested.
The FEM has initially been developed for structural stress analysis. As the name of the method suggests, it is element based. It ensures geometric flexibility and allow different element sizes. A recent employment of the FEM in reservoir simulation can be found in [59]. High order discrete approximations of the solution are relatively simple to obtain in the finite element setting. In particular, local basis functions allows for different orders of approximation in each element. However, the basis functions are symmetric in space, and this may cause stability issues for problems based on conservation laws, in which information flows in specific directions. These issues may be solved by the discontinuous Galerkin finite element method (DG-FEM). The DG-FEM is basically a hybrid between the FEM and FVM.

The method ensures flexibility both in terms geometry and in the choice of the numerical flux, high order approximations are relatively simple to obtain on general grids, and it maintains local conservation. On the negative side, the number of unknowns increases and DG-FEM solvers can be computationally expensive in comparison to FVM and FEM solvers. However, because of the appealing properties, the method is gaining interest in reservoir simulation [60, 61, 62] and a general description of the DG-FEM is found in [63].
Because of the close relation to the conservation laws, we have chosen the FVM for spatial discretization. We assume an isotropic permeability field, thus, we use the TPFA to reconstruct the discrete flux terms. In Section 3.1 and 3.2 we describe how we derive the spatially dicretized flow equations. Section 3.3 discusses how to evaluate the transmissibilities. In Section 3.4 we present a new differential equation model for the spatially dicretized flow equations, as proposed in Paper A. Finally, in the last section we illustrate the Jacobian structure of the spatially discretized model.

3.1 Nomenclature

For the finite volume approach, we divide the spatial domain Ω into N subregions Ω_i with boundaries $\partial\Omega_i$, $i \in \mathcal{N} = \{1, 2, \dots, N\}$, such that

$$\bigcup_{i\in\mathcal{N}} \Omega_i = \Omega \tag{3.1a}$$

$$(\Omega_i \backslash \partial\Omega_i) \cap (\Omega_j \backslash \partial\Omega_j) = \emptyset \quad i,j \in \mathcal{N} \quad i \neq j \tag{3.1b}$$

where $\mathcal{N}$ is the set of indices of non-overlapping control volumes that cover the whole domain Ω. This is illustrated in Figure 3.1. In the following we will use Figure 3.1 to explain the problem setup and the symbols used. Define

$$\gamma_{ij} = \Omega_i \cap \Omega_j \quad i,j \in \mathcal{N} \quad i \neq j \tag{3.2}$$

and for the i^{th} control volume Ω_i, let $\mathcal{N}^{(i)} \subset \mathcal{N}$ denote the set of indices of neighbouring subregions. Then

$$\gamma_{ij} = \Omega_i \cap \Omega_j \quad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.3}$$

denotes the interface between two adjacent control volumes. In particular

$$\gamma_{ij} \neq \emptyset \quad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.4a}$$

$$\gamma_{ij} = \emptyset \quad i \in \mathcal{N} \quad j \notin \mathcal{N}^{(i)} \tag{3.4b}$$

Each control volume contains a nodal point. Properties of the model are represented at the nodal points as the average over the control volumes surrounding them. The nodal point of the i^{th} control volume is located such that

$$\mathbf{s}_i \in \Omega_i \quad i \in \mathcal{N} \tag{3.5}$$

and we define

$$\mathbf{s}_{ij} = \mathbf{s}_j - \mathbf{s}_i \quad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.6}$$

as the internode connection between the grid node locations of two neighbouring grid blocks i and j. For the discrete spatial domain we assume that

$$\mathbf{s}_{ij} \perp \gamma_{ij} \quad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.7}$$

by construction. Figure 3.1 illustrates a 2-D grid maintaining the property in (3.7). Because the location of the grid nodes $\mathbf{s}_i$ and $\mathbf{s}_j$ are taken as the circumcenters of each block, we see that $\mathbf{s}_{ij}$ (perpendicular) bisects the interface γ_{ij}, in this particular example.
For the spatial discretization below we use the following notation: ψ_{ij} means that we evaluate the property ψ at the interface γ_{ij}, and ψ_i means that we evaluate the property ψ at the nodal point $\mathbf{s}_i$. In particular, for properties depending on both time and space we have $\psi_i = \psi(t, \mathbf{s}_i)$, and for time invariant properties that only depend on space we have $\psi_i = \psi(\mathbf{s}_i)$.

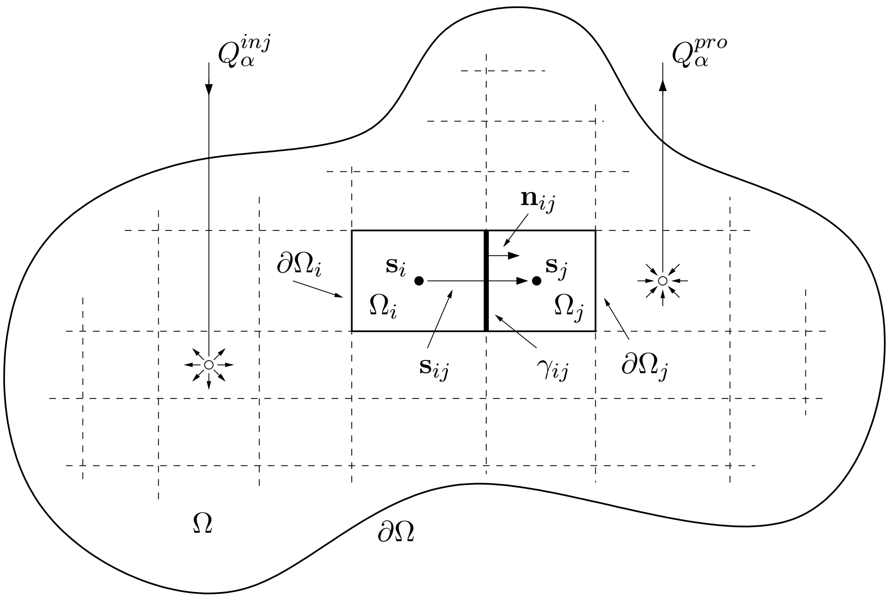


Figure 3.1: A reservoir spatially discretized by the FVM.

3.2 Finite Volume Approach

By integration of (2.1) over the control volume Ω_i we obtain the integral form of the governing equations

$$\frac{\partial}{\partial t}\int_{\Omega_i} C_\alpha dV = -\int_{\Omega_i} \nabla\cdot\mathbf{F}_\alpha dV + \int_{\Omega_i} Q_\alpha dV \quad i\in\mathcal{N} \tag{3.8}$$

The volume integral on the left-hand side in (3.8), the accumulation term, is discretized using the average value of the accumulated mass C_α over the control volume

$$\int_{\Omega_i} C_\alpha dV \approx (C_\alpha V)_i \quad i\in\mathcal{N} \tag{3.9}$$

The volume integral in the second term on the right-hand side in (3.8), the source term, is discretized using the average value of the source Q_α over the control volume

$$\int_{\Omega_i} Q_\alpha dV \approx (Q_\alpha V)_i \quad i \in \mathcal{N} \tag{3.10}$$

In particular, $Q_{\alpha,i} = 0$, $i \notin \mathcal{I} \cup \mathcal{P}$. Remember that $\mathcal{I}$ is defined in Section 2.5 as the set of indices of grid blocks containing an injector, and $\mathcal{P}$ is defined as the set of indices of grid blocks containing a producer.
The volume integral in the first term on the right-hand side in (3.8), the convective (flux) term, is rewritten as an integral over the entire bounding surface of the control volume by application of Gauss' divergence theorem

$$\int_{\Omega_i} \nabla \cdot \mathbf{F}_\alpha dV = \int_{\partial\Omega_i} \mathbf{n} \cdot \mathbf{F}_\alpha dA \quad i \in \mathcal{N} \tag{3.11}$$

where $\mathbf{n}$ is a unit vector normal to the surface elements dA of control volume Ω_i. Using (3.4), we may rewrite (3.11) as follows

$$\int_{\Omega_i} \nabla \cdot \mathbf{F}_\alpha dV = \sum_{j \in \mathcal{N}^{(i)}} \int_{\gamma_{ij}} \mathbf{n} \cdot \mathbf{F}_\alpha dA \quad i \in \mathcal{N} \tag{3.12}$$

in which we sum over interfaces between the i^{th} control volume and its neighbouring subregions Ω_j, $j \in \mathcal{N}^{(i)}$. We approximate the surface integrals in (3.12) using the midpoint rule

$$\int_{\Omega_i} \nabla \cdot \mathbf{F}_\alpha dV \approx \sum_{j \in \mathcal{N}^{(i)}} (\mathbf{n} \cdot \mathbf{F}_\alpha A)_{ij} \quad i \in \mathcal{N} \tag{3.13}$$

where $\mathbf{n}_{ij}$ is the normal vector of interface γ_{ij} (pointing outwards in relation to Ω_i), $\mathbf{F}_{\alpha,ij}$ is the flux across interface γ_{ij}, and A_{ij} is the area of interface γ_{ij}. We now evaluate the fluxes across the interfaces using the TPFA. Remembering (2.6), (2.16), (2.17), and neglecting gravity, the flux across interface γ_{ij} becomes

$$\mathbf{F}_{\alpha,ij} = -(\rho_\alpha \lambda_\alpha \mathbf{k} \nabla P)_{ij} \quad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.14}$$

where the properties $\rho_{\alpha,ij}$, $\lambda_{\alpha,ij}$, $\mathbf{k}_{ij}$, and the pressure gradient ∇P_{ij} are evaluated at interface γ_{ij}. Substituting the flux in (3.13) with (3.14) yields

$$\begin{aligned}\int_{\Omega_i} \nabla \cdot \mathbf{F}_\alpha dV &\approx - \sum_{j\in\mathcal{N}^{(i)}} \mathbf{n}_{ij} \cdot (\rho_\alpha \lambda_\alpha \mathbf{k} \nabla P)_{ij} A_{ij} \\ &= - \sum_{j\in\mathcal{N}^{(i)}} A_{ij} (\rho_\alpha \lambda_\alpha)_{ij} \mathbf{n}_{ij} \cdot (\mathbf{k}\nabla P)_{ij}\end{aligned} \qquad i \in \mathcal{N} \tag{3.15}$$

We only consider isotropic permeability fields, $\mathbf{k}_{ij} = k_{ij}\mathbf{I}$. Consequently

$$\int_{\Omega_i} \nabla \cdot \mathbf{F}_\alpha dV \approx - \sum_{j\in\mathcal{N}^{(i)}} (Ak)_{ij} (\rho_\alpha \lambda_\alpha)_{ij} \mathbf{n}_{ij} \cdot \nabla P_{ij} \quad i \in \mathcal{N} \tag{3.16}$$

The flux over interface γ_{ij} is driven by the pressure gradient ∇P_{ij}. For the TPFA we use a first order approximation of the pressure gradient

$$\nabla P_{ij} \approx \left(\frac{\Delta P}{\Delta s} \frac{\mathbf{s}}{\Delta s} \right)_{ij} \qquad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.17}$$

where $\Delta P_{ij} = P_j - P_i$ is the pressure difference between the nodal points $\mathbf{s}_i$ and $\mathbf{s}_j$ of two adjacent control volumes Ω_i and Ω_j. We define $\mathbf{s}_{ij} = \mathbf{s}_j - \mathbf{s}_i$, such that $\Delta s_{ij} = |\mathbf{s}_{ij}|$ is the internode distance between $\mathbf{s}_i$ and $\mathbf{s}_j$. Substitution of ∇P_{ij} in (3.16) with (3.17) yields

$$\int_{\Omega_i} \nabla \cdot \mathbf{F}_\alpha dV \approx - \sum_{j\in\mathcal{N}^{(i)}} (Ak)_{ij} (\rho_\alpha \lambda_\alpha)_{ij} \mathbf{n}_{ij} \cdot \left(\frac{\Delta P}{\Delta s} \frac{\mathbf{s}}{\Delta s} \right)_{ij} \quad i \in \mathcal{N} \tag{3.18}$$

which we rearrange

$$\int_{\Omega_i} \nabla \cdot \mathbf{F}_\alpha dV \approx - \sum_{j\in\mathcal{N}^{(i)}} \left(\frac{Ak}{\Delta s} \right)_{ij} (\rho_\alpha \lambda_\alpha)_{ij} \Delta P_{ij} \mathbf{n}_{ij} \cdot \left(\frac{\mathbf{s}}{\Delta s} \right)_{ij} \quad i \in \mathcal{N} \tag{3.19}$$

Because of the relation in (3.7), the grid is constructed such that

$$\mathbf{n}_{ij} = \left(\frac{\mathbf{s}}{\Delta s}\right)_{ij} \quad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.20}$$

and $|\mathbf{n}_{ij}| = 1$, so that finally the discrete convective term becomes

$$\int_{\Omega_i} \nabla \cdot \mathbf{F}_\alpha dV \approx -\sum_{j\in\mathcal{N}^{(i)}} \left(\frac{Ak}{\Delta s}\right)_{ij} (\rho_\alpha \lambda_\alpha)_{ij} \Delta P_{ij} \quad i \in \mathcal{N} \tag{3.21}$$

It is common in the petroleum literature to define the *transmissibility*

$$\Upsilon_{\alpha,ij} = \left(\frac{Ak}{\Delta s}\right)_{ij} (\rho_\alpha \lambda_\alpha)_{ij} \quad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.22}$$

allowing us to write (3.21) in the compact form

$$\int_{\Omega_i} \nabla \cdot \mathbf{F}_\alpha dV \approx -\sum_{j\in\mathcal{N}^{(i)}} (\Upsilon_\alpha \Delta P)_{ij} \quad i \in \mathcal{N} \tag{3.23}$$

A detailed representation of the transmissibility is considered in Section 3.3. We now substitute the accumulation term, the source term and the convective term in (3.8) with (3.9), (3.10) and (3.23) respectively. Consequently, we may express the model (3.8) as a system of differential equations in the form

$$\frac{dC_{\alpha,i}}{dt} V_i = \sum_{j\in\mathcal{N}^{(i)}} (\Upsilon_\alpha \Delta P)_{ij} + (Q_\alpha V)_i \quad i \in \mathcal{N} \tag{3.24}$$

The left-hand side in this system of differential equations is not trivial. It is a function (2.4) of the primary variables that we want to compute, i.e. pressure and saturation. Before we can simulate and predict the field development using (3.24), we must choose a method for temporal discretization that maintains the mass preserving property of the spatially discrete equations. Consequently, we have to take the special structure of (3.24) into consideration when choosing a numerical scheme for integration. We will discuss the reasons for that and the structure of (3.24) in more detail in Section 3.4, and we will suggest a new formulation of systems of differential equations for process simulation problems that are based on conservation of mass, energy, and momentum.

3.3 Calculation of Transmissibility

In this section we describe how to compute the transmissibility (3.22) for grid blocks of unequal size and varying permeability. Transmissibility is formed as the product of two parts. The first part, the *geometric* part, contains the effects of absolute permeability and grid geometry. The second part, the *fluid* part, depends purely on fluid properties. These parts are given seperate designations since they are treated differently.

3.3.1 Treatment of the Geometric Part

We designate the geometric part

$$\Gamma_{ij} = \left(\frac{Ak}{\Delta s}\right)_{ij} \qquad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.25}$$

When the reservoir is discretized spatially, the permeability is approximated by a piecewise constant function, where k_i is the average grid block permeability of each control volume Ω_i. Consider an irregular grid with blocks of unequal size. We focus on two adjacent grid blocks $i \in \mathcal{N}$ and $j \in \mathcal{N}^{(i)}$, of permeabilities k_i and k_j. In general, grid geometry such as the area A_{ij} of the interface and the internode distance Δs_{ij} are straightforward to compute. However, the value to use for the interface permeability k_{ij} in (3.25) is not obvious if k_i and k_j differ.
The flow between two adjacent grid blocks is expected to be predominated by the block with the lowest permeability. This suggests a harmonic averaging of the absolute permeabilities of neighbouring grid blocks along their interfaces [64, 34]. Imposing flux continuity across the grid block interfaces leads to

$$k_{ij} = \frac{\Delta s_{ij}}{\Delta s_i / k_i + \Delta s_j / k_j} \qquad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.26}$$

which corresponds to the *weighted harmonic average* of k_i and k_j. Δs_i is the distance between $\mathbf{s}_i$ and γ_{ij}, and Δs_j is the distance between $\mathbf{s}_j$ and γ_{ij}. In particular, $\Delta s_{ij} = \Delta s_i + \Delta s_j$. Note that the geometric part of the transmissibility can be computed in a preprocessing step for the simulation.

3.3.2 Treatment of the Fluid Part

The fluid part of the transmissibility is treated differently than the geometric part. The effects of transportiveness of the fluid flow must be taken into consideration [65]. The fluid part is denoted

$$\mathrm{H}_{\alpha,ij} = (\rho_\alpha \lambda_\alpha)_{ij} \quad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.27}$$

The hyperbolic character of the governing equations introduces a direction dependency into the system. Therefore, the fluid part of the transmissibility is *upwinded*. That is, the interface terms in (3.22) related to the fluids are evaluated using *upstream information*. Although many different schemes exist for upwind interpolation, we will consider the simplest of them - the single-point upstream (SPU) scheme, which is first order accurate. The main advantage of the SPU scheme is its simplicity and its strictly non-oscillatory behaviour. Higher order schemes may produce spurious oscillations in the solution near sharp fronts, unless sophisticated methods to dampen these oscillations are applied, e.g. such as flux limiters. The main drawback of the SPU scheme is the low order. A very fine grid is needed for producing an accurate solution using lower order schemes. For first order accurate methods, the second order term is the dominating part in an expansion of the error. Second order terms behave diffusion-like, and the truncation error introduced by first order schemes is often referred to as *numerical diffusion*. Thus, the diffusive nature of the error tends to smear out sharp fronts in the solution.
Different phases can flow in opposite directions at different speeds across the same interface. However, we assume purely advective flow in a porous medium, and we neglect the effects of gravity and capillary pressure. Under these assumptions we obtain *unidirectional flow*, where the phases always flow in the same direction across the same interface, even at different speeds.

Therefore, the direction of the phase velocity can be readily determined without explicitly computing the velocity by simply determining the sign of ΔP_{ij}, such that

$$\mathrm{H}_{\alpha,ij} = \begin{cases} (\rho_\alpha \lambda_\alpha)_i & \Delta P_{ij} < 0 \\ (\rho_\alpha \lambda_\alpha)_j & \Delta P_{ij} > 0 \end{cases} \quad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \tag{3.28}$$

While the geometric part of the transmissibility can be computed in a preprocessing step for the simulation, the fluid part is computed concurrently.

3.4 Differential Equation Model

We use an explicit singly diagonally implicit Runge-Kutta (ESDIRK) method for the solution of (3.24). ESDIRK methods have previously been developed for systems of ordinary differential equations (ODEs) [66, 67, 68, 69]

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}) \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.29}$$

and systems of index-1 differential algebraic equations (DAEs)

$$\mathbf{M}(\mathbf{x})\frac{d\mathbf{x}}{dt} = \mathbf{f}(t, \mathbf{x}) \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.30}$$

While the spatially discretized model for the two-phase flow problem may be formulated as (3.30), such a formulation is not guaranteed to preserve mass upon discretization in time. This is a major problem, as the differential equations in (3.24) were formulated based on conservation of mass. Process simulation problems in general are based on conservation of mass, energy, and momentum. It is desirable to preserve such properties upon numerical discretization in time. In Paper A we propose a new differential equation model that upon temporal discretization maintains these properties for process simulation problems. This model can be formulated as

$$\frac{d\mathbf{g}(\mathbf{x})}{dt} = \mathbf{f}(t, \mathbf{x}) \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.31}$$

in which $\mathbf{x} = \mathbf{x}(t)$ are the system states, $\mathbf{g}(\mathbf{x})$ are the properties conserved, while the right-hand side function $\mathbf{f}(t, \mathbf{x})$ has the usual interpretation. In general, problems based on conservation of mass, energy, and momentum, can directly be formulated as the model (3.31). Upon discretization in time this model preserves mass, energy, and momentum. This is not in general the case, if these problems are expressed as (3.30) using the chain rule, i.e. $\frac{d\mathbf{g}(\mathbf{x})}{dt} = \frac{d\mathbf{g}(\mathbf{x})}{d\mathbf{x}} \frac{d\mathbf{x}}{dt} = \mathbf{M}(\mathbf{x}) \frac{d\mathbf{x}}{dt}$ with $\mathbf{M}(\mathbf{x}) = \frac{d\mathbf{g}(\mathbf{x})}{d\mathbf{x}}$. In Section 4.1.3 and 4.1.4 we will formulate Runge-Kutta and ESDIRK methods for the purpose of solving differential equation models that has the structure of (3.31).
In the two-phase flow problem considered, $\mathbf{x}$ is a vector with pressure and water saturation in each grid block, $\mathbf{g}(\mathbf{x})$ is a vector with oil and water concentrations in each grid block, and $\mathbf{f}(t, \mathbf{x})$ is a vector with the fluxes of oil and water into each grid block plus any sources/sinks due to wells. Consequently, the spatially discretized model for two-phase flow (3.24) has the structure of (3.31).

3.5 Structure of the Jacobian Matrix

The application of ESDIRK methods for solution of the differential equation model (3.31) involves the Jacobian of the residual form of this model. In particular, for solution of the two-phase flow model considered using ESDIRK methods, we must derive the Jacobian of (3.24) with respect to pressure and saturation. Since each grid block is associated with two equations (one for oil and one for water) and two unknowns (pressure and saturation), the Jacobian matrix will be of size $2N \times 2N$. Because the flux term in (3.24) is dependent on pressure and saturation in several grid blocks, we will use the flux term to illustrate the non-zero structure of the Jacobian matrix. The net flux of oil and water into the i^{th} grid block is

$$F_{\alpha,i} = \sum_{j \in \mathcal{N}^{(i)}} (\Upsilon_\alpha \Delta P)_{ij} \quad i \in \mathcal{N} \tag{3.32}$$

Define the vector

$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T \tag{3.33}$$

in which

$$\mathbf{x}_i = [P_i, S_i]^T \quad i \in \mathcal{N} \tag{3.34}$$

Define the vector

$$\mathbf{F} = [\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_N]^T \tag{3.35}$$

with

$$\mathbf{F}_i = [F_{o,i}, F_{w,i}]^T \quad i \in \mathcal{N} \tag{3.36}$$

where $\mathbf{F} = \mathbf{F}(\mathbf{x})$ and $\mathbf{F}_i = \mathbf{F}_i(\mathbf{x})$. $\mathbf{x} = \mathbf{x}(t)$ is a vector with pressure and saturation in each grid block, and $\mathbf{F}$ is a vector with the net fluxes of oil and water into each grid block. Elements of the Jacobian are obtained by differentiating all elements of the vector (3.35) with respect to all the unknowns (3.33) in the discrete domain. Consequently, the Jacobian of the flux term

$$\frac{\partial \mathbf{F}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{F}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{F}_1}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{F}_1}{\partial \mathbf{x}_N} \\ \frac{\partial \mathbf{F}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{F}_2}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{F}_2}{\partial \mathbf{x}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{F}_N}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{F}_N}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{F}_N}{\partial \mathbf{x}_N} \end{bmatrix} \tag{3.37}$$

is a block matrix with $N \times N$ blocks. Each block in (3.37) is of size 2×2 and defined as follows

$$\frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_j} = \begin{bmatrix} \frac{\partial F_{o,i}}{\partial P_j} & \frac{\partial F_{o,i}}{\partial S_j} \\ \frac{\partial F_{w,i}}{\partial P_j} & \frac{\partial F_{w,i}}{\partial S_j} \end{bmatrix} \quad i, j \in \mathcal{N} \tag{3.38}$$

In particular

$$\frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_j} = \begin{cases} \bullet^{2\times 2} & j \in \{i\} \cup \mathcal{N}^{(i)} \\ \mathbf{0}^{2\times 2} & j \in \mathcal{N} \backslash (\{i\} \cup \mathcal{N}^{(i)}) \end{cases} \quad i \in \mathcal{N} \tag{3.39}$$

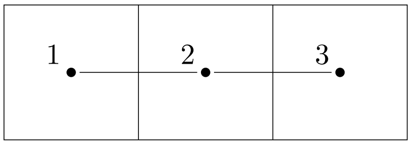


Figure 3.2: A 1-dimensional grid. The Jacobian is depicted in Figure 3.3.

$$\begin{array}{c|ccc|}
 & 1 & 2 & 3 \\
\hline
1 & \bullet & \bullet & \\
2 & \bullet & \bullet & \bullet \\
3 & & \bullet & \bullet \\
\hline
\end{array}$$

Figure 3.3: The Jacobian for the 1-dimensional grid in Figure 3.2.

where the bullet • denotes that (3.38) is a block containing one or more non-zero elements. Some elements in (3.38) may be zero due the upstream weighting of the fluid part (3.28) of the transmissibility defined in (3.22). Remember that $\mathcal{N}^{(i)}$ is defined in Section 3.1 as the set of indices of grid blocks adjacent to the i^{th} grid block. A thorough derivation of the elements in (3.38) can be found in Appendix A.
The structure of the Jacobian matrix depends on both grid structure and grid numbering, and in the following we will use three grids in 1-, 2- and 3-dimensions respectively for illustrating the non-zero structure of the Jacobian that is defined by (3.39). We use the bullet in (3.39) to illustrate non-zero blocks in the figures that illustrate the Jacobian structures. However, we must mention that the dots in the figures that illustrate the grids denote the nodal points of the control volumes. The 1-dimensional grid consists of 3 grid blocks, the 2-dimensional grid consists of 9 grid blocks, and the 3-dimensional grid consists of 18 grid blocks. The 2- and 3-dimensional grids are constructed using the 1-dimensional grid in Figure 3.2 as a basis. The structure of the Jacobian for the 1-dimensional grid is depicted in Figure 3.3.

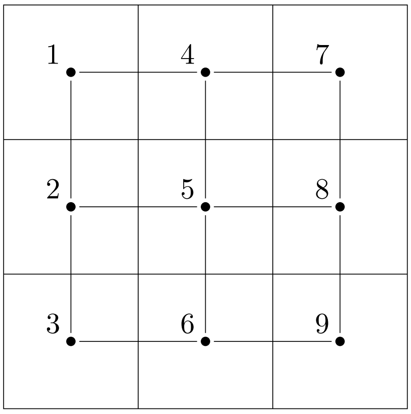


Figure 3.4: A 2-dimensional grid. The Jacobian is depicted in Figure 3.5.

	1	2	3	4	5	6	7	8	9
1	●	●		●					
2	●	●	●		●				
3		●	●			●			
4	●			●	●		●		
5		●		●	●	●		●	
6			●		●	●			●
7				●			●	●	
8					●		●	●	●
9						●		●	●

Figure 3.5: The Jacobian for 2-dimensional the grid in Figure 3.4.

We construct the 2-dimensional grid in Figure 3.4 by repeating the 1-dimensional grid three times along the second coordinate axis. We see that the 2-dimensional Jacobian in Figure 3.5 comprises three 1-dimensional Jacobians. Furthermore, two diagonals with index 3 and -3 are added (the

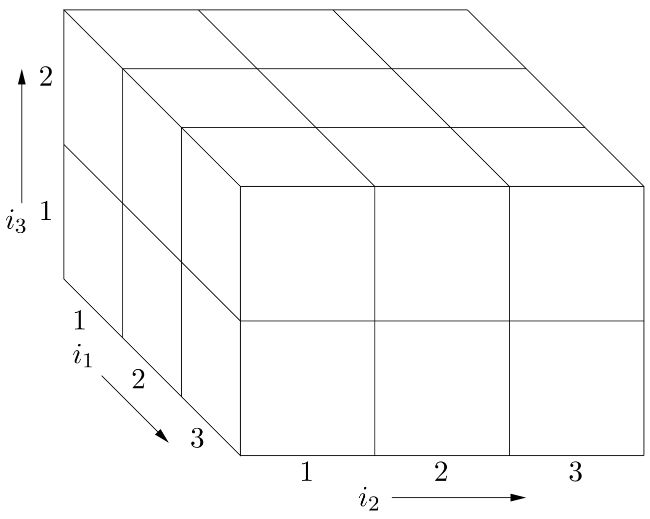


Figure 3.6: A 3-dimensional grid. The Jacobian is depicted in Figure 3.7.

main diagonal has index 0). These diagonals represent the new connections established by repeating the 1-dimensional grid. Let us use the grid block with index 2 in the 2-dimensional grid to illustrate the non-zero structure defined by (3.39): the grid is identified by the set $\mathcal{N} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and $\mathcal{N}^{(2)} = \{1, 3, 5\}$ is the set of grid blocks adjacent to the 2nd grid block. Implying that

$$\frac{\partial \mathbf{F}_2}{\partial \mathbf{x}_j} = \begin{cases} \bullet^{2\times 2} & j \in \{1, 2, 3, 5\} \\ \mathbf{0}^{2\times 2} & j \in \{4, 6, 7, 8, 9\} \end{cases}$$

is the non-zero structure that describes the connections associated with block number 2 in the 2-dimensional grid.

The 3-dimensional grid in Figure 3.6 is constructed by repeating the 2-dimensional grid two times along the third coordinate axis. We see that the 3-dimensional Jacobian in Figure 3.7 comprises two 2-dimensional Jacobians. In addition, two diagonals with index 9 and -9 are added. These diagonals represent the new connections established by the repetition of the

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	●	●		●						●								
2	●	●	●		●						●							
3		●	●			●						●						
4	●			●	●		●						●					
5		●		●	●	●		●						●				
6			●		●	●			●						●			
7				●			●	●								●		
8					●		●	●	●								●	
9						●		●	●									●
10	●									●	●		●					
11		●								●	●	●		●				
12			●								●	●			●			
13				●						●			●	●		●		
14					●						●		●	●	●		●	
15						●						●		●	●			●
16							●						●			●	●	
17								●						●		●	●	●
18									●						●		●	●

Figure 3.7: The Jacobian for the 3-dimensional grid in Figure 3.6.

2-dimensional grid. For the purpose of illustrating the 3-dimensional grid and appertaining Jacobian we have defined the following global grid block index

$$i = i_1 + (i_2 - 1)3 + (i_3 - 1)9 \quad i_1, i_2 = 1, 2, 3 \quad i_3 = 1, 2$$

In general we see that the bandwidth of the Jacobian matrix increases as the size of the grid and/or the dimension of the grid grows. This significantly increases the computational cost when solving the non-linear equation system.

The Jacobian matrices in Figure 3.3, 3.5 and 3.7 are not symmetric. This is because upwinding of the fluid part of the transmissibility introduces asymmetry in the blocks (3.38). In general, the SPU scheme generates asymmetric Jacobian matrices, which is a drawback in terms of solving the system of non-linear equations. However, assuming unidirectional flow and using a sequential method for temporal discretization (see Section 4.1), a reordering strategy can be applied [61]. The idea is to obtain a triangular structure of the Jacobian. This may reduce the computational cost of the solution procedure. This adds another advantage to the SPU scheme, besides its simplicity and its non-oscillating behavior mentioned above.

3.6 Building the Differential Equation Model and the Jacobian Matrix

To solve the spatially discretized two-phase flow model using ESDIRK methods, we must interpret (3.24) so that we can formulate the problem in the form of (3.31). Define the vector

$$\mathbf{C} = [\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_N]^T \tag{3.40}$$

with the accumulation terms in (3.24), in which

$$\mathbf{C}_i = [C_{o,i}, C_{w,i}]^T \quad i \in \mathcal{N} \tag{3.41}$$

where $\mathbf{C} = \mathbf{C}(\mathbf{x})$ and $\mathbf{C}_i = \mathbf{C}_i(\mathbf{x})$. Define the vector

$$\mathbf{Q} = [\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_N]^T \tag{3.42}$$

with the source terms in (3.24), where

$$\mathbf{Q}_i = [Q_{o,i}, Q_{w,i}]^T \quad i \in \mathcal{N} \tag{3.43}$$

in which we define $\mathbf{Q} = \mathbf{Q}(\mathbf{x})$ and $\mathbf{Q}_i = \mathbf{Q}_i(\mathbf{x})$. In Algorithm 3.6.1 we show how to build the differential equation model (3.31).
As mentioned in Section 3.5, ESDIRK methods require the Jacobian of the residual form of the differential equation model (3.31). Therefore, we must

Algorithm 3.6.1: Building the differential equation model (3.31).

```
input: $\mathbf{x}$.
output: $\mathbf{f}$ and $\mathbf{g}$.
for $i \in \mathcal{N}$ do
    Compute $\mathbf{f}_i$ by the flux term (3.36)
        $$\mathbf{f}_i \leftarrow \mathbf{F}_i$$
    Remember the upstream weighting of the fluid part (3.28) of the
    transmissibility defined in (3.22).
    if $i \in \mathcal{I} \cup \mathcal{P}$ then
        Add the source term (3.43) to $\mathbf{f}_i$
            $$\mathbf{f}_i \leftarrow \mathbf{f}_i + (\mathbf{Q}V)_i$$
    Compute $\mathbf{g}_i$ by the accumulation term (3.41)
        $$\mathbf{g}_i \leftarrow (\mathbf{C}V)_i$$
```

compute the Jacobian of the both the accumulation term and the source term in (3.24). The accumulation term for the i^{th} grid block is dependent on pressure and saturation in this block alone. Consequently, the Jacobian of the accumulation term

$$\frac{\partial \mathbf{C}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{C}_1}{\partial \mathbf{x}_1} & & & \\ & \frac{\partial \mathbf{C}_2}{\partial \mathbf{x}_2} & & \\ & & \ddots & \\ & & & \frac{\partial \mathbf{C}_N}{\partial \mathbf{x}_N} \end{bmatrix} \tag{3.44}$$

is a block diagonal matrix with N blocks. Each block in (3.44) is of size 2×2 and defined as follows

Algorithm 3.6.2: Building the Jacobian matrix, $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, of the right-hand side and the Jacobian matrix, $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$, the left-hand side of the differential equation model (3.31).

```
input: $\mathbf{x}$.
output: $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$.
for $i \in \mathcal{N}$ do
    Compute $\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}$ by the Jacobian of the flux term (3.38)
        $$\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i} \leftarrow \frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_i}$$
    for $j \in \mathcal{N}^{(i)}$ do
        Compute $\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j}$ by the Jacobian of the flux term (3.38)
            $$\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_j} \leftarrow \frac{\partial \mathbf{F}_i}{\partial \mathbf{x}_j}$$
    Remember the upstream weighting of the fluid part (3.28) of
    the transmissibility defined in (3.22).
    if $i \in \mathcal{I} \cup \mathcal{P}$ then
        Add the Jacobian of the source term (3.47) to $\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}$
            $$\frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i} \leftarrow \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i} + \frac{\partial \mathbf{Q}_i}{\partial \mathbf{x}_i} V_i$$
    Compute $\frac{\partial \mathbf{g}_i}{\partial \mathbf{x}_i}$ by the Jacobian of the accumulation term (3.45)
        $$\frac{\partial \mathbf{g}_i}{\partial \mathbf{x}_i} \leftarrow \frac{\partial \mathbf{C}_i}{\partial \mathbf{x}_i} V_i$$
```

$$\frac{\partial \mathbf{C}_i}{\partial \mathbf{x}_i} = \begin{bmatrix} \frac{\partial C_{o,i}}{\partial P_i} & \frac{\partial C_{o,i}}{\partial S_i} \\ \frac{\partial C_{w,i}}{\partial P_i} & \frac{\partial C_{w,i}}{\partial S_i} \end{bmatrix} \quad i \in \mathcal{N} \tag{3.45}$$

Like the accumulation term, the source term in the i^{th} grid block is also dependent on pressure and saturation in this block alone. Consequently, the Jacobian of the source term

$$\frac{\partial \mathbf{Q}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{Q}_1}{\partial \mathbf{x}_1} & & & \\ & \frac{\partial \mathbf{Q}_2}{\partial \mathbf{x}_2} & & \\ & & \ddots & \\ & & & \frac{\partial \mathbf{Q}_N}{\partial \mathbf{x}_N} \end{bmatrix} \tag{3.46}$$

is a block diagonal matrix with N blocks. Each block in (3.46) is of size 2×2 and defined as follows

$$\frac{\partial \mathbf{Q}_i}{\partial \mathbf{x}_i} = \begin{bmatrix} \frac{\partial Q_{o,i}}{\partial P_i} & \frac{\partial Q_{o,i}}{\partial S_i} \\ \frac{\partial Q_{w,i}}{\partial P_i} & \frac{\partial Q_{w,i}}{\partial S_i} \end{bmatrix} \quad i \in \mathcal{N} \tag{3.47}$$

In particular

$$\frac{\partial \mathbf{Q}_i}{\partial \mathbf{x}_i} = \begin{cases} \bullet^{2\times 2} & i \in \mathcal{I} \cup \mathcal{P} \\ \mathbf{0}^{2\times 2} & i \in \mathcal{N} \backslash (\mathcal{I} \cup \mathcal{P}) \end{cases} \tag{3.48}$$

where the bullet $\bullet$ denotes that (3.47) is a block containing one or more non-zero elements. Remember that $\mathcal{I}$ is defined in Section 2.5 as the set of indices of grid blocks containing an injector, and $\mathcal{P}$ is defined as the set of indices of grid blocks containing a producer. Algorithm 3.6.2 shows how we build the Jacobian matrix of the right-hand side and the Jacobian matrix the left-hand side of the differential equation model (3.31). $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ denotes the Jacobian matrix of the right-hand side and $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$ denotes the Jacobian matrix of the left-hand side.

3.7 Summary

We have described the FVM for spatial discretization of the two-phase flow problem. We compute the fluxes with the use of TPFA. We use the flux terms to illustrate the Jacobian structure of the spatially discretized model.

We describe how to evaluate the interface transmissibilities: the geometrical part of the transmissibilities is computed by a harmonic averaging of the absolute permeabilities, and the fluid part is upwinded using the SPU scheme. We have presented a new differential equation model, as proposed in Paper A. This model maintains the mass preserving property of the spatially discretized two-phase flow problem upon temporal discretization. Many process simulation problems are derived upon conservation of mass, energy and momentum, and therefore we may use (3.31) as a formulation for such systems in general.

CHAPTER 4

Temporal Discretization

One of the key aspects in reservoir simulation is the integration of the differential equation system constituting the model. The number and the type of equations to be solved depend on the geological characteristics of the reservoir, the characteristics of the reservoir fluids, and the oil recovery process to be modelled. As a consequence of the system complexity it is necessary to find approximate solutions by numerical integration. Choosing the appropriate method of integration involves deciding on factors such as the order of the integration scheme, stability properties, and concern on computational efficiency. In addition, a robust adaptive step size control is essential to an efficient numerical integration. An adaptive step size selection aims to keep the error estimate of the solution bounded, i.e. close to a user-specified tolerance by adjusting the size of the time step.
The implicit pressure explicit saturation (IMPES) method [70, 71, 72], the sequential implicit method (SIM) [73, 74], and the implicit Euler method, normally referred to as the fully implicit method (FIM), are commonly used for temporal discretization in reservoir simulation. The IMPES scheme benefits from the explicit treatment of the saturation, the SIM benefits from the sequential approach, while the FIM offers unconditionally stability in

the sense of discrete approximations. However, in tems of controlling the integration error, the low order of the FIM leads to small integration steps, while the sequential approach and the explicit treatment of the saturation restrict the step sizes for IMPES and the SIM respectively. Current reservoir simulators often apply algorithms for time step adjustment that are based on safeguarded heuristics [31, 75, 76]. Even, if they are provided with an adaptive time step selection on the basis of error estimates [77, 78], they can neither guarantee convergence in the underlying equation solver, nor provide an estimate of the relation between convergence, integration error and step size.

We apply high order explicit singly diagonally implicit Runge-Kutta (ESDIRK) methods, with an embedded error estimate, for the numerical integration of the flow equations. ESDIRK methods have been used successfully for solution of convection-diffusion-reaction problems [79]. This class of methods is computationally efficient, and both A- and L-stable stiffly accurate ESDIRK methods of various order, with an embedded method for error estimation, have been derived [66, 67, 68, 69]. The non-linear residual equations arising in fully implicit methods have to be solved iteratively. Thus, in implicit integration, both the integration error and the convergence of the equation solver has to be monitored. Therefore we utilize a strategy for adaptive step size control. The controller is based on the error estimate provided by the embedded ESDIRK method as well as the convergence rate of the equation solver [80].

In Section 4.1 we discuss assumptions, advantages and drawbacks of the traditional IMPES method and the SIM approach, without going into computational details. Furthermore, with special emphasis on the differential equation system that we have proposed in Paper A, we present the Runge-Kutta method for numerical integration. We shortly describe the different classes of Runge-Kutta methods, and in particular the family of ESDIRK methods with an embedded error estimator for solution of (3.31) is presented. Section 4.2 is about error and convergence measures. That is, how the integration error is estimated, and how to measure the convergence rate in the iterative nonlinear equation solver. Section 4.3 focusses on adaptive step size selection, and in particular we describe the modifications of the error and convergence control that we suggest in Paper C. Finally, in Section 4.4 we compare and discuss the performance of three different ESDIRK methods that we have implemented, and we discuss the performance of the step size controller with and without our suggested modifications.

4.1 Integration Methods

In this section we discuss the IMPES method and the SIM, which are the most commonly used schemes for numerical integration in reservoir simulation. The different classes of Runge-Kutta methods are outlined and in particular ESDIRK methods are described.

4.1.1 IMPES

The IMPES method is motivated from the observation that pressure waves travel much faster than saturation waves. Pressure behaves elliptically in that pressure changes propagate almost instantly throughout the reservoir. In the limiting case of an incompressible system, pressure waves would travel infinitely fast. Saturation behaves hyperbolically, travelling in certain directions and at certain speeds through the reservoir. The two-phase flow model considered is essentially a mixed elliptic/hyperbolic problem and IMPES is a tailored numerical scheme that takes advantage of this mixed characteristic behaviour.
The basic assumption of the IMPES method is that the saturation is constant within a time step. Under this assumption, the flow equations are combined to obtain an equation for pressure and an equation for saturation. The IMPES method evaluates pressure implicitly and keeps saturation explicit. In that sense IMPES may be regarded as a simple implicit-explicit (IMEX) scheme, where the pressure equation and the saturation equation are solved sequentially:

- First, the pressure equation is solved using an implicit scheme. In most commercial simulators the pressure solution is obtained using backward Euler integration.

- After the pressure solution is obtained, the saturation is updated explicitly.

On a per time step basis, the computational cost of the IMPES method is significantly lower than that of a FIM. This is mainly because the IMPES approach reduces the system of implicit equations to one per grid block. However, the explicit treatment of the saturation introduces stability restrictions to the method. It may be difficult to obtain a pressure solution if saturation changes rapidly during a time step. The IMPES method can cope with this by decreasing the time step, but this approach requires a huge number of time steps to achieve a stable solution. This means that in reservoirs where properties change rapidly, a FIM may provide a solution in less computational time than the IMPES method, even though each time step takes longer to complete.

4.1.2 SIM

The motivation of the SIM is to improve the stability of the IMPES method by incorporating implicit treatment of the saturation equation, but still without solving for pressure and saturation simultaneously. Both methods use an operator splitting scheme, and the SIM can be coded to include IMPES as an option. However, the SIM is not an IMEX type method, since both pressure and saturation are treated implicitly. Like IMPES, the SIM also advances pressure and saturation in time sequentially. It uses the same pressure equation as IMPES but follows it with an implicit saturation calculation. Thus, both SIM steps require the solution of a system of implicit equations of same size. Therefore, the work for one time step with the SIM is around twice the work for the IMPES method.
The SIM only approximately conserves mass, since the second step of the method assumes incompressible flow. At the end of a time step the fluid volume in a grid block differs from the PV of the block. This volumetric discrepancy is interpreted as an erroneous injection/production of a small amount of fluid at that particular time step. It appears in the flow equations the same way a source term would. Using the volumetric discrepancy

as a driving force, the error may be corrected for by appropriate injection/production in the following time step. The SIM method is outlined as follows:

- First, the pressure equation is solved using an implicit scheme. In most commercial simulators the pressure solution is obtained using backward Euler integration.

- After the pressure solution is obtained, the saturation equation is solved using an implicit scheme. In most commercial simulators the saturation equation is solved using backward Euler integration.

- When the saturation has been advanced in time, the volumetric discrepancy is computed and used as a driving force in a source term that rectifies the volumetric error in the next time step.

The volume balancing may be exploited for reducing the computational cost of the second step in the SIM. This is done by limiting the number of iterations performed by the equation solver in the implicit integrator. Thus, the solution to the saturation equation is only approximate, and the resulting volume discrepancy is then corrected for as described above. The SIM may be faster than a FIM, but like IMPES it also suffers from stability restrictions.

4.1.3 Runge-Kutta Integration

We formulate an s-stage Runge-Kutta method for numerical integration of the differential equation system that we have proposed in (3.31) (see Paper A), in which $\mathbf{x}(t) \in \mathbb{R}^m$, $\mathbf{g}(\mathbf{x}) \in \mathbb{R}^m$ and $\mathbf{f}(t, \mathbf{x}) \in \mathbb{R}^m$, as follows

$$T_i = t_n + h_n c_i \qquad n \in \mathcal{N} \quad i \in \mathcal{S}^{(1)} \tag{4.1a}$$

$$\mathbf{g}(\mathbf{X}_i) = \mathbf{g}(\mathbf{x}_n) + h_n \sum_{j=1}^{s} a_{ij}\mathbf{f}(T_j, \mathbf{X}_j) \quad n \in \mathcal{N} \quad i \in \mathcal{S}^{(1)} \tag{4.1b}$$

$$\mathbf{g}(\mathbf{x}_{n+1}) = \mathbf{g}(\mathbf{x}_n) + h_n \sum_{j=1}^{s} b_j\mathbf{f}(T_j, \mathbf{X}_j) \quad n \in \mathcal{N} \tag{4.1c}$$

where $\mathbf{X}_i = \mathbf{X}_i(T_i)$ are the internal stage values being approximations to $\mathbf{x}(T_i)$ at $T_i = t_n + h_n c_i$. $\mathbf{x}_{n+1} = \mathbf{x}_{n+1}(t_{n+1})$ is the step computed at $t_{n+1} = t_n + h_n$. The set $\mathcal{N} = \{0, 1, \ldots, N-1\}$ denotes the time steps, and $\mathcal{S}^{(i)} = \{i, i+1, \ldots, s\}$ denotes the set of internal stages. The s-stage Runge-Kutta method (4.1) may be designated in terms of its Butchter tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline \mathbf{x}_{n+1} & b_1 & b_2 & \cdots & b_s \end{array} \tag{4.2}$$

from which different classes of Runge-Kutta methods can be obtained, depending on the structure of the matrix $A = [a_{ij}]$. This is illustrated in Figure 4.1. Explicit Runge-Kutta (ERK) methods have a strictly lower triangular A-matrix which allows all internal stages (4.1b) to be solved explicitly. Therefore, ERK methods are computationally fast and straightforward to implement, but they may suffer from stability limitations making them unsuitable for stiff problems [81]. The four remaining classes of Runge-Kutta methods are all implicit, that is, the value of the internal stages are no longer computed explicitly from the values of the previous stages. Implicit methods are characterized by an A-matrix that is not strictly lower triangular, and the stage values $\mathbf{X}_i$ are computed iteratively by solution of (4.1b). Fully implicit Runge-Kutta (FIRK) methods, identified by a full A-matrix, have

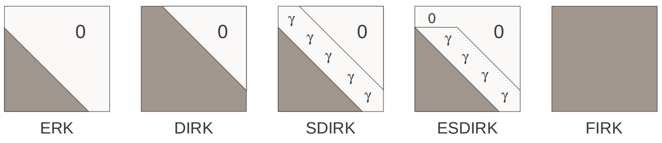


Figure 4.1: The A-matrix of Runge-Kutta methods.

excellent stability properties making them usefull for solving stiff systems of ODEs. However, the excellent stability properties comes with high computational cost in the sense that each integration step involves the solution of ms coupled nonlinear equations. To achieve some of the stability properties of the FIRK methods but at lower computational cost, various methods in between the ERK and the FIRK methods have been constructed. Diagonally implicit Runge-Kutta (DIRK) methods, singly diagonally implicit Runge-Kutta (SDIRK) methods and ESDIRK methods all have a lower triangular A-matrix. Instead of solving ms nonlinear equations simultaneously, like in the FIRK method, the internal stages in the DIRK, SDIRK and ESDIRK methods are decoupled in such a way that the solution of s systems of m nonlinear equations may be conducted sequentially.

4.1.4 ESDIRK Methods

ESDIRK methods have a lower triangular A-matrix. By construction they retain the stability properties of FIRK methods but at significant lower computational cost. Because $c_1 = 0$ and $a_{11} = 0$ the first stage in ESDIRK methods is explicitly implying that the first stage value equals the last step, i.e. $\mathbf{X}_1 = \mathbf{x}_n$. The subsequent stages are diagonally implicit. Thus, the stage values $\mathbf{X}_i$, $i \in \mathcal{S}^{(2)}$, may be solved sequentially by solution of the residual

$$\mathbf{R}(T_i, \mathbf{X}_i) = \mathbf{g}(\mathbf{X}_i) - h_n \gamma \mathbf{f}(T_i, \mathbf{X}_i) - \boldsymbol{\psi}_i = 0 \quad n \in \mathcal{N} \quad i \in \mathcal{S}^{(2)} \tag{4.3}$$

with the term

$$\boldsymbol{\psi}_i = \mathbf{g}(\mathbf{x}_n) + h_n \sum_{j=1}^{i-1} a_{ij}\mathbf{f}(T_j, \mathbf{X}_j) \quad n \in \mathcal{N} \quad i \in \mathcal{S}^{(2)} \tag{4.4}$$

using Newton-Raphson's (NR's) iterative method, in which we need the Jacobian of (4.3). The Jacobian of the residual is

$$\frac{\partial \mathbf{R}(T_i, \mathbf{X}_i)}{\partial \mathbf{X}_i} = \frac{d\mathbf{g}(\mathbf{X}_i)}{d\mathbf{X}_i} - h_n\gamma\frac{\partial \mathbf{f}(T_i, \mathbf{X}_i)}{\partial \mathbf{X}_i} \quad n \in \mathcal{N} \quad i \in \mathcal{S}^{(2)} \tag{4.5}$$

in which $\frac{d\mathbf{g}(\mathbf{X}_i)}{d\mathbf{X}_i}$ and $\frac{\partial \mathbf{f}(T_i,\mathbf{X}_i)}{\partial \mathbf{X}_i}$ are the Jacobians of the left- and right-hand sides in (3.31) respectively. Since ESDIRK methods are singly diagonally, the Jacobian may be reused in several iterations in the sense of a modified NR method [82]. The identical diagonal elements in the A-matrix implies that it may be sufficient just to update/factorize (4.5) once per integration step. Later in this chapter we will describe an algorithm for adaptive step size adjustment that reuses the same Jacobian matrix during several integration steps. We only consider methods that are assumed to be stiffly accurate by construction, i.e. $c_s = 1$ and $a_{sj} = b_j$, $j \in \mathcal{S}^{(1)}$. Which means that the quadrature function (4.1c) corresponds to the last internal stage in (4.1b). Consequently, the next step equals the last stage value, i.e. $\mathbf{x}_{n+1} = \mathbf{X}_s$. The Butcher tableau for a stiffly accurate ESDIRK method is presented in (4.6).

$$\begin{array}{c|cccccc}
0 & 0 & & & & & \\
c_2 & a_{21} & \gamma & & & & \\
c_3 & a_{31} & a_{32} & \gamma & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & & \\
c_{s-1} & a_{s-1,1} & a_{s-1,2} & a_{s-1,3} & \cdots & \gamma & \\
1 & b_1 & b_2 & b_3 & \cdots & b_{s-1} & \gamma \\
\hline
\mathbf{x}_{n+1} & b_1 & b_2 & b_3 & \cdots & b_{s-1} & \gamma
\end{array} \tag{4.6}$$

4.2 Error and Convergence Measures

In this section we describe how the integration error is estimated, and how this estimate is related to user specified absolute and relative tolerances. Finally, we describe how to measure the convergence rate in the iterative nonlinear equation solver.

4.2.1 Integration Error

The ESDIRK method stated in (4.6) may be equipped with an embedded method

$$\mathbf{g}(\hat{\mathbf{x}}_{n+1}) = \mathbf{g}(\mathbf{x}_n) + h_n \sum_{j=1}^{s} \hat{b}_j \mathbf{f}(T_j, \mathbf{X}_j) \quad n \in \mathcal{N} \tag{4.7}$$

which is used for computing the solution $\hat{\mathbf{x}}_{n+1}$. The embedded method is of different order than the basic method. By subtracting the two solutions (4.1c) and (4.7) we obtain an estimate of the local truncation error

$$\mathbf{e}_{n+1} = \mathbf{g}(\mathbf{x}_{n+1}) - \mathbf{g}(\hat{\mathbf{x}}_{n+1}) = h_n \sum_{j=1}^{s} d_j \mathbf{f}(T_j, \mathbf{X}_j) \quad n \in \mathcal{N} \tag{4.8}$$

which is an error estimate of the conserved quantities $\mathbf{g}(\mathbf{x}_{n+1})$ in (3.31) and not the states $\mathbf{x}_{n+1}$ themselves. This error estimate is essentially free, since it involves no additional function evaluations of the system. Measures of the error such as

$$r_{n+1} = \frac{1}{\sqrt{m}} \left\| \frac{|\mathbf{e}_{n+1}|}{\text{abstol} + |\mathbf{g}(\mathbf{x}_{n+1})| \cdot \text{reltol}} \right\|_2 \quad n \in \mathcal{N} \tag{4.9a}$$

$$r_{n+1} = \left\| \frac{|\mathbf{e}_{n+1}|}{\text{abstol} + |\mathbf{g}(\mathbf{x}_{n+1})| \cdot \text{reltol}} \right\|_p \quad n \in \mathcal{N} \tag{4.9b}$$

may be controlled adjusting the step size. $\frac{1}{\sqrt{m}} \|\cdot\|_2$ in (4.9a) denotes the root-mean-square, whereas $\|\cdot\|_p$ in (4.9b) is the p-norm. Usually we use the 2- or the ∞-norm as a measurement of the error. Only solution points for which the error-tolerance relation $r_{n+1} \leq 1$ are accepted. The Butcher tableau for an embedded ESDIRK method is presented in (4.10).

$$\begin{array}{c|cccccc}
0 & 0 & & & & & \\
c_2 & a_{21} & \gamma & & & & \\
c_3 & a_{31} & a_{32} & \gamma & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & & \\
c_{s-1} & a_{s-1,1} & a_{s-1,2} & a_{s-1,3} & \cdots & \gamma & \\
1 & b_1 & b_2 & b_3 & \cdots & b_{s-1} & \gamma \\
\hline
\mathbf{x}_{n+1} & b_1 & b_2 & b_3 & \cdots & b_{s-1} & \gamma \\
\hat{\mathbf{x}}_{n+1} & \hat{b}_1 & \hat{b}_2 & \hat{b}_3 & \cdots & \hat{b}_{s-1} & \hat{b}_s \\
\hline
\mathbf{e}_{n+1} & d_1 & d_2 & d_3 & \cdots & d_{s-1} & d_s
\end{array} \tag{4.10}$$

By ESDIRK$k\hat{k}$ we denote an ESDIRK integration method of order k with an embedded method of order $\hat{k}$. The method of order k is the advancing method that provides us with the solution. The method of order $\hat{k}$ is the embedded method that provides us with an error estimate of the solution. We have implemented three different ESDIRK methods: ESDIRK12, ESDIRK23 and ESDIRK34. The coefficients for the methods can be found in Appendix B. They are computed such that the integration method satisfies the Runge-Kutta order conditions, and such that the integration method is both A- and L-stable.

4.2.2 Convergence Rate

The solution of the stage values $\mathbf{X}_i$, $i \in \mathcal{S}^{(2)}$, is done iteratively by a modified NR method

$$\mathbf{J_R} \Delta \mathbf{X}_i^{(k)} = -\mathbf{R}(T_i, \mathbf{X}_i^{(k)}) \qquad i \in \mathcal{S}^{(2)} \tag{4.11a}$$

$$\mathbf{X}_i^{(k+1)} = \mathbf{X}_i^{(k)} + \Delta \mathbf{X}_i^{(k)} \qquad i \in \mathcal{S}^{(2)} \tag{4.11b}$$

in which the iteration matrix $\mathbf{J_R}$ is an approximation

$$\mathbf{J}_{\mathbf{R}} \approx \frac{\partial \mathbf{R}(T_i, \mathbf{X}_i^{(k)})}{\partial \mathbf{X}_i^{(k)}} \tag{4.12}$$

to the Jacobian (4.5) of the residual (4.3). We solve the linear equation system (4.11a) with a direct method. We use sparse LU-factorizations, because the Jacobian is nonsymmetric, see Section 3.5. The controller needs to supervise the equation solver and answer questions like: when should the Jacobian be evaluated/factorized, and what restrictions should be put on the step size to ensure good convergence? To help answer these questions, the convergence rate of the NR solver is monitored and measured. For reasons of robustness, the convergence rate is measured by the residuals [83]

$$\alpha = \max_{i,k} \frac{r_{\mathbf{R},i}^{(k)}}{r_{\mathbf{R},i}^{(k-1)}} \qquad i \in \mathcal{S}^{(2)} \tag{4.13}$$

in which the iteration error of the k^{th} iteration is computed as the following residual-tolerance relation

$$r_{\mathbf{R},i}^{(k)} = \frac{1}{\sqrt{m}} \left\| \frac{|\mathbf{R}(T_i, \mathbf{X}_i^{(k)})|}{\text{abstol} + |\mathbf{g}(\mathbf{X}_i^{(k)})| \cdot \text{reltol}} \right\|_2 \qquad i \in \mathcal{S}^{(2)} \tag{4.14a}$$

$$r_{\mathbf{R},i}^{(k)} = \left\| \frac{|\mathbf{R}(T_i, \mathbf{X}_i^{(k)})|}{\text{abstol} + |\mathbf{g}(\mathbf{X}_i^{(k)})| \cdot \text{reltol}} \right\|_p \qquad i \in \mathcal{S}^{(2)} \tag{4.14b}$$

using the same componentwise absolute and relative error tolerances as in (4.9). For robustness the iteration sequence should be contractive. If for some k during the iterations $\alpha \geq 1$ the iteration sequence is terminated. In the event of termination either a Jacobian update/factorization is called for or the step size is restricted. In case of convergence, the equation solver is successfully stopped when $r_{\mathbf{R},i}^{(k)} \leq \tau$. The choice of τ affects the efficiency of the controller [81]. We must choose $\tau \ll 1$, such that the error estimate will not be dominated by the contribution from the iteration error. A large value of τ may lead to one or more large components in the integration error, with too many rejected steps as a result. We have chosen $\tau = 0.1$ as a compromise between robustness and computational speed. In Algorithm 4.2.1 we present an outline of the ESDIRK integration method using adaptive step size control.

4.3 Step Size Selection

Adaptive step size selection is, in essence, a control problem, with the objective of producing numerical solutions for which the integration error (4.8) is kept within a certain tolerance. We have modified and implemented an algorithm for adaptive step size selection originally developed by [80]. Our modifications concern: the error control, the convergence control, and a simplification of the algorithm in general. In this section we briefly describe the key parts of the control strategy and the modifications of the error and convergence control that we propose. The motivation for our modifications and the modified version of the algorithm can be found in Paper C. Before we continue let us mention that we have two possible candidates for the next step size, h_{n+1}. The first candidate, h_r, is based on an estimate of the integration error. The second candidate, h_α, is based on the convergence rate in the equation solver. This is seen in Figure 4.2, depicting a block diagram of the modified control strategy.

4.3.1 Error Control

The original algorithm uses two rules for controlling the error of the solution: the asymptotic rule for step size adjustment, and a proportional integral (PI) step size controller. We start by describing the rules above, and finally the modification that we propose. The asymptotic rule for step size selection, which is the conventional control law that is commonly implemented in ODE solvers, is as follows

Algorithm 4.2.1: ESDIRK integration using adaptive step size control.

```
input: $\mathbf{x}_0$, $t_0$ and $t_f$ (the final time).
output: $\{\mathbf{x}_{n+1}\}_{n=0}^{N-1}$.
while $t_n < t_f$ do
    Assign
        $T_1 \leftarrow t_n$
        $\mathbf{X}_1 \leftarrow \mathbf{x}_n$
    for $i \in \mathcal{S}^{(2)}$ do
        Compute the stage value time $T_i$ by (4.1a).
        Assign
            $\mathbf{X}_i \leftarrow \mathbf{x}_n$
        Compute the residual $\mathbf{R}(T_i, \mathbf{X}_i)$ by (4.3).
        while iterations not converged do
            Compute the stage value $\mathbf{X}_i$ by solution of (4.11).
            Compute the residual-tolerance relation $r_{\mathbf{R},i}^{(k)}$ by (4.14).
            Compute the convergence rate $\alpha$ by (4.13).
    Assign
        $t_{n+1} \leftarrow T_s$
        $\mathbf{x}_{n+1} \leftarrow \mathbf{X}_s$
    Compute the error-tolerance relation $r_{n+1}$ by (4.9).
    Apply step size control cf. Algorithm 4.3.3.
```

$$h_r = \left(\frac{\varepsilon}{r_{n+1}}\right)^{1/\hat{k}} h_n \quad n \in \mathcal{N} \tag{4.15}$$

The asymptotic step size selection rule uses only current information about the step size and the error in order to achieve $r_{n+1} = \varepsilon$. Occasionally,

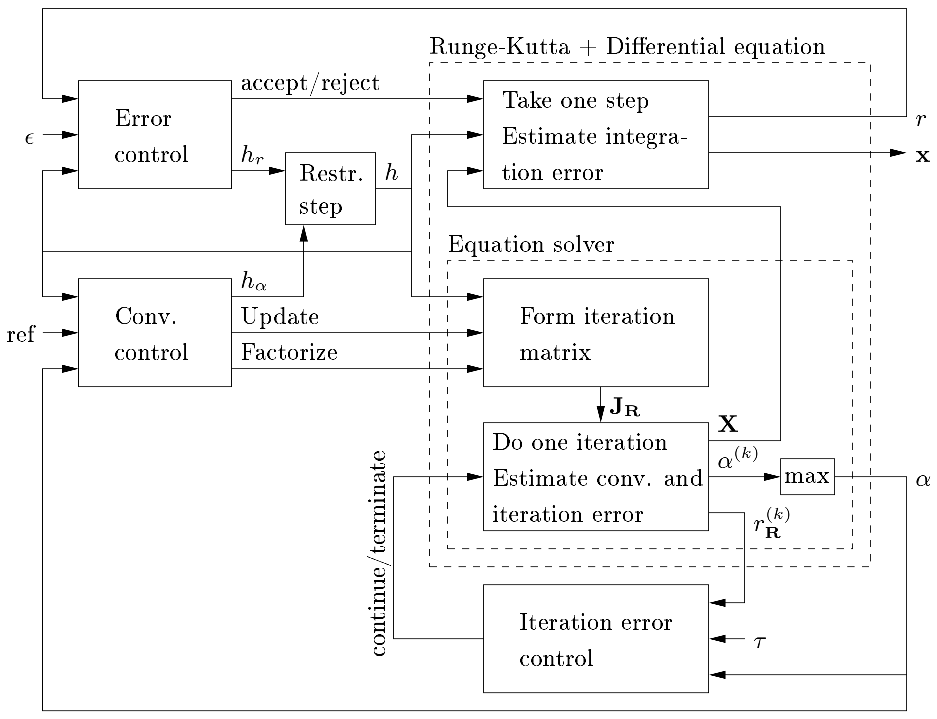


Figure 4.2: A block diagram of the modified control strategy in Paper C. $\mathbf{J_R}$ refers to the approximate Jacobian (4.12). "ref" is an abbreviation that covers both α_{ref}, α_{Jac} and α_{LU}.

the error estimate (4.8) may be unusually small or large, thus advocating (4.15) to produce very large step size changes. This affects the efficiency of the solver: the smaller the step size the more steps will be necessary to simulate the differential equations, and on the other hand, large steps risk being rejected. Both situations are expensive in terms of computations. This can to some extent be avoided by e.g. constraining the size of such changes. However, information about previous time steps and related errors should also be exploited in the controller to further improve robustness and the quality of the predicted time step. This is why the PI controller

$$h_r = \frac{h_n}{h_{n-1}} \left(\frac{r_n}{r_{n+1}}\right)^{k_1/\hat{k}} \left(\frac{\varepsilon}{r_{n+1}}\right)^{k_2/\hat{k}} h_n \quad n \in \mathcal{N} \tag{4.16}$$

is utilized as the core step size adjustment rule in the algorithm. k_1 and k_2 are the gain parameters of the proportional and the integral parts respectively. We use $k_1 = k_2 = 1$ corresponding to deadbeat control [80]. ε is the set point of the error-tolerance relation. In theory $\varepsilon = 1$ is an acceptable threshold. However, in the practical implementation we have chosen 0.8 as a safeguard.
While proceeding with the integration the controller must also ensure the convergence of the equation solver, which can be done in two ways: restricting the step size, and evaluate/factorize the Jacobian matrix (more on this in Section 4.3.2). The asymptotic step size adjustment rule is used, if the controller restricts the step size because of poor convergence. Convergence restricting the step size, i.e. $h_\alpha < h_r$, leads to an error estimate $r_{n+1} \ll \varepsilon$. Consequently, the size of the corresponding step produced by (4.15) may be disproportionately large, making the subsequent error estimates fluctuate. We avoid this by adding a filter to (4.15) by using information of previous step sizes

$$h_r = \frac{h_n}{h_{n-1}} \left(\frac{\varepsilon}{r_{n+1}}\right)^{1/\hat{k}} h_n \quad n \in \mathcal{N} \tag{4.17}$$

such that it supports the trend in former step sizes. This may dampen the fluctuations in the subsequent error estimates when the controller switches back from having used h_α to use h_r. The error control based on (4.15), (4.16) and (4.17) is presented in Algorithm 4.3.1.

4.3.2 Convergence Control

As mentioned above, to ensure good convergence in the modified NR iteration, the controller can either restrict the step size or update/factorize the Jacobian matrix. For large systems of differential equations that usually arise in reservoir simulation, large computational savings can be achieved,

Algorithm 4.3.1: Error control by adaptive step size adjustments.

```
input: Current inf.: $r_{n+1}$, $h_n$ and $\alpha$. Prior inf.: $r_n$, $h_{n-1}$ and $h_{LU}$.
output: $h_{n+1}$.
if iterations converged then
    if step accepted then
        if step restricted then
            Compute $h_r$ by (4.17).
        else
            Compute $h_r$ by (4.16).
        Assign
            $r_n \leftarrow r_{n+1}$
            $h_{n-1} \leftarrow h_n$
    else
        Compute $h_r$ by (4.15).
    $\vdots$
else
    $\vdots$
```

if the same Jacobian matrix is used during several integration steps. However, using a Jabobian based on old information leads to poor convergence of the equation solver. This can result in rejected integration steps and thus jeopardize the effeciency of the solver. As an alternative to frequent and expensive Jacobian evaluations and factorizations, the controller can restrict the step size to achieve good convergence. This however, may lead to a large number of integration steps, which increases the computational cost of the solution process. Both extremes are expensive in terms of computations. In the following we describe: how the original algorithm limits the step size in case of poor convergence, the strategy adopted for when to update/factorize the Jacobian matrix, and finally the modification that we

suggest.
The Jacobian is not evaluated at every integration step. Thus, the Jacobian is an approximation calculated at some solution point. The approximation will deteriorate gradually for the subsequent steps, but at least we expect good convergence at this point. However, in some cases the distance between the stage points may be too large, and the convergence will be poor although the Jacobian is based on current data. In this case, the only option is to decrease the distance between the stage points by reducing the step size. If convergence is poor, i.e. if $\alpha > \alpha_{ref}$ the step size should be chosen as

$$h_\alpha = \frac{\alpha_{ref}}{\alpha} h_n \quad n \in \mathcal{N} \tag{4.18}$$

to achieve $\alpha = \alpha_{ref}$ in the next step [80]. The main purpose of the step length control is to comply with the accuracy requirement. Therefore, the step size suggested by (4.18) has to be coordinated with one from the error control. Consequently, if poor convergence is observed in spite of a Jacobian based on current data, the step size is implemented as

$$h_{n+1} = \min(h_r, h_\alpha) \quad n \in \mathcal{N} \tag{4.19}$$

restricting the step size by convergence if $h_\alpha < h_r$. Slow convergence in the equation solver can to some extend be avoided by step size reductions. However, considering (4.5) we see that a step size change may imply a factorization of the Jacobian matrix. In some cases however, it is not enough simply to factorize the Jacobian matrix. To ensure convergence, it may be necessary to both evaluate and factorize the Jacobian. This implies, that we need a factorization/evaluation strategy for the Jacobian matrix. By monitoring the relative step size change since the last factorization was done, the following relation

$$|h_{n+1} - h_{LU}|/h_{LU} > \alpha_{LU} \quad n \in \mathcal{N} \tag{4.20}$$

is used as a rule for when a factorization should be applied. The strategy tries to anticipate possible convergence failures and factorizes the Jacobian, if a planned step size change is likely to jeopardize convergence. Should poor convergence be experienced despite of a recent factorization, the following rule

$$\alpha - |h_{n+1} - h_{LU}|/h_{LU} > \alpha_{Jac} \quad n \in \mathcal{N} \tag{4.21}$$

determines whether the Jacobian should be updated instead [84]. This strategy both monitors the convergence rate and balances the amount of computational time spent on Jacobian updates and factorizations versus function evaluations in the equation solver.
When convergence restricts the step size, i.e. $h_\alpha < h_r$, any value of α that deviates from α_{ref} will be brought to the reference in one step. This may be a too agressive strategy, making the subsequent error estimates fluctuate. We prevent this to some extent by using (4.17) instead of (4.15) for the subsequent calculation of h_r. In addition, we modify (4.18) to

$$h_\alpha = \left(\frac{\alpha_{ref}}{\alpha}\right)^{1/\hat{k}} h_n \quad n \in \mathcal{N} \tag{4.22}$$

such that the convergence restriction becomes less agressive. Thus, we may avoid that the corresponding error estimate deviates too much, and we get a smoother transition in the error estimate when the controller switches back from having used h_α to use h_r.
The efficiency of an implicit integration method depends highly on the convergence in the equation solver, i.e. the value of the set point α_{ref}. In most cases, any value $0.2 < \alpha_{ref} < 0.5$ is acceptable as reference for the convergence rate, with robustness favoring the lower values. We use $\alpha_{ref} = 0.4$ in our implementation [85, 80]. For the administration of the Jacobian, we have chosen $\alpha_{Jac} = 0.2$ and $\alpha_{LU} = 0.2$. For the two-phase flow problem, this choice seems like a good balance between the computational load of the equation solver in comparison to the computational cost of Jacobian evaluations/factorizations. With these settings we try to minimize the total computational time spent on the solution process. The maximum number of iterations allowed in the equation solver is 10. The convergence control based on (4.19), (4.20), (4.21) and (4.22) is presented in Algorithm 4.3.2.

Algorithm 4.3.2: Convergence control by adaptive step size adjustments and Jacobian evaluations/factorizations.

```
input: Current inf.: $r_{n+1}$, $h_n$ and $\alpha$. Prior inf.: $r_n$, $h_{n-1}$ and $h_{LU}$.
output: $h_{n+1}$.
if iterations converged then
    ⋮
    if new Jacobian and $\alpha > \alpha_{ref}$ then
        Compute $h_\alpha$ by (4.22).
        Convergence restrict the step size by (4.19).
    else
        Assign
            $h_{n+1} \leftarrow h_r$
    if (4.21) is satisfied then
        Evaluate and factorize the Jacobian.
        Assign
            $h_{LU} \leftarrow h_{n+1}$
    else if (4.20) is satisfied then
        Factorize the Jacobian.
        Assign
            $h_{LU} \leftarrow h_{n+1}$
else
    ⋮
```

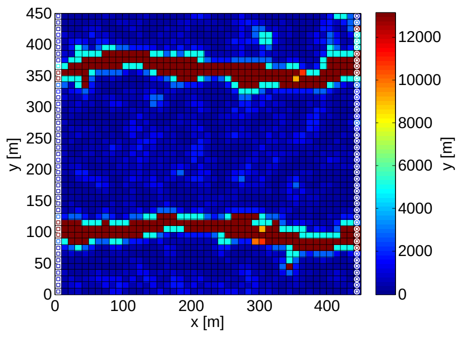
The complete modified PI step size controller for an implicit Runge-Kutta method is presented in Algorithm 4.3.3.

Algorithm 4.3.3: The complete modified PI step size controller for an implicit Runge-Kutta method.

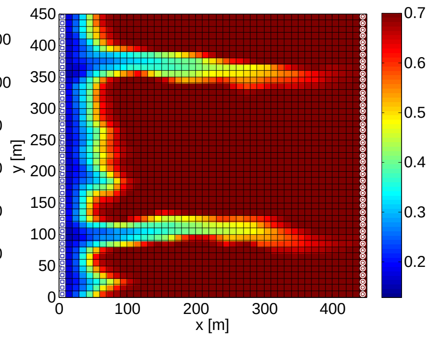
```
input: Current inf.: $r_{n+1}$, $h_n$ and $\alpha$. Prior inf.: $r_n$, $h_{n-1}$ and $h_{LU}$.
output: $h_{n+1}$.
if iterations converged then
    Apply error control cf. Algorithm 4.3.1.
    Apply convergence control cf. Algorithm 4.3.2.
else
    if new Jacobian then
        if $\alpha > \alpha_{ref}$ then
            Compute $h_\alpha$ by (4.22).
            Assign
                $$h_{n+1} \leftarrow h_\alpha$$
        else
            Assign
                $$h_{n+1} \leftarrow h_n/2$$
    else
        Evaluate the Jacobian.
    Factorize the Jacobian.
    Assign
        $$h_{LU} \leftarrow h_{n+1}$$
```

4.4 Choice of Methods

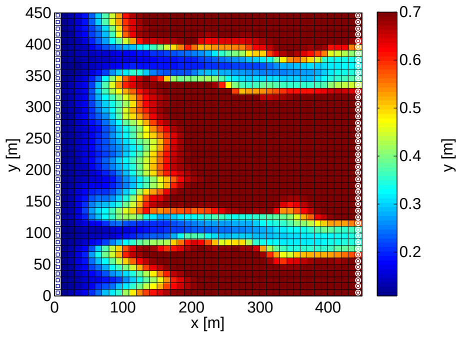
In this section we use the two-phase flow problem and the permeability field in Figure 4.3a as a benchmark problem. The field consists of $45 \times 45 \times 1$ grid blocks. 45 vertical injectors are placed along the left-hand side and 45 vertical producers are placed along the right-hand side. Figure 4.3b, 4.3c and 4.3d show the field development if we inject a total of 1 PV (see page 15) of water over a period of 360 days. We compare and discuss the performance of the three different ESDIRK methods used for the solution of



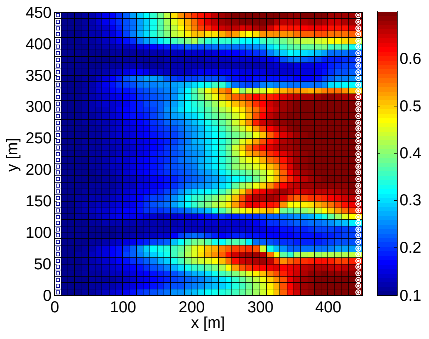
(a) Permeability field [mDa] [1].



(b) 0.11 PV injected after 40 days.



(c) 0.33 PV injected after 120 days.



(d) 1 PV injected after 360 days.

Figure 4.3: Water flooding example showing the oil saturations after 40, 120 and 360 days of injection with a total of 1 PV of water injected over a period of 360 days. The field in Figure 4.3a consists of $45 \times 45 \times 1$ grid blocks. 45 vertical injectors are placed along the left-hand side and 45 vertical producers are placed along the right-hand side.

the water flooding problem. Finally, we discuss the performance of the step size controller in its original form in relation to the version that includes our proposed modifications.

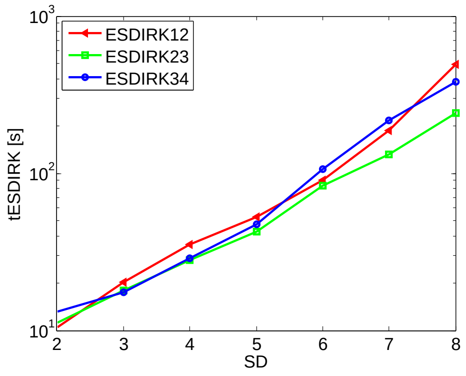
By PI97 we denote the controller in its original form, and by PI10 we refer to the modified version (according to Paper C). For the work-precision diagrams we used a fixed absolute tolerance of 10^{-8} and relative tolerances in the range from 10^{-8} to 10^{-2}. We denote these as significant digits (SDs), such that SD $= 4$ corresponds to a relative tolerance of 10^{-4}. The number of function evaluations, which are listed in the following, can be directly equated with the number of iterations in the equation solver.

4.4.1 Choice of ESDIRK Methods

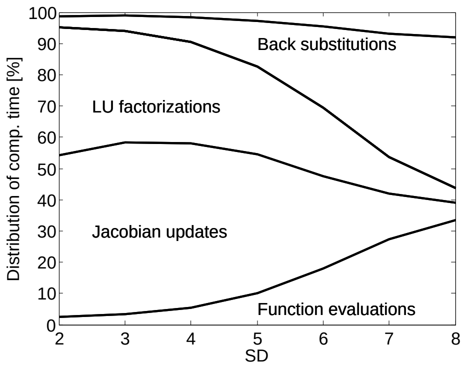
From the work-precision diagram in Figure 4.4a, we see that the total computational cost of ESDIRK12 increases in a more profound way than the other two methos. This is due to the small step sizes, which are necessary for the method in order to satisfy the required accuracy of the solution. This implies an increased workload of the equation solver when trying to retain $(r_R)_i^k \leq \tau$. In addition, small step sizes lead to an increase in overhead time, which can be seen in Figure 4.4b. As can be seen in Figure 4.4c and Figure 4.4d, ESDIRK23 and ESDIRK34 are better at maintaining an appropriate distribution of the workload as the requirements in accuracy increase. Except for SD $= 2$, we can observe from the work-precision diagram of the three methods, that ESDIRK23 is overall the most computationally efficient method for temporal discretization of the two-phase flow problem.

4.4.2 Choice of step size Controller

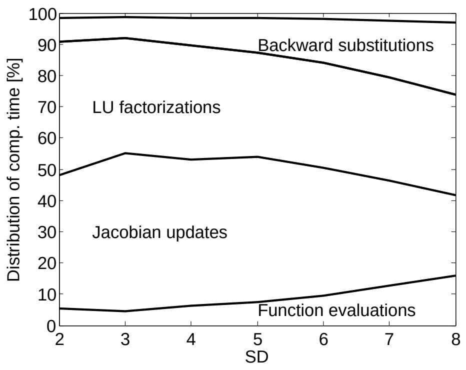
The step size sequences for the PI97 and the PI10 controllers are depicted in Figure 4.5a and in Figure 4.5b respectively. It must be mentioned that `nStep`, `nFail` and `nSlow` refer to the number of timesteps used in order to obtain the solution, the number of steps rejected by the error control, i.e. $r_{n+1} > 1$, and the amount of steps rejected because the maximum number of iterations is reached. By comparing the performance of the two controllers, we see that fewer Jacobian evaluations/factorizations are required by the PI10 controller in the sense of maintaining good convergence in the equation solver. Thus fewer iterations are necessary in order to complete the integration. Because of the less aggressive step size change suggested by



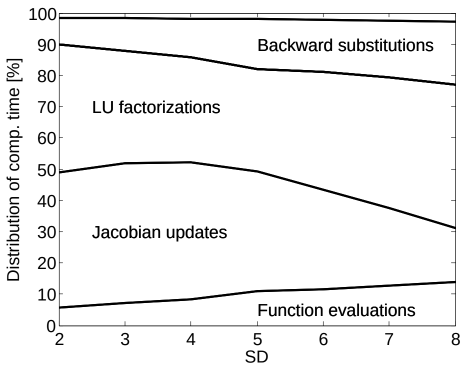
(a) Total computational cost.



(b) Workload distribution, ESDIRK12.



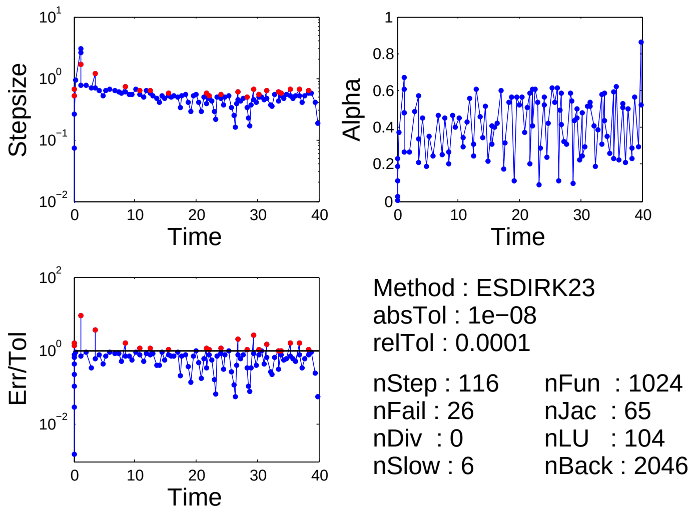
(c) Workload distribution, ESDIRK23.



(d) Workload distribution, ESDIRK34.

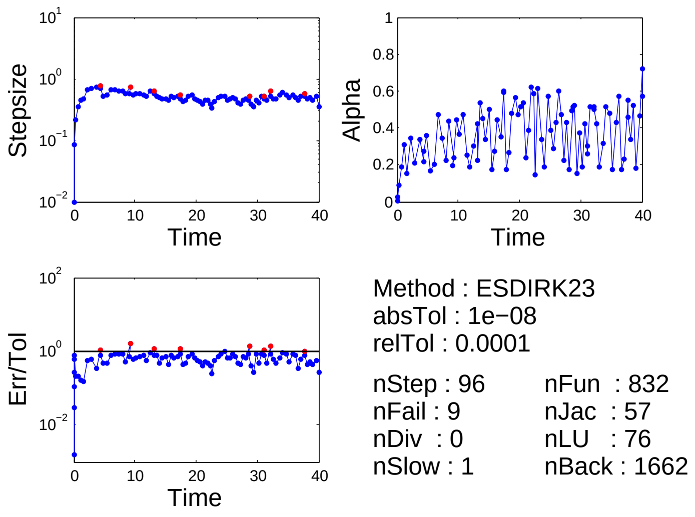
Figure 4.4: Total computational cost and distribution of workload of ESDIRK12, ESDIRK23 and ESDIRK34 with the use of the P10 controller (the uppermost patch in the diagrams is overhead time).

(4.22) in connection with the filtering of the asymptotic step size selection rule, a smoother step size sequence is obtained by the PI10 controller. This implies that the PI10 controller provides a solution with lesser rejected steps when compared to the performance of the PI97 controller. In Figure 4.5a several solution points for which $r_{n+1} \ll \varepsilon$ can be observed. For the same



```
Method : ESDIRK23
absTol : 1e−08
relTol : 0.0001

nStep : 116       nFun  : 1024
nFail : 26        nJac  : 65
nDiv  : 0         nLU   : 104
nSlow : 6         nBack : 2046
```

(a) Performance of the PI97 controller applied to ESDIRK23.



```
Method : ESDIRK23
absTol : 1e−08
relTol : 0.0001

nStep : 96        nFun  : 832
nFail : 9         nJac  : 57
nDiv  : 0         nLU   : 76
nSlow : 1         nBack : 1662
```

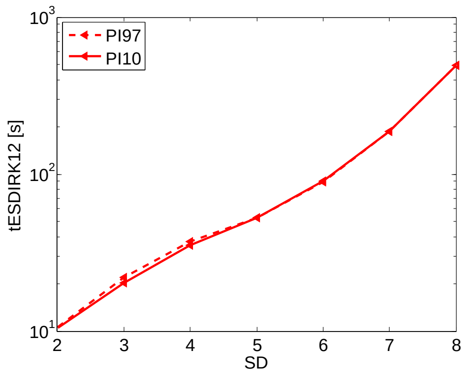
(b) Performance of the PI10 controller applied to ESDIRK23.

Figure 4.5: Performances of the PI97 and the PI10 controller applied to ESDIRK23 computing the solution shown in Figure 4.3b.

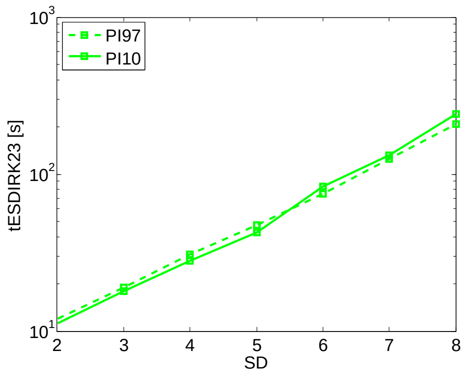
solution points the corresponding step sizes are small. Due to the ability of (4.22) to increase the step size as well, this behaviour is only to a lesser extent observed in Figure 4.5b. Hence the PI10 controller may produce larger step sizes in situations where the integration error and the convergence rate are below their respective set points. In Figure 4.6 we compare the computational cost of the two controllers. We notice, that in comparison to the PI97 controller, a better performance is obtained in the range from 2 to 5 SDs for the PI10 controller. The requirements for the accuracy in reservoir simulation is often within the range mentioned above. Therefore, we propose the use of the modified step length controller in Paper C for implicit integration of dynamic systems such as the two-phase flow problem.

4.5 Summary

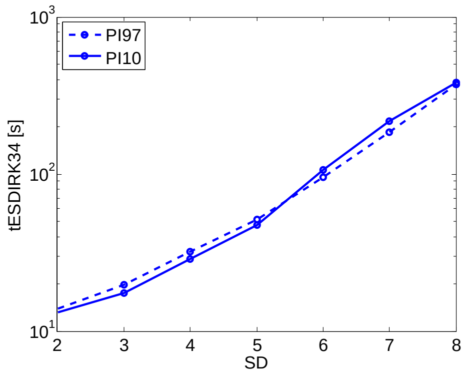
We have described the most commonly used schemes for temporal discretization in reservoir simulation: the FIM, the IMPES method and the SIM. With special emphasies on the differential equation system (3.31) that we propose in Paper A, we have presented three different ESDIRK methods with an embedded error estimator: ESDIRK12, ESDIRK23 and ESDIRK34. All three methods are both A- and L-stable, as well as stiffly accurate. We have described an adaptive step size control that is based on the embedded error estimator. In particular, we have described the modifications of the error and convergence control that we suggest in Paper C. Finally, we have discussed the performance of the three ESDIRK methods, and the performance of the step size controller both in its original form and with the modifications that we suggest.



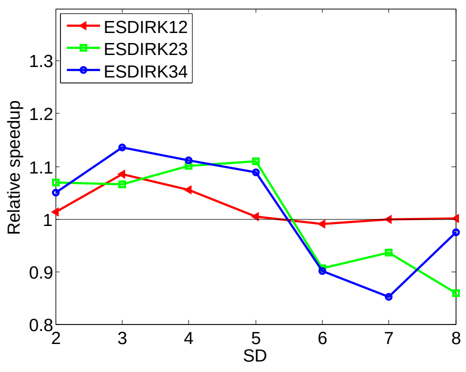
(a) Total computational cost, ESDIRK12.



(b) Computational cost of ESDIRK23.



(c) Computational cost of ESDIRK34.



(d) Relative speedup, $\frac{t_{\mathbf{PI10}}}{t_{\mathbf{PI97}}}$.

Figure 4.6: Comparison of the PI97 and the PI10 controller separately used with ESDIRK12, ESDIRK 23 and ESDIRK34.

CHAPTER 5

Production Optimization

Petroleum reservoirs are subsurface formations of porous rocks with hydrocarbons trapped in the pores. Initially, the reservoir pressure may be sufficiently large to push the fluids to the production facilities. However, as the fluids are produced the pressure declines and production reduces over time. When the natural pressure becomes insufficient, the pressure must be maintained artificially by injection of water. Conventional technologies for recovery leaves more than 50% of the oil in the reservoir. Wells with adjustable downhole flow control devices coupled with modern control technology offer the potential to increase the oil recovery significantly. Optimal control of smart-wells have been introduced [1]. In these applications, downhole sensor equipment and remotely controlled valves are used in combination with large-scale subsurface flow models and gradient based optimization methods in a NMPC framework to increase the production and economic value of an oil reservoir [15, 25, 13, 26, 19]. Wether the objective is to maximize recovery or some financial measure like net present value (NPV), the increased production is achieved by manipulation of the well rates and BHPs of the injection and production wells. The optimal water injection rates and production well BHPs are computed by solution

of a large-scale constrained optimal control problem.
In this chapter, we focus on maximizing the economical value of an oil field and describe the gradient based method to compute the optimal control strategy. The discrete-time optimal control problem can be solved using single-shooting [86, 87], multiple-shooting [88, 89], and the simultaneous method [90, 91, 92]. Reservoir models are large-scale and the number of states are easily in the order of magnitude of 10^5-10^6 for realistic problems. Thus, we describe a single-shooting algorithm for solution of the non-linear constrained optimal control problem. An ESDIRK method with an adaptive step size control, see Chapter 4 and Paper C, is used for computationally efficient solution of the model. The gradients are efficiently computed using the adjoint method [93, 36, 22, 37]. The constrained optimization is performed using quasi-Newton sequential quadratic programming (SQP) with line-search and modified BFGS approximations [94]. The adjoint equations associated with the integration scheme are solved by integrating backwards in time. The necessary information for the adjoint computation is computed and stored during the forward solution of the model. The backward adjoint computation assembles this information to compute the gradients [13]. With the use of the spatial discretization presented in Section 3.1 - 3.2, we demonstrate the optimal control strategy using the 2-dimensional water flooding example presented in Chapter 2.
The chapter is organized as follows. Section 5.1 states the general constrained optimal control problem using the novel representation of the system dynamics that we propose in Section 3.4 and Paper A. The numerical methods for the constrained optimal control problem are described in Section 5.2 - 5.4. Section 5.5 describes specific details related to the objective function and the constraints for the water flooding production optimization problem. In this section we also describe the numerical case study illustrating the optimization algorithm when applied to the water flooding problem.

5.1 Optimal Control Problem

We formulate the water flooding problem as a continuous time Bolza problem

$$\min_{[\mathbf{x}(t),\mathbf{u}(t)]_{t_0}^{t_f}} \quad \int_{t_0}^{t_f} J(\mathbf{x}(t),\mathbf{u}(t))dt + H(\mathbf{x}(t_f)) \tag{5.1a}$$

$$\text{s.t.} \quad \frac{d\mathbf{g}(\mathbf{x}(t))}{dt} = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t)) \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{5.1b}$$

$$\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max} \tag{5.1c}$$

$$-\Delta\mathbf{u}_{min} \leq \frac{d\mathbf{u}}{dt}(t) \leq \Delta\mathbf{u}_{max} \tag{5.1d}$$

The algorithm developed for solution of this problem is suitable for production optimization of oil reservoirs. $\mathbf{x}(t)$ is a vector holding the system states, and $\mathbf{u}(t)$ is a vector holding the manipulated variables. (5.1b) represents the dynamic model formulated as the differential equation system that we have proposed in (3.31) (see Paper A). We use a zero-order-hold parameterization for $\mathbf{u}(t)$. This implies that the constraints (5.1d) should be interpreted as the movement constraints (5.4d).
To convert the infinite-dimensional problem (5.1) into a numerically tractable finite-dimensional problem, we divide the temporal domain $[t_0, t_f]$ into K control steps and each control step into N_k time steps for the integration of the differential equations. We then define a set of control step indices $\mathcal{K}_i = \{i, i+1, \ldots, K-1\}$ and a set of time step indices $\mathcal{N}_k = \{0, 1, \ldots, N_k - 1\}$, $k \in \mathcal{K}_0$. The number of control steps is known in advance due to the zero-order-hold parametrization of the manipulated variables. A control step $k \in \mathcal{K}_0$ is defined as an interval between the times $t_{0,k}$ and $t_{N_k,k}$. Note that $t_{0,0} = t_0$ and $t_{N_{K-1},K-1} = t_f$. For a given control interval k, the number of time steps are not known in advance as we use an adaptive step length controller in the numerical integrator, see Chapter 4. This indexing of the control steps and the time steps is illustrated in Fig. 5.1.
Using the ESDIRK12 scheme for temporal discretization of (3.31), we can

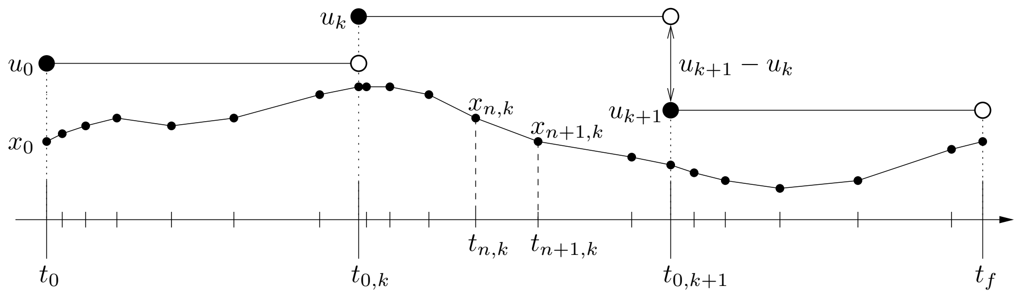


Figure 5.1: The zero order hold parametrization and the relation between the control steps and the time steps. For a given time step $t_{n,k}$ in a given control step k the optimal control problem can be described by the system states $\mathbf{x}_{n,k}$ and the control settings $\mathbf{u}_k$.

compute the trajectory $\{\{\mathbf{x}_{n+1,k}\}_{n=0}^{N_k-1}\}_{k=0}^{K-1}$ as the solution of the system of difference equations

$$\mathbf{g}(\mathbf{x}_{n+1,k}) = \mathbf{g}(\mathbf{x}_{n,k}) - \mathbf{f}(\mathbf{x}_{n+1,k}, \mathbf{u}_k)h_{n,k} \quad n \in \mathcal{N}_k \quad k \in \mathcal{K}_0 \tag{5.2}$$

in which $\mathbf{x}(t_{n,k}) = \mathbf{x}_{n,k}$ and $\mathbf{u}(t_{n,k}) = \mathbf{u}_k$. For notational convenience we define the residual function

$$\begin{aligned} &\mathbf{R}_{n+1,k}(\mathbf{x}_{n+1.k}, \mathbf{x}_{n,k}, \mathbf{u}_k) = \\ &\quad \mathbf{g}(\mathbf{x}_{n+1,k}) - \mathbf{g}(\mathbf{x}_{n,k}) - \mathbf{f}(\mathbf{x}_{n+1,k}, \mathbf{u}_k)h_{n,k} = 0 \end{aligned} \tag{5.3}$$

and formulate the continuous-time optimal control problem (5.1) as the following discrete-time optimal control problem

$$\min_{\{\{\mathbf{x}_{n+1,k}\}_{n=0}^{N_k-1},\mathbf{u}_k\}_{k=0}^{K-1}} \quad \sum_{k=0}^{K-1}\sum_{n=0}^{N_k-1} J_{n,k}(\mathbf{x}_{n+1,k},\mathbf{x}_{n,k},\mathbf{u}_k) \tag{5.4a}$$

$$\text{s.t.} \quad \mathbf{R}_{n+1,k}(\mathbf{x}_{n+1.k},\mathbf{x}_{n,k},\mathbf{u}_k) = 0 \tag{5.4b}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \tag{5.4c}$$

$$-\Delta\mathbf{u}_{min} \leq \Delta\mathbf{u}_k \leq \Delta\mathbf{u}_{max} \tag{5.4d}$$

where $\Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}$ and

$$J_{n,k}(\mathbf{x}_{n+1,k},\mathbf{x}_{n,k},\mathbf{u}_k) = \int_{t_{n,k}}^{t_{n+1,k}} J(\mathbf{x}(t),\mathbf{u}_k)dt \quad n \in \mathcal{N}_k \quad k \in \mathcal{K}_0 \tag{5.5}$$

Furthermore, we define the cost-to-go function $H_{N_k,k}(\mathbf{x}_{N_k,k}) = 0$. (5.5) may be solved with the use of some quadrature rule, e.g. the trapezoidal method, Simpson's rule, etc. We use the quadrature rule defined by the method used for integration of the differential equations (4.1).

5.2 Single Shooting Optimization

To keep the dimension of the optimization problem small and to be able to use adaptive temporal step size, we use the single-shooting method for solution of the discrete-time problem (5.4). In the single-shooting method, the manipulated variables are fixed at each iteration and used to solve the difference equations (5.4b) numerically. Knowledge of the initial state $\mathbf{x}_0$, the manipulated variables $\{\mathbf{u}_k\}_{k=0}^{K-1}$, and the requirement that the systems dynamics are satisfied determines the states $\{\{\mathbf{x}_{n+1,k}\}_{n=0}^{N_k-1}\}_{k=0}^{K-1}$. This implies that the states may be regarded as dependent variables as they can be expressed as functions of the manipulated variables and the initial state. In practical computations, the system constraints (5.4b) are satisfied by solving (5.3), i.e. by doing a system simulation. In this way, a single-shooting method for (5.4) can be approximated by the finite dimensional optimization problem

$$
\begin{aligned}
\min_{\{\mathbf{u}_k\}_{k=0}^{K-1}} \quad & \psi(\{\mathbf{u}_k\}_{k=0}^{K-1}, \mathbf{x}_0) && \text{(5.6a)}\\
\text{s.t.} \quad & \mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} && \text{(5.6b)}\\
& -\Delta\mathbf{u}_{min} \leq \Delta\mathbf{u}_k \leq \Delta\mathbf{u}_{max} && \text{(5.6c)}
\end{aligned}
$$

in which

$$
\begin{aligned}
&\psi(\{\mathbf{u}_k\}_{k=0}^{K-1}, \mathbf{x}_0) = \\
&\left\{ \sum_{k=0}^{K-1} \sum_{n=0}^{N_k-1} J_{n,k}(\mathbf{x}_{n+1,k}, \mathbf{x}_{n,k}, \mathbf{u}_k) : \right. \\
&\left. \mathbf{R}_{n+1,k}(\mathbf{x}_{n+1,k}, \mathbf{x}_{n,k}, \mathbf{u}_k) = 0 \quad n \in \mathcal{N}_k \quad k \in \mathcal{K}_0 \right\}
\end{aligned}
\tag{5.7}
$$

is the objective function that only depends on the manipulated variables and the fixed initial state.

5.3 Gradient Computation by the Adjoint Method

In solving the reduced system (5.6), we must compute the gradient $\nabla_{\mathbf{u}_k}\psi$. The system states in dynamic optimization problems are dependent on the control variables, in the sense that any past change of the control variables has an influence on all subsequent system states. Consequently, the gradient information of (5.7) is not directly accessible. The necessary information for computing $\nabla_{\mathbf{u}_k}\psi$ is obtained during the simulation step at each optimization iteration in the single-shooting approach. The adjoint method uses this information efficiently to compute the gradients.
Assume that the manipulated variables $\{\mathbf{u}_k\}_{k=0}^{K-1}$ satisfies the input constraints (5.4c) and (5.4d). Thus, we may define the Lagrangian of (5.4) as follows

$$\begin{aligned}\mathcal{L}(\{\{\mathbf{x}_{n+1,k}\}_{n=0}^{N_k-1}, \mathbf{u}_k, \{\boldsymbol{\lambda}_{n+1,k}\}_{n=0}^{N_k-1}\}_{k=0}^{K-1}) = \\ \sum_{k=0}^{K-1}\sum_{n=0}^{N_k-1} (J_{n,k}(\mathbf{x}_{n+1,k}, \mathbf{x}_{n,k}, \mathbf{u}_k) \\ - \boldsymbol{\lambda}_{n+1,k}^T \mathbf{R}_{n+1,k}(\mathbf{x}_{n+1,k}, \mathbf{x}_{n,k}, \mathbf{u}_k))\end{aligned} \tag{5.8}$$

The adjoint method can be derived using the first order necessary conditions for optimality with respect to the dependent variables, $\{\{\mathbf{x}_{n+1,k}\}_{n=0}^{N_k-1}\}_{k=0}^{K-1}$, and the Lagrange multipliers (adjoint variables), $\{\{\boldsymbol{\lambda}_{n+1,k}\}_{n=0}^{N_k-1}\}_{k=0}^{K-1}$, i.e.

$$\nabla_{\mathbf{x}_{n+1,k}} \mathcal{L} = 0 \quad n \in \mathcal{N}_k \quad k \in \mathcal{K}_0 \tag{5.9a}$$

$$\nabla_{\boldsymbol{\lambda}_{n+1,k}} \mathcal{L} = 0 \quad n \in \mathcal{N}_k \quad k \in \mathcal{K}_0 \tag{5.9b}$$

The first order optimality condition corresponding to the adjoint derivative (5.9b) yields the system constraints (5.4b). The first order optimality condition corresponding to the state derivative (5.9a) is

$$\begin{aligned}&\nabla_{\mathbf{x}_{n,k}} J_{n-1,k}(\mathbf{x}_{n,k}, \mathbf{x}_{n-1,k}, \mathbf{u}_k) \\ &\quad - \nabla_{\mathbf{x}_{n,k}} \mathbf{R}_{n,k}(\mathbf{x}_{n,k}, \mathbf{x}_{n-1,k}, \mathbf{u}_k)\boldsymbol{\lambda}_{n,k} \\ &\quad + \nabla_{\mathbf{x}_{n,k}} J_{n,k}(\mathbf{x}_{n+1,k}, \mathbf{x}_{n,k}, \mathbf{u}_k) \\ &\quad - \nabla_{\mathbf{x}_{n,k}} \mathbf{R}_{n+1,k}(\mathbf{x}_{n+1,k}, \mathbf{x}_{n,k}, \mathbf{u}_k)\boldsymbol{\lambda}_{n+1,k} = 0\end{aligned} \tag{5.10}$$

which upon rearrangement yields

$$\begin{aligned}&\nabla_{\mathbf{x}_{n,k}} \mathbf{R}_{n,k}(\mathbf{x}_{n,k}, \mathbf{x}_{n-1,k}, \mathbf{u}_k)\boldsymbol{\lambda}_{n,k} = \\ &\quad + \nabla_{\mathbf{x}_{n,k}} J_{n-1,k}(\mathbf{x}_{n,k}, \mathbf{x}_{n-1,k}, \mathbf{u}_k) \\ &\quad + \nabla_{\mathbf{x}_{n,k}} J_{n,k}(\mathbf{x}_{n+1,k}, \mathbf{x}_{n,k}, \mathbf{u}_k) \\ &\quad - \nabla_{\mathbf{x}_{n,k}} \mathbf{R}_{n+1,k}(\mathbf{x}_{n+1,k}, \mathbf{x}_{n,k}, \mathbf{u}_k)\boldsymbol{\lambda}_{n+1,k}\end{aligned} \tag{5.11}$$

from which we can compute the adjoint variables $\boldsymbol{\lambda}_{n_k}$ in a backward recursion. The Lagrange multiplier at the final time is $\boldsymbol{\lambda}_{N_{K-1},K-1} = 0$ since the cost-to-go function is zero. $\boldsymbol{\lambda}_{N_{K-1},K-1} = 0$ is used to initialize the backward march for computation of the adjoint variables $\boldsymbol{\lambda}_{n,k}$, $n \in \mathcal{N}_k$, $k \in \mathcal{K}_0$. Special attention must be given when computing $\boldsymbol{\lambda}_{0,k+1}$ at the transition between $\mathbf{u}_k$ and $\mathbf{u}_{k+1}$ for $k \in \{0, 1, \dots, K-2\}$. At this point, the left-hand side and the first term on the right-hand side in (5.11) both belong to control step $\mathbf{u}_k$, while the second and third term on the right-hand side belong to control step $\mathbf{u}_{k+1}$. When the state variables $\{\{\mathbf{x}_{n+1,k}\}_{n=0}^{N_k-1}\}_{k=0}^{K-1}$ and the adjoint variables $\{\{\boldsymbol{\lambda}_{n+1,k}\}_{n=0}^{N_k-1}\}_{k=0}^{K-1}$ satisfy (5.9), we have

$$\begin{aligned}
&\psi(\{\mathbf{u}_k\}_{k=0}^{K-1}, \mathbf{x}_0) = \\
&\quad \Big\{ \mathcal{L}(\{\{\mathbf{x}_{n+1,k}\}_{n=0}^{N_k-1}, \mathbf{u}_k, \{\boldsymbol{\lambda}_{n+1,k}\}_{n=0}^{N_k-1}\}_{k=0}^{K-1}) : \\
&\qquad \mathbf{R}_{n+1,k}(\mathbf{x}_{n+1,k}, \mathbf{x}_{n,k}, \mathbf{u}_k) = 0 \quad n \in \mathcal{N}_k \quad k \in \mathcal{K}_0 \Big\}
\end{aligned} \tag{5.12}$$

such that we can compute the sensitivity $\nabla_{\mathbf{u}_k}\psi$ as the sensitivity $\nabla_{\mathbf{u}_k}\mathcal{L}$. Thus, considering the partial derivatives of (5.8) with respect to the manipulated variables $\{\mathbf{u}_k\}_{k=0}^{K-1}$, we may compute the gradients of the objective function of the reduced system (5.6) as follows

$$\begin{aligned}
&\nabla_{\mathbf{u}_k}\psi(\{\mathbf{u}_k\}_{k=0}^{K-1}, \mathbf{x}_0) = \\
&\quad \sum_{n=0}^{N_k-1} (\nabla_{\mathbf{u}_k} J_{n,k}(\mathbf{x}_{n+1,k}, \mathbf{x}_{n,k}, \mathbf{u}_k) \qquad k \in \mathcal{K}_0 \\
&\qquad - \nabla_{\mathbf{u}_k}\mathbf{R}_{n+1,k}(\mathbf{x}_{n+1,k}, \mathbf{x}_{n,k}, \mathbf{u}_k)\boldsymbol{\lambda}_{n+1,k})
\end{aligned} \tag{5.13}$$

The gradients $\nabla_{\mathbf{u}_k}\psi$ may be computed using (5.13) in combination with solution of the adjoint equations (5.11) marching backwards. In Algorithm 5.3.1 we show how to compute the gradients $\nabla_{\mathbf{u}_k}\psi$, $k \in \mathcal{K}$, using adjoints in a single shooting framework.

Algorithm 5.3.1: Computation of the gradients in (5.13) using adjoints in a single shooting framework.

```
input: $\mathbf{x}_0$ and $\{\mathbf{u}_k\}_{k=0}^{K-1}$.
output: $\{\nabla_{\mathbf{u}_k}\psi\}_{k=0}^{K-1}$.
forward computation
    Compute the system states $\{\{\mathbf{x}_{n+1,k}\}_{n=0}^{N^{(k)}}\}_{k=0}^{K-1}$ by solution of (5.3)
    using ESDIRK12.
backward computation
    Assign
        $\boldsymbol{\lambda}_{N^{(K-1)},K-1} \leftarrow 0$
    for $k = K-1, K-2, \dots, 1$ do
        for $n = N^{(k)}-1, N^{(k)}-2, \dots, 0$ do
            Compute the adjoint variables $\boldsymbol{\lambda}_{n,k}$ by solution of (5.11).
        Compute the gradient $\nabla_{\mathbf{u}_k}\psi$ by (5.13).
    for $n = N^{(0)}-1, N^{(0)}-2, \dots, 1$ do
        Compute the adjoint variables $\boldsymbol{\lambda}_{n,0}$ by solution of (5.11).
    Compute the gradient $\nabla_{\mathbf{u}_0}\psi$ by (5.13).
```

5.4 Sequential Quadratic Programming

We solve the reduced problem (5.6) using SQP with line-search and modified BFGS approximations, $\mathbf{B}$, of the Hessian of the Lagrangian of (5.6) [94]. In each iteration, we solve the convex quadratic program

$$\min_{\Delta\mathbf{u}} \quad \tfrac{1}{2}\Delta\mathbf{u}^T\mathbf{B}\Delta\mathbf{u} + \nabla_{\mathbf{u}}\psi^T\Delta\mathbf{u} \tag{5.14a}$$

$$\text{s.t.} \quad \nabla_{\mathbf{u}}\mathbf{c}(\mathbf{u})^T\Delta\mathbf{u} \geq -\mathbf{c}(\mathbf{u}) \tag{5.14b}$$

in which $\mathbf{u} = \{\mathbf{u}_k\}_{k=0}^{K-1}$. $\mathbf{c}(\mathbf{u})$ is a function that represents the input constraints (5.6b) - (5.6c). The optimal solution of (5.14), $\Delta\mathbf{u} = \{\Delta\mathbf{u}_k\}_{k=0}^{K-1}$,

Algorithm 5.4.1: Solution of the reduced problem (5.6) in an SQP framework using line-search and modified BFGS approximations.

```
input: $\mathbf{x}_0$ and an initial control strategy $\mathbf{u}$.
output: Optimal solution of $\mathbf{u}$.
while not optimal do
    Compute $\Delta\mathbf{u}$ by solving (5.14).
    Compute the line-search parameter $\alpha$ using Powell's exact penalty
    function and update the manipulated variables $\mathbf{u}$ by (5.15).
    Compute the gradients $\{\nabla_{\mathbf{u}_k}\psi\}_{k=0}^{K-1}$ cf. Algorithm 5.3.1.
    Update the Hessian in (5.14) using modified BFGS approximations.
```

combined with a line-search method based on Powell's exact penalty function [95] are used to determine the next iterate

$$\mathbf{u}^{(i+1)} = \mathbf{u}^{(i)} + \alpha^{(i)}\Delta\mathbf{u}^{(i)} \tag{5.15}$$

$\alpha^{(i)}$ is the line-search parameter. Algorithm 5.4.1 outlines how we solve the reduced problem (5.6) in an SQP framework using line-search and modified BFGS approximations.

5.5 Water Flooding Production Optimization

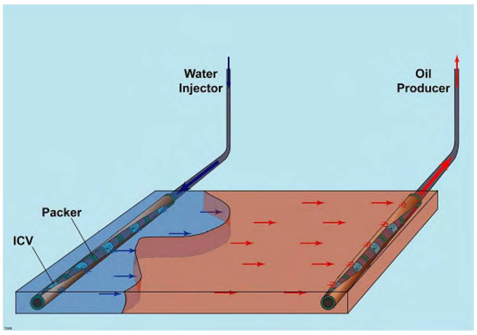
The objective of oil reservoir management is to maximize the economic value of the oil reservoir. Essentially, we want to produce as much oil as possible while keeping the operational cost at a minimum. We do this by maximizing the net present value (NPV). Consequently, the stage cost $J(t) = J(t, x(t), u(t))$ in (5.1) becomes

$$J(t) = -e^{-dt}\left(\sum_{j\in\mathcal{N}^{pro}}\left(\frac{r_o^{pro}}{\rho_o^\circ}q_{o,j}^{pro}(t) - \frac{r_w^{pro}}{\rho_w^\circ}q_{w,j}^{pro}(t)\right) - \sum_{j\in\mathcal{N}^{inj}} r_w^{inj} q_j^{inj}(t)\right) \tag{5.16}$$

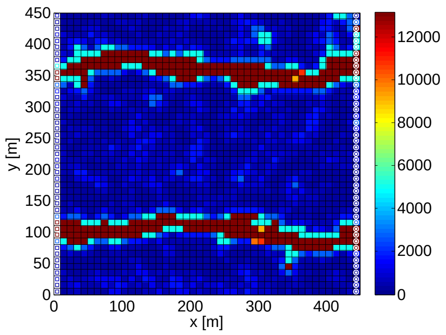
The factor e^{-dt} accounts for the time value of capital. The terms contributing to $J(t)$ are the value of the produced oil, the cost of separating water from the produced oil, and the cost of water injection. r_o^{pro} is the oil price, r_w^{pro} is the cost of water separation, and r_w^{inj} is the water injection cost. $q_{o,j}^{pro}(t)$ is the oil production and $q_{w,j}^{pro}(t)$ is the water production at production wells, $j \in \mathcal{N}^{pro}$, at time t. $q_j^{inj}(t)$ is the rate of injected water at the injection wells, $j \in \mathcal{N}^{inj}$, at time t. The term d denotes the continuous discount rate (cost of capital per unit time).
For water flooding using multiple injectors and producers, the well rates and pressures are adjusted by the optimal control problem (5.1) such that the NPV is maximized [1, 26]. The inequality constraints in (5.1) are bound constraints (5.1c) and rate-of-change constraints (5.1d). The bound constraints correspond to limitations on the water injection for each injection well and limitations on the BHP for each production well. The lower bounds on the water injection rates are zero, while the upper bound is computed such that no more than a predefined number of pore volumes (PVs) of water are allowed to be injected over the time horizon considered, $[t_0, t_f]$. That is, PV$_{max}$ is the maximum amount of water that may be injected into the reservoir during the planned production period $T = [t_0, t_f]$. These bound constraints implies that we will implicitly satisfy

$$0 \leq \sum_{j=1}^{\mathcal{N}^{inj}} \sum_{k=0}^{K-1} \sum_{n=0}^{N_k-1} \int_{t_{n,k}}^{t_{n+1,k}} q_j^{inj}(t)dt \leq \mathrm{PV}_{max} \tag{5.17}$$

The reservoir fluids are trapped inside the pores of a porous medium. The PV of a reservoir is defined by the fraction (the porosity) of the porous



(a) Two horizontal smart-wells.



(b) Permeability field (mDa).

Figure 5.2: Left: Schematic view of horizontal smart-wells in the water flooding problem (injector left and producer right) [1]. Right: Permeability field (mDa) with two high permeable streaks [1]. 45 injector segments (white squares to the left) and 45 producer segments (white circles to the right).

medium that is not occupied by reservoir rock, i.e. the void space of the reservoir rock that may contain the reservoir fluids. Ideally we would replace and thus produce all the reservoir fluids by injecting 1 PV of water into the reservoir. The BHPs in the production wells are restricted to be lower than the initial pressure of the reservoir. The lower bound of the BHPs is chosen such that the pressure in the well is high enough to push the produced fluids to the production facilities. The rate-of-change constraints of both the injection rates and BHPs are chosen such that the controller is able to change e.g. the injection rate from maximum to minimum within a predefined number of control steps.
In the following, we apply the single shooting approach for the constrained optimal control problem (5.1) to maximize the NPV of a horizontal 2-dimensional reservoir using water flooding and smart-well technology as depicted in Figure 5.2a. The permeability field of the reservoir is illustrated in Figure 5.2b [1]. The reservoir dimensions are 450 m $\times$ 450 m $\times$ 10 m and it is discretized into 45 $\times$ 45 $\times$ 1 grid blocks. One horizontal injector

Symbol	Description	Value	Unit
ϕ	Porosity	0.2	-
S_{wc}	Critical water saturation	0.20	-
S_{or}	Residual oil saturation	0.15	-
k_{rw}°	Water end point relative permeability	0.6	-
k_{ro}°	Oil end point relative permeability	0.8	-
n_w	Corey exponent, water	1.5	-
n_o	Corey exponent, oil	2.0	-
ρ_w°	Water density (at 1 atm)	1000	kg·m^{-3}
ρ_o°	Oil density (at 1 atm)	800	kg·m^{-3}
c_w	Water compressibility	10^{-5}	atm^{-1}
c_o	Oil compressibility	10^{-5}	atm^{-1}
μ_w	Water viscosity	1.0	cP
μ_o	Oil viscosity	1.0	cP
S_{init}	Initial water saturation	0.3	-
P_{init}	Initial reservoir pressure	200	atm

Table 5.1: Reservoir properties.

Symbol	Description	Value	Unit
r_o^{pro}	Oil price	283,0	\$/m^3
r_w^{pro}	Cost of water separation	31,5	\$/m^3
r_w^{inj}	Cost of water injection	5,0	\$/m^3
d	Discount rate	0	-

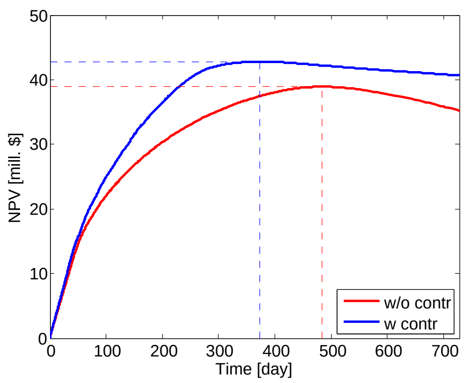
Table 5.2: Economic data [15, 26, 36].

divided into 45 individually controllable segments is positioned on the left-hand side of the reservoir (white squares, first segment at $(x, y) = (5, 5)$ m, last segment at $(x, y) = (5, 445)$ m), and one horizontal producer divided into 45 individually controllable segments is positioned on the right-hand side of the reservoir (white circles, first segment at $(x, y) = (445, 5)$ m, last segment at $(x, y) = (445, 445)$ m). In this setup we mimic horizontal

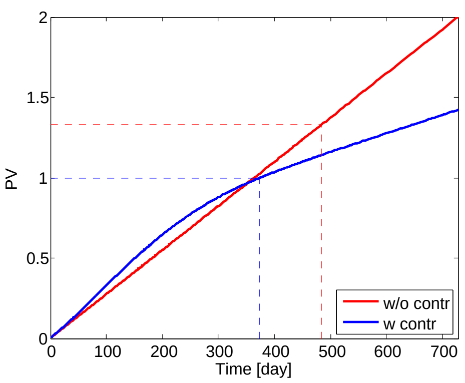
Symbol	Description	Value	Unit
q_{min}^{inj}	Min. water injection rate per injector	0	$m^3 \cdot d^{-1}$
q_{max}^{inj}	Max. water injection rate per injector	50	$m^3 \cdot d^{-1}$
P_{min}^{bh}	Min. BHP in the producers	150	atm
P_{max}^{bh}	Max. BHP in the producers	200	atm
Δq_{min}^{inj}	Min. rate of change of the water injection	-3.85	$m^3 \cdot d^{-1}$
Δq_{max}^{inj}	Max. rate of change of the water injection	3.85	$m^3 \cdot d^{-1}$
ΔP_{min}^{bh}	Min. rate of change of the BHP	-3.85	atm
ΔP_{max}^{bh}	Max. rate of change of the BHP	3.85	atm
PV_{max}	Max. amount of water allowed for injection	$4V_p$	m^3
T	Planned production period	728	d

Table 5.3: Controller settings.

wells by composing multiple vertical wells. Each vertical well is completed in adjacent grid blocks, see Section 2.5, such that a vertical well mimics a horizontal well segment. Table 5.1 lists the geological properties and the fluid properties of the reservoir. The economical data are listed in Table 5.2 [15, 26]. The discount rate is zero, $d = 0$ [36]. Table 5.3 provides the constraints of the injection rates and the BHPs as well as the maximum allowed number of PVs to be injected during the production period.
We apply two different production strategies. In the first approach, we inject 2.00 PV over a period of 728 days (2 years) using *fixed* injection rates and BHPs. Throughout the 2-year production period, all injector segments are assigned the same constant injection rate, and all producer segments are assigned the same constant BHP. In the second approach, we apply *optimized* well rates and BHPs to the injectors and the producers, respectively. Each well segment is adjusted individually by the optimizer every 28 days. As shown in Figure 5.3a, this strategy leads to an optimal production period of 374 days (1 year and 10 days) with an injection of only 1.00 PV of water in total. Figure 5.3b illustrates injected PVs as function of time. Figure 5.4a and 5.4b depicts the recovery factor (produced oil related to the initial mass of oil in the reservoir) and the water cut (produced



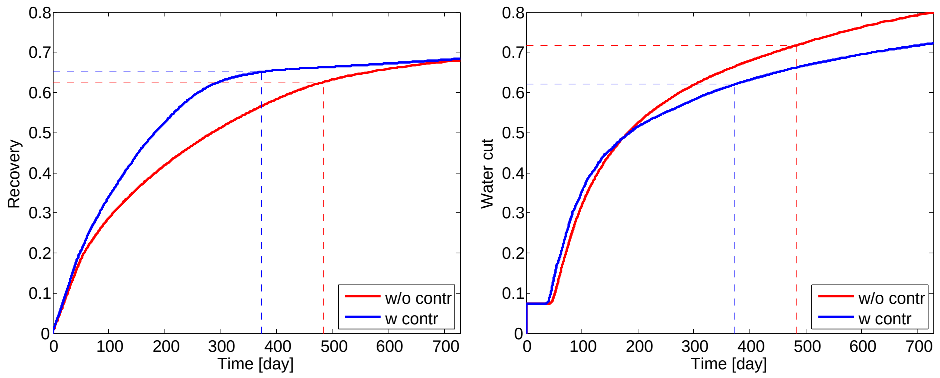
(a) NPV, 728 days of production.



(b) PVs injected, 728 days of production.

Figure 5.3: NPV and PVs injected over 728 days of production. The red dashed line represents maximum NPV using fixed rates and BHPs. The blue dashed line represents maximum NPV using optimized rates and BHPs.

oil related to the total mass of produced reservoir fluids) as function of time, respectively. Figure 5.3a shows the NPV as function of the time. Figure 5.3a shows that the optimal production period using constant well rates and pressures is 484 days (1 year and 120 days). In this period we inject 1.33 PV. By letting the optimizer control the well rates and the BHPs, we can thus reduce the production period from 484 days to 374 days. If we compare the two different production strategies, we see an increase in NPV of approximately 10% (from \$ 39 mill. to \$ 43 mill.). The recovery factor corresponding to optimal operation in the controlled case is 65%. In the uncontrolled case, the optimal recovery factor is 63%. The corresponding optimal water cuts are 62% in the controlled case and 72% in the uncontrolled case. Thus, the 10% increase in NPV for the controlled case is due to 2% increased oil recovery, a 10% decrease in produced water, and a 25% reduction in PVs of water injected (from 1.33 PV to 1.00 PV). Figure 5.5 illustrates how the optimizer administrates the injection rates and the BHPs over time for each individual well segment. Figure 5.6c and



(a) Recovery factor, 728 days of production. **(b)** Water cut, 728 days of production.

Figure 5.4: Recovery factor and water cut over 728 days of production. The red dashed line represents maximum NPV using fixed rates and BHPs. The blue dashed line represents maximum NPV using optimized rates and BHPs.

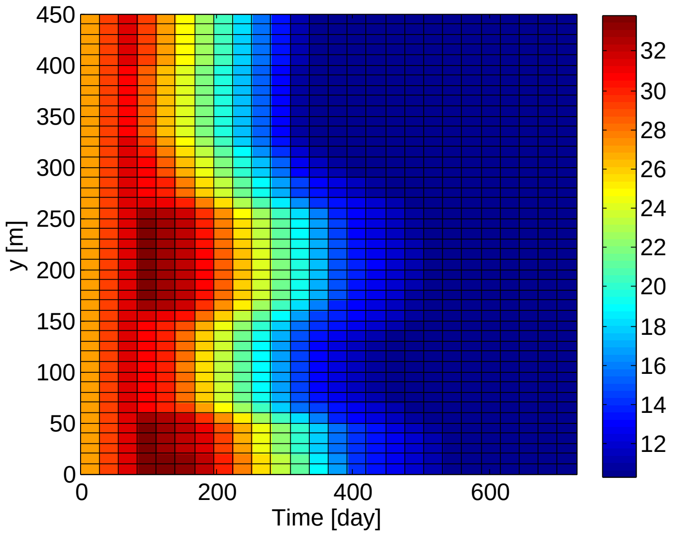
5.6d show signs of water-breakthrough at the production wells after approximately 50 days of production. In Figure 5.5a we see that the optimizer responds by reducing the injection rates in areas with high permeability and by increasing the injection rates in areas with low permeability. Figure 5.5b shows that the optimizer also responds by increasing the BHP in areas with high permeability and by decreasing the BHP in areas with low permeability. In this way, the optimizer tries to avoid water breakthrough, and thus avoiding production of injected water as this will decrease the NPV significantly. After approximately 374 days of production, the optimizer reduces the injection rates in all injector segments. High BHP is maintained in those producer segments that are positioned in areas with low permeability, and low BHP is maintained in areas with high permeability. Applying these settings minimizes both the injection and the production of water, thereby maximizing the NPV.

Figure 5.6 - 5.10 illustrates the development of the reservoir when we apply the two different production strategies. In particular, Figure 5.9e depicts

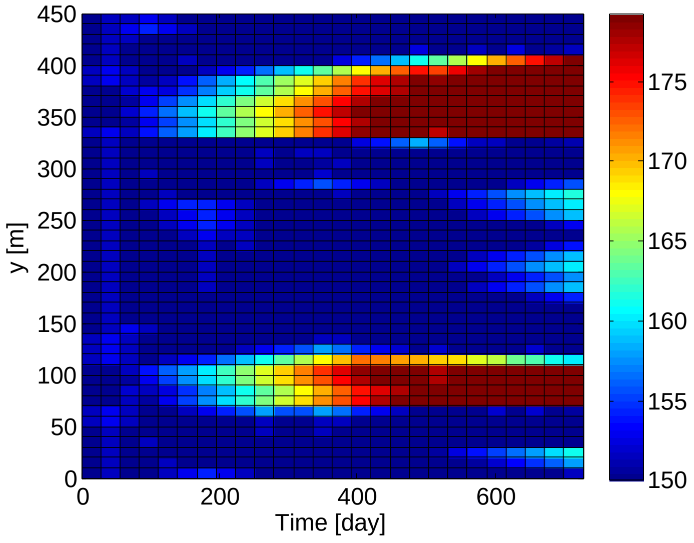
the oil saturation corresponding to maximum NPV using fixed injection rates and BHPs (represented by red dashed lines in Figure 5.3 and 5.4), and Figure 5.9b illustrates the oil saturation corresponding to the maximum NPV using optimized injection rates and BHPs (represented by blue dashed lines in Figure 5.3 and 5.4).

5.6 Summary

We have implemented a numerical method for solution of large-scale constrained optimal control problems (5.1). The implementation uses a novel representation of the system dynamics that is relevant to describe flow in porous media, see Paper A. We use an ESDIRK method for the integration along with adaptive temporal step sizes, see Paper C. The optimization is based on single-shooting, the SQP optimization algorithm with line-search and BFGS approximations of the Hessian, and the adjoint method for computation of the gradients. We use this algorithm to maximize NPV using water flooding as method for oil recovery. The developed large-scale constrained optimal control algorithm computes the optimal profiles of water injection rates and the bottom hole pressures. Compared to the uncontrolled case, the NPV in the controlled case increases by 10%. This figure demonstrates a significant economic potential of applying smart well technology along with constrained optimal control in oil reservoir management.

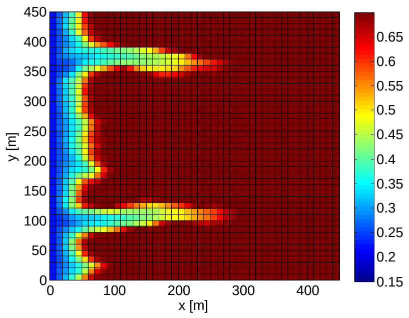


(a) Individual injection rates (m^3/day) of 45 injectors.

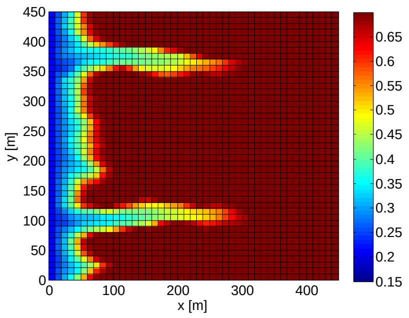


(b) Individual BHPs (atm) of 45 producers.

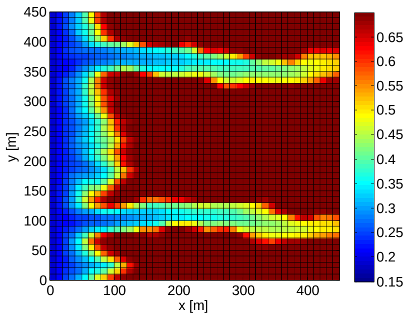
Figure 5.5: Injection rates and BHPs over 728 days of production, updated each 28 days. The injectors and producers are depicted in Figure 5.2b.



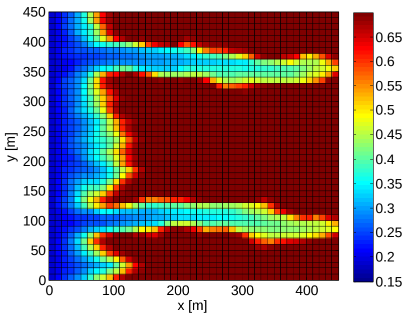
(a) 0.07 PV, 25 days, fixed.



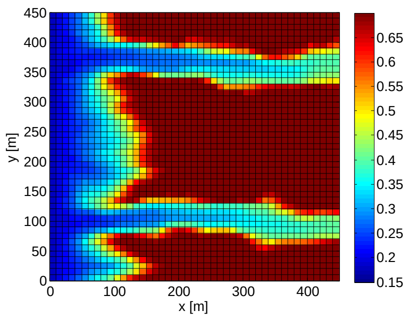
(b) 0.07 PV, 25 days, optimized.



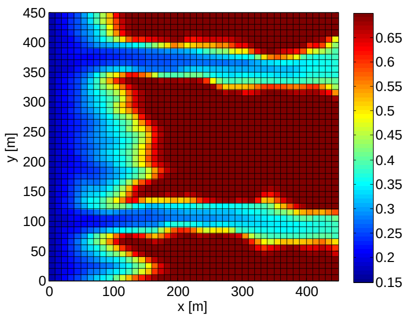
(c) 0.14 PV, 50 days, fixed.



(d) 0.14 PV, 50 days, optimized.

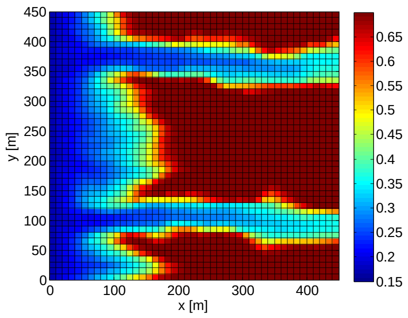


(e) 0.21 PV, 75 days, fixed.

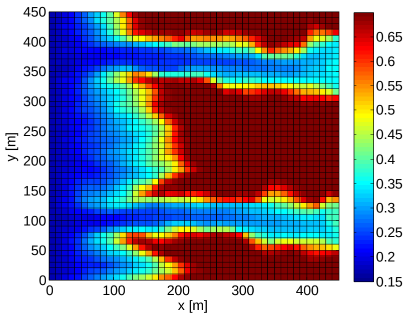


(f) 0.24 PV, 75 days, optimized.

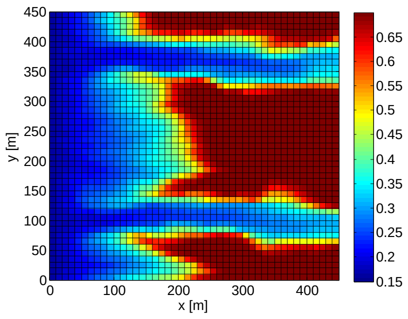
Figure 5.6: Oil saturations after 25, 50 and 75 days of production. Left column: with the use of *fixed* injection rates and BHPs. Right column: with the use of *optimized* injection rates and BHPs.



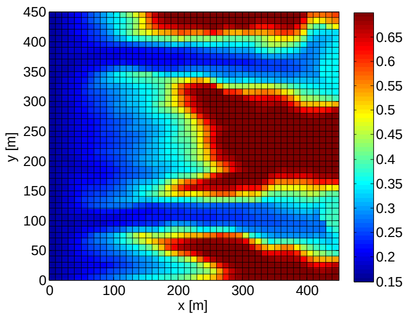
(a) 0.28 PV, 100 days, fixed.



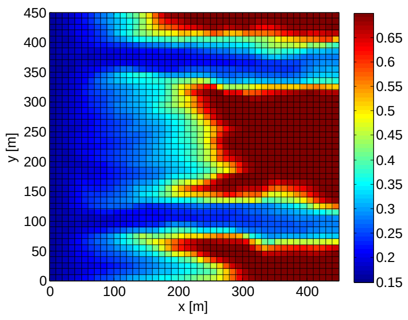
(b) 0.33 PV, 100 days, optimized.



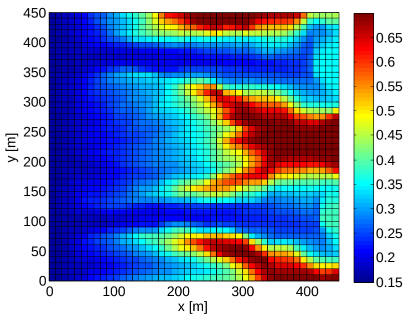
(c) 0.41 PV, 125 days, fixed.



(d) 0.50 PV, 125 days, optimized.

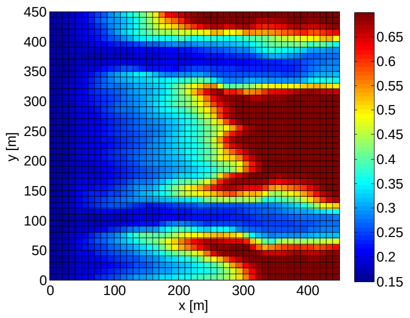


(e) 0.56 PV, 150 days, fixed.

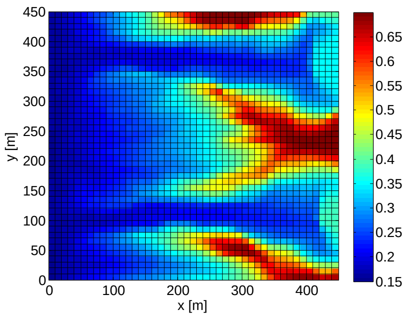


(f) 0.64 PV, 150 days, optimized.

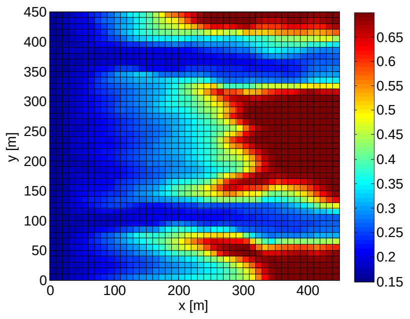
Figure 5.7: Oil saturations after 100, 150 and 200 days of production. Left column: with the use of *fixed* injection rates and BHPs. Right column: with the use of *optimized* injection rates and BHPs.



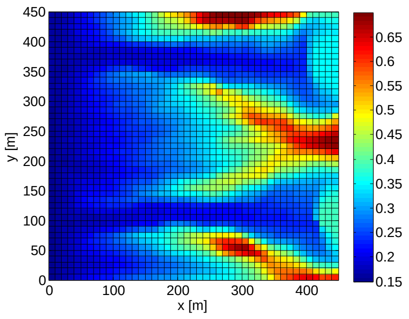
(a) 0.62 PV, 225 days, fixed.



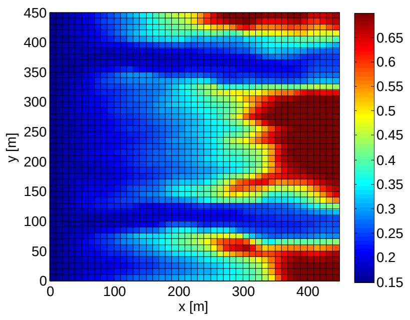
(b) 0.71 PV, 225 days, optimized.



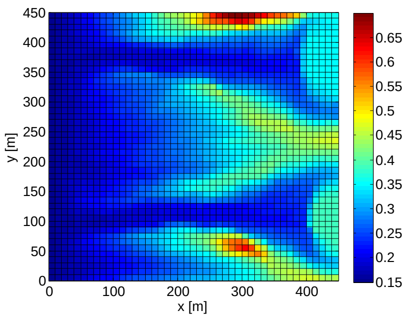
(c) 0.69 PV, 250 days, fixed.



(d) 0.77 PV, 250 days, optimized.

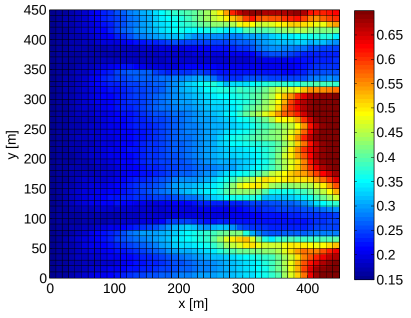


(e) 0.82 PV, 300 days, fixed.

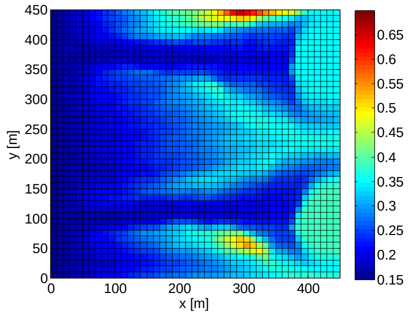


(f) 0.88 PV, 300 days, optimized.

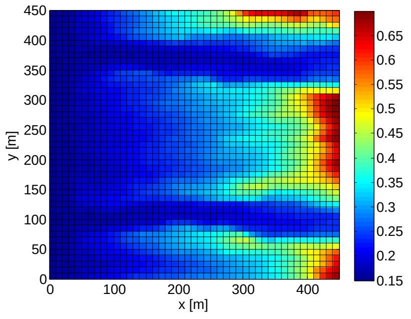
Figure 5.8: Oil saturations after 225, 250 and 300 days of production. Left column: with the use of *fixed* injection rates and BHPs. Right column: with the use of *optimized* injection rates and BHPs.



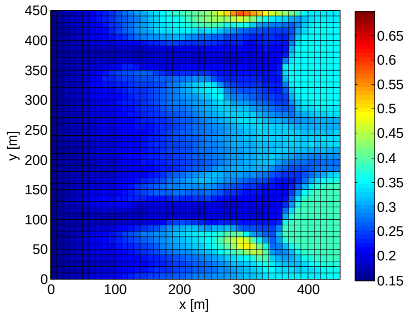
(a) 1.04 PV, 374 days, fixed.



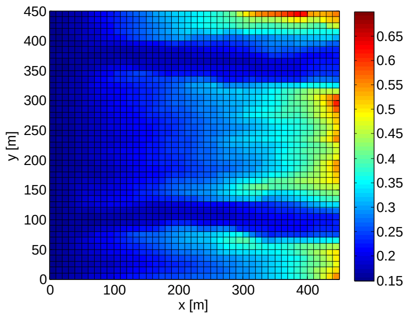
(b) 1.00 PV, 374 days, optimized.



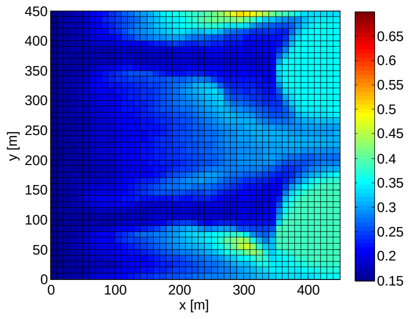
(c) 1.17 PV, 425 days, fixed.



(d) 1.07 PV, 425 days, optimized.

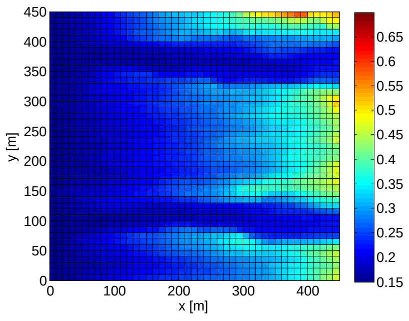


(e) 1.33 PV, 484 days, fixed.

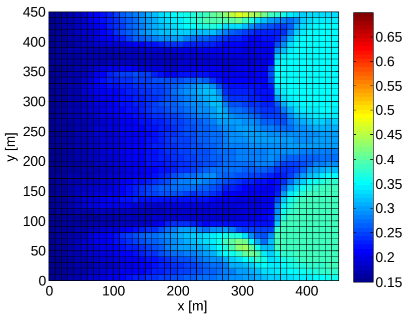


(f) 1.14 PV, 484 days, optimized.

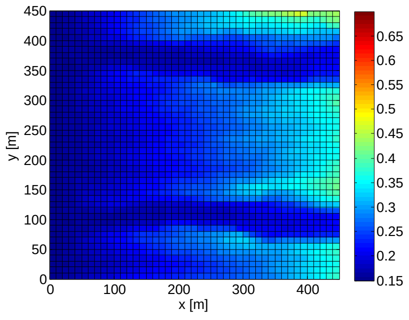
Figure 5.9: Oil saturations after 374, 425 and 484 days of production. Left column: with the use of *fixed* injection rates and BHPs. Right column: with the use of *optimized* injection rates and BHPs.



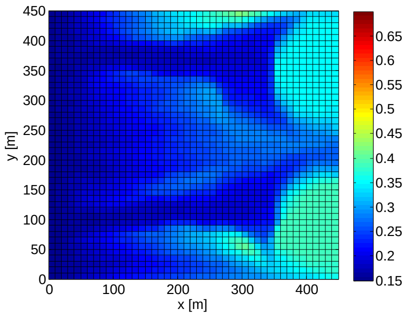
(a) 1.44 PV, 525 days, fixed.



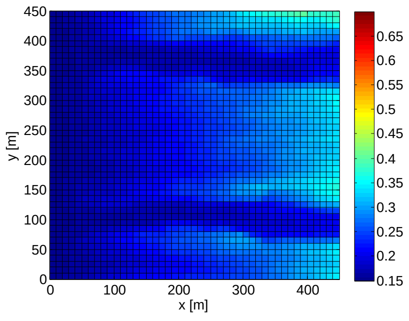
(b) 1.19 PV, 525 days, optimized.



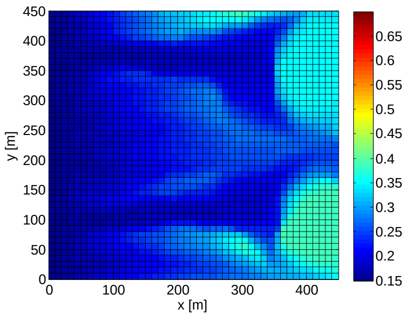
(c) 1.72 PV, 625 days, fixed.



(d) 1.30 PV, 625 days, optimized.



(e) 2.00 PV, 728 days, fixed.



(f) 1.42 PV, 728 days, optimized.

Figure 5.10: Oil saturations after 525, 625 and 728 days of production. Left column: with the use of *fixed* injection rates and BHPs. Right column: with the use of *optimized* injection rates and BHPs.

CHAPTER 6

Conclusion

The primary focus of the work in this thesis has been the application of mass conserving high order explicit singly diagonal implicit Runge-Kutta (ESDIRK) methods with embedded error estimators for step size control and the adjoint approach for gradient computation in a single-shooting framework with emphasis on optimal control of the water flooding process for oil recovery. The overall achievements of the work are summarized in this chapter.

- We have implemented the immiscible two-phase flow problem for simulation of combined flow of water and oil in isothermal reservoirs with heterogenous isotropic permeability field. The well models are based on the Peaceman well index for vertical wells in non-square Cartesian grids. Injection wells are operated at variable injection rates, whereas production wells are operated at variable bottom hole pressure (BHP).

- The fluid flow in the reservoir model is governed by a system of partial differential equations (PDEs). We use the method of lines for solution of the model. We have discretized the model spatially using the finite volume method (FVM), and we have applied the two point flux approximation (TPFA) and the single-point upstream (SPU) scheme for computing the interface fluxes of the water and oil phases. We solve the spatially discretized model using a fully implicit mass conserving high order method for numerical integration.

- We have proposed a new mass conserving formulation of the differential equation system that arise as a consequence of the spatial discretization of the two-phase flow model. The two-phase flow model is developed under the assumption of mass conservation. In general, process system models are based on conservation of mass, energy and momentum, and they can be directly represented in the form of the proposed differential equation system. Upon discretization in time, the proposed equation system ensures the preservation of such properties.

- We have presented the family of Runge-Kutta integration methods, and we have developed new mass conserving ESDIRK methods with embedded error estimators for adaptive step size control. In situations where solutions with high accuracy is required, we have demonstrated that high-order ESDIRK methods can provide a solution of the water flooding problem more efficiently than the traditional low-order methods, often referred to in the petroleum literature as: the implicit pressure explicit saturation (IMPES) method, the sequential implicit method (SIM), and the fully implicit method (FIM).

- Current reservoir simulation tools use step size control, which is based on safeguarded heuristics. These can neither deliver solutions with predetermined accuracy or guarantee the convergence in the modified Newton-Raphson (NR) iterations. We have established predictive step size control based on error estimates, which can be calculated from the embedded ESDIRK methods. The adaptive step size control monitors both the integration error of the solution and the convergence of the iterative solver. We have added some modifications to both error and convergence monitoring in order to achieve a smoother control of the integration error. In this way we minimize the computational cost per simulation of the two-phase flow problem.

- We have used nonlinear model predictive control (NMPC) to maximize the net present value (NPV) of an oil reservoir by manipulating injection rates and BHPs of injection wells and production wells, respectively. We have implemented a single-shooting method for solution of large-scale constrained optimization problems. The forward integration is done using ESDIRK methods along with adaptive temporal step sizes. The optimization is based on quasi-Newton sequential quadratic programming (SQP) with line-search and BFGS approximations of the Hessian, and the adjoint method for efficient computation of the gradients. We have demonstrated that the application of NMPC for optimal control of smart-wells has the potential to increase the economic value of an oil reservoir.

Bibliography

[1] D. Brouwer and J.-D. Jansen, "Dynamic optimization of waterflooding with smart wells using optimal control theory," *SPE Journal*, vol. 9, no. 4, pp. 391–402, 2004.

[2] M. Konopczynski, "Design and modelling of intelligent well downhole valves for adjustable flow control of smart wells," *presented at the SPE ATW Modeling and Optimization of Smart Wells*, 2005.

[3] D. Brouwer, G. Nævdal, J. Jansen, E. Vefring, and C. van Kruijsdijk, "Improved reservoir management through optimal control and continuous model updating," in *SPE Annual Technical Conference and Exhibition*, Houston, Texas, 2004.

[4] M. Handels, M. Zandvliet, R. Brouwer, and J. D. Jansen, "Adjoint-based well-placement optimization under production constraints," in *SPE Reservoir Simulation Symposium.* Houston, Texas, U.S.A.: Society of Petroleum Engineers, 2007.

[5] M. Zandvliet, "Model-based lifecycle optimization of well locations and production settings in petroleum reservoirs," Ph.D. dissertation, 2008.

[6] P. Sarma and W. H. Chen, "Applications of optimal control theory for efficient production optimisation of realistic reservoirs," in *International Petroleum Technology Conference.* Kuala Lumpur, Malaysia: International Petroleum Technology Conference, 2008.

[7] W. Chen, G. Gavalas, J. Seinfeld, and M. Wasserman, “A new algorithm for automatic history matching,” no. 12, 1974.

[8] G. CHAVENT, M. DUPUY, and P. LEMONNIER, “History matching by use of optimal theory,” no. 02, 1975.

[9] W. F. Ramirez, *Application of Optimal Control Theory to Enhanced Oil Recovery*, ser. Developments in Petroleum Science. New York, US: Elsevier Science Inc., 1987, vol. 21.

[10] D. R. Brouwer, “Dynamic water flood optimization with smart wells using optimal control theory,” Ph.D. dissertation, Delft, NL, 2004.

[11] B. Sudaryanto and Y. C. Yortsos, “Optimization of fluid front dynamics in porous media using rate control. i. equal mobility fluids,” *Physics of Fluids*, vol. 12, no. 7, pp. 1656–1670, 2000.

[12] B. Sudaryanto and Y. C. Yortsos, “Optimization of displacements in porous media using rate control,” in *SPE Annual Technical Conference and Exhibition*. New Orleans, Louisiana: Copyright 2001, Society of Petroleum Engineers Inc., 2001.

[13] P. Sarma, K. Aziz, and L. Durlofsky, “Implementation of adjoint solution for optimal control of smart wells,” in *SPE Reservoir Simulation Symposium*. The Woodlands, Texas: 2005,. Society of Petroleum Engineers Inc., 2005.

[14] P. de Montleau, A. Cominelli, K. Neylon, D. Rowan, I. Pallister, O. Tesaker, and I. Nygard, “Production optimization under constraints using adjoint gradients,” in *Proceedings of ECMOR X*. Amsterdam, NL: European Association of Geoscientists and Engineers, Sept. 2006.

[15] J. F. B. M. Kraaijevanger, P. J. Egberts, J. R. Valstar, and H. W. Buurman, “Optimal waterflood design using the adjoint method,” in *SPE Reservoir Simulation Symposium*. Houston, Texas, U.S.A.: Society of Petroleum Engineers, 2007.

[16] G. van Essen, M. Zandvliet, P. V. den Hof, O. Bosgra, and J.-D. Jansen, “Robust waterflooding optimization of multiple geological scenarios,” *SPE Journal*, no. 03, 2009.

[17] G. van Essen, J.-D. Jansen, R. Brouwer, S. G. Douma, M. Zandvliet, K. I. Rollett, and D. Harris, “Optimization of smart wells in the st. joseph field,” *SPE Reservoir Evaluation & Engineering*, no. 08, 2010.

[18] P. Sarma, W. H. Chen, L. J. Durlofsky, and K. Aziz, “Production optimization with adjoint models under nonlinear control-state path inequality constraints,” in *Intelligent Energy Conference and Exhibition.* Amsterdam, The Netherlands: Society of Petroleum Engineers, 2006.

[19] E. Suwartadi, S. Krogstad, and B. Foss, “On state constraints of adjoint optimization in oil reservoir waterflooding,” in *SPE/EAGE Reservoir Characterization and Simulation Conference.* Abu Dhabi, UAE: Society of Petroleum Engineers, 2009.

[20] E. Suwartadi, S. Krogstad, and B. Foss, “A lagrangian-barrier function for adjoint state constraints optimization of oil reservoirs water flooding,” in *Decision and Control (CDC), 2010 49th IEEE Conference on*, dec. 2010, pp. 3884 –3889.

[21] E. Suwartadi, S. Krogstad, and B. Foss, “Nonlinear output constraints handling for production optimization of oil reservoirs,” *Computational Geosciences*, pp. 1–19, 2011, 10.1007/s10596-011-9253-3.

[22] E. Suwartadi, S. Krogstad, and B. Foss, “Second-order adjoint-based control for multiphase flow in subsurface oil reservoirs,” in *Decision and Control (CDC), 2010 49th IEEE Conference on*, dec. 2010, pp. 1866 –1871.

[23] G. Naevdal, T. Mannseth, and E. H. Vefring, “Instrumented wells and near-well reservoir monitoring through ensemble kalman filter,” in *Proceedings of ECMOR VIII.* Freiberg, DE: European Association of Geoscientists and Engineers, Sept. 2002.

[24] G. Naevdal, D. Brouwer, and J.-D. Jansen, “Waterflooding using closed-loop control,” pp. 37–60, 2006-03-26.

[25] J.-D. Jansen, O. H. Bosgra, and P. M. Van den Hof, “Model-based control of multiphase flow in subsurface oil reservoirs,” *Journal of Process Control*, vol. 18, no. 9, pp. 846–855, Oct. 2008.

[26] J.-D. Jansen, R. Brouwer, and S. G. Douma, “Closed loop reservoir management,” in *SPE Reservoir Simulation Symposium.* The Woodlands, Texas: Society of Petroleum Engineers, 2009.

[27] P. Sarma, L. Durlofsky, K. Aziz, and W. Chen, “Efficient real-time reservoir management using adjoint-based optimal control and model updating,” *Computational Geosciences*, vol. 10, pp. 3–36, 2006, 10.1007/s10596-005-9009-z.

[28] C. Chen, G. Li, and A. C. Reynolds, “Robust constrained optimization of short and long-term npv for closed-loop reservoir management,” in *SPE Reservoir Simulation Symposium.* The Woodlands, Texas, USA: Society of Petroleum Engineers, 2011.

[29] L. Peters, R. Arts, G. Brouwer, C. Geel, S. Cullick, R. J. Lorentzen, Y. Chen, N. Dunlop, F. C. Vossepoel, R. Xu, P. Sarma, A. H. Alhuthali, and A. Reynolds, “Results of the brugge benchmark study for flooding optimization and history matching,” *SPE Reservoir Evaluation & Engineering*, no. 06, 2010.

[30] D. W. Peaceman, *Fundamentals of Numerical Reservoir Simulation.* Amsterdam, Holland: Developments in Petroleum Science, 1977, vol. 6.

[31] K. Aziz and A. Settari, *Petroleum Reservoir Simulation*, 1st ed. London: Applied Science Publishers Ltd, 1979.

[32] H. Tchelepi and K. Aziz, “Advances in reservoir simulation,” in *In Proceedings of the 8th International Forum on Reservoir Simulation*, Stresa, Italy, 2005.

[33] M. G. Gerritsen and L. J. Durlofsky, "Modeling fluid flow in oil reservoirs," *Annual Review of Fluid Mechanics*, vol. 37, no. 1, pp. 211–238, 2005.

[34] Z. Chen, *Reservoir Simulation : Mathematical Techniques in Oil Recovery.* Philadelphia: Society for Industrial and Applied Mathematics (SIAM), 2007.

[35] D. W. Peaceman, "Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability," *SPEJ, Society of Petroleum Engineers Journal*, vol. 23, no. 3, pp. 531–543, 1983.

[36] P. Sarma, W. H. Chen, L. J. Durlofsky, and K. Aziz, "Production optimization with adjoint models under nonlinear control-state path inequality constraints," *SPE Reservoir Evaluation & Engineering*, vol. 11, no. 2, pp. 326–339, 2008.

[37] J. D. Jansen, "Adjoint-based optimization of multi-phase flow through porous media - a review," *Computers &amp; Fluids*, vol. 46, no. 1, pp. 40 – 51, 2011, <ce:title>10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010)</ce:title>.

[38] A. T. Corey, "The interrelation between gas and oil relative permeabilities," *Producers Monthly*, vol. 19, pp. 38 – 41, 1954.

[39] R. A. Berenblyum, A. A. Shapiro, E. H. Stenby, K. Jessen, and F. M. Orr Jr., "Black oil streamline simulator with capillary effects," *Proceedings - SPE Annual Technical Conference and Exhibition*, pp. 133–141, 2003.

[40] H. P. G. Darcy, *Les Fontaines Publiques de la Ville de Dijon*, Dalmont, Paris, 1856.

[41] D. Peaceman, "Interpretation of well-block pressures in numerical reservoir simulation (includes associated paper 6988)," no. 06, 1978.

[42] J. H. Abou-Kassem and K. Aziz, “Analytical well models for reservoir simulation,” no. 08, 1985.

[43] C. Palagi and K. Aziz, “Modeling vertical and horizontal wells with voronoi grid,” *SPE Reservoir Engineering*, no. 02, 1994.

[44] J. A. Holmes, “Modeling advanced wells in reservoir simulation,” *SPE Journal of Petroleum Technology*, no. 11, 2001.

[45] C. Wolfsteiner, L. Durlofsky, and K. Aziz, “Approximate model for productivity of nonconventional wells in heterogeneous reservoirs,” *SPE Journal*, no. 06, 2000.

[46] C. Wolfsteiner, L. J. Durlofsky, and K. Aziz, “Calculation of well index for nonconventional wells on arbitrary grids,” *Computational Geosciences*, vol. 7, no. 1, pp. 61–82, 2003.

[47] C. Guichard, J. Brac, R. Eymard, and R. Masson, “Finite volume schemes for multiphase flow simulation on near well grids,” in *Proceedings of ECMOR XII.* Oxford, UK: European Association of Geoscientists and Engineers, Sept. 2010.

[48] I. Aavatsmark and R. Klausen, “Well index in reservoir simulation for slanted and slightly curved wells in 3d grids,” *SPE Journal*, no. 03, 2003.

[49] R. J. LeVeque, *Finite difference methods for ordinary and partial differential equations : Steady-State and Time-Dependent Problems.* Philadelphia, US: Society for Industrial and Applied Mathematics (SIAM), 2007.

[50] I. Aavatsmark, “Interpretation of a two-point flux stencil for skew parallelogram grids,” *Computational Geosciences*, vol. 11, pp. 199–206, 2007, 10.1007/s10596-007-9042-1.

[51] M. G. Edwards and C. F. Rogers, “Finite volume discretization with imposed flux continuity for the general tensor pressure

equation," *Computational Geosciences*, vol. 2, pp. 259–290, 1998, 10.1023/A:1011510505406.

[52] G. T. Eigestad and R. A. Klausen, "On the convergence of the multi-point flux approximation o-method: Numerical experiments for discontinuous permeability," *Numerical Methods for Partial Differential Equations*, vol. 21, no. 6, pp. 1079–1098, 2005.

[53] Y. Chen, B. Mallison, and L. Durlofsky, "Nonlinear two-point flux approximation for modeling full-tensor effects in subsurface flow simulations," *Computational Geosciences*, vol. 12, pp. 317–335, 2008, 10.1007/s10596-007-9067-5.

[54] I. Aavatsmark, "An introduction to multipoint flux approximations for quadrilateral grids," *Computational Geosciences*, vol. 6, pp. 405–432, 2002, 10.1023/A:1021291114475.

[55] Q.-Y. Chen, R. T. Mifflin, J. Wan, and Y. Yang, "A new multipoint flux approximation for reservoir simulation," in *SPE Reservoir Simulation Symposium*, Houston, Texas, U.S.A., 2007.

[56] I. Aavatsmark, G. Eigestad, B.-O. Heimsund, B. Mallison, J. Nordbotten, and E. Øian, "A new finite-volume approach to efficient discretization on challenging grids," *SPE Journal*, no. 09, 2010.

[57] E. Keilegavlen and I. Aavatsmark, "Monotonicity for control volume methods on unstructured grids," in *Proceedings of ECMOR XI*. Bergen, Norway: European Association of Geoscientists and Engineers, Sept. 2008.

[58] R. Yorgova and I. Aavatsmark, "Modification of a reservoir grid to achieve monotonicity of the control volume method," in *Proceedings of ECMOR XII*. Oxford, UK: European Association of Geoscientists and Engineers, Sept. 2010.

[59] Z. Chen, G. Huan, and Y. Ma, *Computational Methods for Multiphase Flows in Porous Media*, 2nd ed., ser. Computational science & engineering. Philadelphia: Society for Industrial and Applied Mathematics (SIAM), 2006.

[60] Y. Epshteyn and B. Riviere, "Fully implicit discontinuous finite element methods for two-phase flow," *Applied Numerical Mathematics*, vol. 57, no. 4, pp. 383 – 401, 2007.

[61] J. R. Natvig and K.-A. Lie, "Fast computation of multiphase flow in porous media by implicit discontinuous galerkin schemes with optimal ordering of elements," *Journal of Computational Physics*, vol. 227, no. 24, pp. 10 108–10 124, 2008.

[62] G. Pencheva, S. Thomas, and M. Wheeler, "Mortar coupling of discontinuous galerkin and mixed finite element methods," in *Proceedings of ECMOR XI*. Bergen, Norway: European Association of Geoscientists and Engineers, Sept. 2008.

[63] J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods : Algorithms, Analysis and Applications*. New York, USA: Springer Science+Business Media, LLC, 2008.

[64] J. D. Huppler, "Waterflood relative permeabilities in composite cores," *J. Pet. Technol.*, vol. 21, pp. 539–540, 1969.

[65] H. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics : The Finite Volume Method*, 2nd ed. Harlow, England: Pearson Education Limited, 2007.

[66] R. Alexander, "Design and implementation of dirk integrators for stiff systems," *Applied Numerical Mathematics*, vol. 46, no. 1, pp. 1 – 17, 2003.

[67] A. Kværnø, "Singly diagonally implicit runge-kutta methods with an explicit first stage," *BIT Numerical Mathematics*, vol. 44, pp. 489 – 502, 2004.

[68] M. R. Kristensen, J. B. Jørgensen, P. G. Thomsen, and S. B. Jøgensen, "An esdirk method with sensitivity analysis capabilities," *Computers &amp; Chemical Engineering*, vol. 28, no. 12, pp. 2695 – 2707, 2004.

[69] J. B. Jørgensen, M. R. Kristensen, and P. G. Thomsen, "A family of esdirk integration methods," *manuscript for SIAM Journal on Scientific Computing*, 2008.

[70] J. Sheldon, B. Zondek, and W. C. Jr., "One-dimensional, incompressible, noncapillary, two-phase fluid flow in a porous medium," 1959.

[71] H. Stone and A. G. Jr., "Analysis of gas-cap or dissolved-gas drive reservoirs," no. 06, 1961.

[72] K. Coats, "A note on impes and some impes-based simulation models," *SPE Journal*, vol. 5, no. 3, pp. 245–251, 2000.

[73] A. Spillette, J. Hillestad, and H. Stone, "A high-stability sequential solution approach to reservoir simulation," in *Fall Meeting of the Society of Petroleum Engineers of AIME*, Las Vegas, Nevada, 1973.

[74] J. Watts, "A compositional formulation of the pressure and saturation equations," *SPE Reservoir Engineering*, no. 05, 1986.

[75] J. Grabowski, P. Vinsome, R. C. Lin, G. Behie, and B. Rubin, "A fully implicit general purpose finite-difference thermal model for in situ combustion and steam," in *SPE Annual Technical Conference and Exhibition*, Las Vegas, Nevada, 1979.

[76] K. H. Coats, "In-situ combustion model," no. 12, 1980.

[77] R. Mehra, M. Hadjitofi, and J. K. Donnelly, "An automatic time-step selector for reservoir models," in *SPE Reservoir Simulation Symposium*, New Orleans, Louisiana, 1982.

[78] P. Sammon and B. Rubin, "Practical control of timestep selection in thermal simulation," *SPE Reservoir Engineering*, no. 03, 1986.

[79] C. A. Kennedy and M. H. Carpenter, "Additive runge-kutta schemes for convection-diffusion-reaction equations," *Applied Numerical Mathematics*, vol. 44, no. 1-2, pp. 139 – 181, 2003.

[80] K. Gustafsson, "Control of error and convergence in ODE solvers," Ph.D. dissertation, Department of Automatic Control, Lund University, Sweden, Mar. 1992.

[81] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, 2nd ed. Springer, 1996.

[82] R. Alexander, "The modified newton method in the solution of stiff ordinary differential equations," *Mathematics of Computation*, vol. 57, no. 196, pp. 673–701, 1991.

[83] N. Houbak, S. Nörsett, and P. Thomsen, "Displacement or residual test in the application of implicit methods for stiff problems," *IMA Journal of Numerical Analysis*, vol. 5, no. 3, pp. 297–305, 1985.

[84] K. Gustafsson and G. Söderlind, "Control strategies for the iterative solution of nonlinear equations in ode solvers," *SIAM J. Sci. Comput.*, vol. 18, no. 1, pp. 23–40, 1997.

[85] G. Söderlind, "*Folklore and fudge factors in the implementation of fixed-point and Newton iterations for nonlinear ODEs*," Talk presented at "Workshop on the Numerical Solution of Stiff ODEs", Sept. 15-17, NTH, Trondheim, Norway, 1986.

[86] D. Kraft, "On converting optimal control problems into nonlinear programming problems," in *Computational mathematical programming (Bad Windsheim, 1984)*, ser. NATO Adv. Sci. Inst. Ser. F Comput. Systems Sci. Berlin: Springer, 1985, vol. 15, pp. 261–280.

[87] M. Schlegel, K. Stockmann, T. Binder, and W. Marquardt, "Dynamic optimization using adaptive control vector parameterization," *Computers &amp; Chemical Engineering*, vol. 29, no. 8, pp. 1731 – 1751, 2005.

[88] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," in *Proceedings of 9th IFAC World Congress*. Budapest, Hungary: Pergamon Press, 1984, pp. 243–247.

[89] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder, "An efficient multiple shooting based reduced sqp strategy for large-scale dynamic process optimization. part 1: theoretical aspects," *Computers &amp; Chemical Engineering*, vol. 27, no. 2, pp. 157 – 166, 2003.

[90] L. T. and Biegler, "An overview of simultaneous strategies for dynamic optimization," *Chemical Engineering and Processing: Process Intensification*, vol. 46, no. 11, pp. 1043 – 1053, 2007, <ce:title>Special Issue on Process Optimization and Control in Chemical Engineering and Processing</ce:title>.

[91] J. T. Betts, *Practical methods for optimal control using nonlinear programming*, ser. Advances in Design and Control. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2001, vol. 3.

[92] L. T. Biegler, *Nonlinear programming*, ser. MOS-SIAM Series on Optimization. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2010, vol. 10, concepts, algorithms, and applications to chemical processes.

[93] J. B. Jørgensen, "Adjoint sensitivity results for predictive control, state- and parameter-estimation with nonlinear models," in *European Control Conference 2007*, 2007.

[94] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, USA: Springer Series in Operations Research and Financial Engineering, 2006.

[95] M. Powell, "A fast algorithm for nonlinearly constrained optimization calculations," in *Numerical Analysis*, ser. Lecture Notes in Mathematics, G. Watson, Ed. Springer Berlin / Heidelberg, 1978, vol. 630, pp. 144–157, 10.1007/BFb0067703.

Appendices

APPENDIX A

Derivatives of the Flux Term

The following sections describe the elements in (3.38), which are derivatives of the flux term (3.32) with respect to pressure and saturation. The first section deals with the derivatives of the oil and water fluxes into grid-block $i \in \mathcal{N}$ with respect to pressure, P_i, and saturation, S_i, in the block. The second section deals with the derivatives of the oil and water fluxes into the i^{th} grid-block with respect to pressure, P_j, and saturation, S_j, in the neighbouring blocks, $j \in \mathcal{N}^{(i)}$.

A.1 Derivatives of the Flux Term $F_{\alpha,i}$ with respect to P_i and S_i

The derivative of the flux term $F_{\alpha,i}$ with respect to P_i is

$$\begin{aligned}
\frac{\partial F_{\alpha,i}}{\partial P_i} &= \sum_{j\in\mathcal{N}^{(i)}} \frac{\partial(\Upsilon_\alpha \Delta P)_{ij}}{\partial P_i} \\
&= \sum_{j\in\mathcal{N}^{(i)}} \left(\frac{\partial \Upsilon_{\alpha,ij}}{\partial P_i}\Delta P_{ij} + \Upsilon_{\alpha,ij} \underbrace{\frac{\partial \Delta P_{ij}}{\partial P_i}}_{=-1} \right) \\
&= \sum_{j\in\mathcal{N}^{(i)}} \Gamma_{ij} \begin{cases} \frac{\partial \mathrm{H}_{\alpha,i}}{\partial P_i}\Delta P_{ij} - \mathrm{H}_{\alpha,i} & \Delta P_{ij} < 0 \\ \underbrace{\frac{\partial \mathrm{H}_{\alpha,j}}{\partial P_i}}_{=0}\Delta P_{ij} - \mathrm{H}_{\alpha,j} & \Delta P_{ij} > 0 \end{cases} \qquad i \in \mathcal{N} \qquad \text{(A.1)} \\
&= \sum_{j\in\mathcal{N}^{(i)}} \Gamma_{ij} \begin{cases} -\mathrm{H}_{\alpha,i}(1 - c_\alpha \Delta P_{ij}) & \Delta P_{ij} < 0 \\ -\mathrm{H}_{\alpha,j} & \Delta P_{ij} > 0 \end{cases}
\end{aligned}$$

The derivative of the flux term $F_{\alpha,i}$ with respect to S_i is

$$\begin{aligned}
\frac{\partial F_{\alpha,i}}{\partial S_i} &= \sum_{j\in\mathcal{N}^{(i)}} \frac{\partial(\Upsilon_\alpha \Delta P)_{ij}}{\partial S_i} \\
&= \sum_{j\in\mathcal{N}^{(i)}} \left(\frac{\partial \Upsilon_{\alpha,ij}}{\partial S_i}\Delta P_{ij} + \Upsilon_{\alpha,ij} \underbrace{\frac{\partial \Delta P_{ij}}{\partial S_i}}_{=0} \right) \\
&= \sum_{j\in\mathcal{N}^{(i)}} \Gamma_{ij} \begin{cases} \frac{\partial \mathrm{H}_{\alpha,i}}{\partial S_i}\Delta P_{ij} & \Delta P_{ij} < 0 \\ \underbrace{\frac{\partial \mathrm{H}_{\alpha,j}}{\partial S_i}}_{=0}\Delta P_{ij} & \Delta P_{ij} > 0 \end{cases} \qquad i \in \mathcal{N} \qquad \text{(A.2)} \\
&= \sum_{j\in\mathcal{N}^{(i)}} \Gamma_{ij} \begin{cases} \mathrm{H}_{\alpha,i}\sigma_{\alpha,i}\Delta P_{ij} & \Delta P_{ij} < 0 \\ 0 & \Delta P_{ij} > 0 \end{cases}
\end{aligned}$$

where we have defined

$$\begin{aligned}\sigma_{\alpha,i} &= \frac{n_\alpha}{s_{\alpha,i}} \frac{\partial s_{\alpha,i}}{\partial S_i} \\ &= \frac{n_\alpha}{s_{\alpha,i}} \begin{cases} \frac{1}{1-S_{wc}-S_{or}} & \alpha = w \\ \frac{-1}{1-S_{wc}-S_{or}} & \alpha = o \end{cases} \qquad i \in \mathcal{N}\end{aligned} \tag{A.3}$$

A.2 Derivatives of the Flux Term $F_{\alpha,i}$ with respect to P_j and S_j

The derivative of the flux term $F_{\alpha,i}$ with respect to P_j is

$$\begin{aligned}\frac{\partial F_{\alpha,i}}{\partial P_j} &= \sum_{k\in\mathcal{N}^{(i)}} \frac{\partial(\Upsilon_\alpha \Delta P)_{ik}}{\partial P_j} \\ &= \sum_{k\in\mathcal{N}^{(i)}\setminus\{j\}} \underbrace{\frac{\partial(\Upsilon_\alpha \Delta P)_{ik}}{\partial P_j}}_{=0} + \frac{\partial(\Upsilon_\alpha \Delta P)_{ij}}{\partial P_j} \\ &= \frac{\partial \Upsilon_{\alpha,ij}}{\partial P_j}\Delta P_{ij} + \Upsilon_{\alpha,ij} \underbrace{\frac{\partial \Delta P_{ij}}{\partial P_j}}_{=1} \qquad i \in \mathcal{N} \quad j \in \mathcal{N}^{(i)} \\ &= \Gamma_{ij} \begin{cases} \underbrace{\frac{\partial \mathrm{H}_{\alpha,i}}{\partial P_j}}_{=0} \Delta P_{ij} + \mathrm{H}_{\alpha,i} & \Delta P_{ij} < 0 \\ \frac{\partial \mathrm{H}_{\alpha,j}}{\partial P_j} \Delta P_{ij} + \mathrm{H}_{\alpha,j} & \Delta P_{ij} > 0 \end{cases} \\ &= \Gamma_{ij} \begin{cases} \mathrm{H}_{\alpha,i} & \Delta P_{ij} < 0 \\ \mathrm{H}_{\alpha,i}(c_\alpha \Delta P_{ij} + 1) & \Delta P_{ij} > 0 \end{cases}\end{aligned} \tag{A.4}$$

The derivative of the flux term $F_{\alpha,i}$ with respect to S_j is

$$\begin{aligned}
\frac{\partial F_{\alpha,i}}{\partial S_j} &= \sum_{k\in\mathcal{N}^{(i)}} \frac{\partial(\Upsilon_\alpha \Delta P)_{ik}}{\partial S_j} \\
&= \sum_{k\in\mathcal{N}^{(i)}\setminus\{j\}} \underbrace{\frac{\partial(\Upsilon_\alpha \Delta P)_{ik}}{\partial S_j}}_{=0} + \frac{\partial(\Upsilon_\alpha \Delta P)_{ij}}{\partial S_j} \\
&= \frac{\partial \Upsilon_{\alpha,ij}}{\partial S_j}\Delta P_{ij} + \Upsilon_{\alpha,ij} \underbrace{\frac{\partial \Delta P_{ij}}{\partial S_j}}_{=0} \qquad\qquad i\in\mathcal{N} \quad j\in\mathcal{N}^{(i)} \qquad \text{(A.5)} \\
&= \Gamma_{ij}\begin{cases} \underbrace{\frac{\partial \mathrm{H}_{\alpha,i}}{\partial S_j}}_{=0}\Delta P_{ij} & \Delta P_{ij} < 0 \\ \frac{\partial \mathrm{H}_{\alpha,j}}{\partial S_j}\Delta P_{ij} & \Delta P_{ij} > 0 \end{cases} \\
&= \Gamma_{ij}\begin{cases} 0 & \Delta P_{ij} < 0 \\ \mathrm{H}_{\alpha,j}\sigma_{\alpha,j}\Delta P_{ij} & \Delta P_{ij} > 0 \end{cases}
\end{aligned}$$

where we have defined

$$\begin{aligned}
\sigma_{\alpha,j} &= \frac{n_\alpha}{s_{\alpha,j}}\frac{\partial s_{\alpha,i}}{\partial S_j} \\
&= \frac{n_\alpha}{s_{\alpha,j}}\begin{cases} \frac{1}{1-S_{wc}-S_{or}} & \alpha = w \\ \frac{-1}{1-S_{wc}-S_{or}} & \alpha = o \end{cases} \qquad j\in\mathcal{N}^{(i)} \qquad \text{(A.6)}
\end{aligned}$$

APPENDIX B

ESDIRK Coefficients

Starting from left to right, the Butcher tableau's for ESDIRK12, ESDIRK23 and ESDIRK34 are as follows

$$
\begin{array}{c|cc}
0 & 0 & \\
1 & 0 & 1 \\
\hline
\mathbf{x}_{n+1} & 0 & 1 \\
\hline
\mathbf{e}_{n+1} & -\frac{1}{2} & \frac{1}{2}
\end{array}
\qquad
\begin{array}{c|ccc}
0 & 0 & & \\
c_2 & a_{21} & \gamma & \\
1 & b_1 & b_2 & \gamma \\
\hline
\mathbf{x}_{n+1} & b_1 & b_2 & \gamma \\
\hline
\mathbf{e}_{n+1} & d_1 & d_2 & d_3
\end{array}
\qquad
\begin{array}{c|cccc}
0 & 0 & & & \\
c_2 & a_{21} & \gamma & & \\
c_3 & a_{31} & a_{32} & \gamma & \\
1 & b_1 & b_2 & b_3 & \gamma \\
\hline
\mathbf{x}_{n+1} & b_1 & b_2 & b_3 & \gamma \\
\hline
\mathbf{e}_{n+1} & d_1 & d_2 & d_3 & d_4
\end{array}
$$

The coefficients for ESDIRK12 are found in the Butcher tableau. The coefficients for ESDIRK23 and ESDIRK34 are found in Table B.1 and B.2 respectively.

Coefficient	Value
c_2	0.585786437626905
a_{21}	0.292893218813452
γ	0.292893218813452
b_1	0.353553390593274
b_2	0.353553390593274
d_1	0.138071187457698
d_2	-0.333333333333333
d_3	0.195262145875635

Table B.1: ESDIRK23 coefficients [69].

Coefficient	Value
c_2	0.871733043016918
c_3	0.468238744851844
a_{21}	0.435866521508459
a_{31}	0.140737774724706
a_{32}	-0.108365551381321
γ	0.435866521508459
b_1	0.102399400619911
b_2	-0.376878452255556
b_3	0.838612530127186
d_1	-0.054625497240414
d_2	-0.494208893625995
d_3	0.221934499735065
d_3	0.326899891131344

Table B.2: ESDIRK34 coefficients [69].

APPENDIX C

Nomenclature

C.1 Physical Quantities

Symbol	Description	SI Units	Mixed Units
C_α	Mass concentration of phase α	$\mathrm{kg \cdot m^{-3}}$	$\mathrm{kg \cdot m^{-3}}$
c_α	Compressibility of phase α	$\mathrm{Pa^{-1}}$	$\mathrm{atm^{-1}}$
g	Gravitational acceleration	$\mathrm{m \cdot s^{-2}}$	$\mathrm{m \cdot d^{-2}}$
$\mathbf{F}_\alpha$	Flux of phase α	$\mathrm{kg \cdot m^{-2} \cdot s^{-1}}$	$\mathrm{kg \cdot m^{-2} \cdot d^{-1}}$
h	Well height	m	m
$\mathbf{k}$	Absolute permeability (tensor)	$\mathrm{m^2}$	mDa
k	Absolute permeability (isotropic)	$\mathrm{m^2}$	mDa
k_{xx}	Absolute permeability in x-direction (anisotropic)	$\mathrm{m^2}$	mDa
k_{yy}	Absolute permeability in y-direction (anisotropic)	$\mathrm{m^2}$	mDa
k_{zz}	Absolute permeability in z-direction (anisotropic)	$\mathrm{m^2}$	mDa
$k_{r\alpha}$	Relative permeability of phase α	-	-

Symbol	Description	SI Units	Mixed Units
k°_{rw}	Water end point relative permeability ($k_{rw} = k^{\circ}_{rw}$ when $1 - S_{or} \leq S_w \leq 1$)	-	-
k°_{ro}	Oil end point relative permeability ($k_{ro} = k^{\circ}_{ro}$ when $1 - S_{wc} \leq S_o \leq 1$)	-	-
n_α	Corey exponent of phase α	-	-
P_α	Pressure of phase α	Pa	atm
P°_α	Reference pressure of phase α with density ρ°_α	Pa	atm
P^{bh}	Bottom hole pressure in producer	Pa	atm
P_{cow}	Capillary pressure due to interfacial tension between the oil phase and the water phase	Pa	atm
Q_α	Source/sink terms of phase α due to injectors/producers	$\mathrm{kg{\cdot}m^{-3}{\cdot}s^{-1}}$	$\mathrm{kg \cdot m^{-3} \cdot d^{-1}}$
Q^{inj}_α	Source term (injector) of phase α	$\mathrm{kg{\cdot}m^{-3}{\cdot}s^{-1}}$	$\mathrm{kg \cdot m^{-3} \cdot d^{-1}}$
Q^{pro}_α	Sink term (producer) of phase α	$\mathrm{kg{\cdot}m^{-3}{\cdot}s^{-1}}$	$\mathrm{kg \cdot m^{-3} \cdot d^{-1}}$
q^{inj}	Volumetric water injection rate	$\mathrm{m^3 \cdot s^{-1}}$	$\mathrm{m^3 \cdot d^{-1}}$
q^{inj}_α	Mass injection rate of phase α	$\mathrm{kg \cdot s^{-1}}$	$\mathrm{kg \cdot d^{-1}}$
q^{pro}_α	Mass production rate of phase α	$\mathrm{kg \cdot s^{-1}}$	$\mathrm{kg \cdot d^{-1}}$
r_e	Equivalent radius (radial distance from the well at which the pressure obtained by the reservoir model and the pressure obtained by the well model are equivalent)	m	m
r_w	Well bore radius	m	m
S_α	Saturation of phase α	-	-
S_{wc}	Critical water saturation (water does not to flow when $0 \leq S_w \leq S_{wc}$)	-	-

Symbol	Description	SI Units	Mixed Units
S_{or}	Residual oil saturation (oil does not to flow when $0 \leq S_o \leq S_{or}$)	-	-
s	Skin factor of well	-	-
$\mathbf{s}$	Position in the spatial domain Ω	m	m
s_α	Normalized saturation of phase α	-	-
t	Time	s	d
T	Temporal domain	s	d
$\mathbf{u}_\alpha$	Flow velocity of phase α (Darcy's law)	$\mathrm{m} \cdot \mathrm{s}^{-1}$	$\mathrm{m} \cdot \mathrm{d}^{-1}$
V	Total bulk volume	m^3	m^3
V_p	Pore volume (PV)	m^3	m^3
WI	Well index	m^3	m^3
Δx	Grid block length in x-direction	m	m
Δy	Grid block length in y-direction	m	m
α	Phase index ($\alpha \in \{w, o\}$)	-	-
λ_α	Mobility of phase α (defined as $k_{r\alpha}/\mu_\alpha$)	$\mathrm{Pa}^{-1} \cdot \mathrm{s}^{-1}$	cP^{-1}
μ_α	Viscosity of phase α	$\mathrm{Pa} \cdot \mathrm{s}$	cP
Ω	Spatial domain	m	m
Φ_α	Flow potential of phase α	$\mathrm{Pa} \cdot \mathrm{m}^{-1}$	$\mathrm{atm} \cdot \mathrm{m}^{-1}$
ϕ	Porosity of the reservoir rock (defined as V_p/V)	-	-
ρ_α	Density of phase α	$\mathrm{kg} \cdot \mathrm{m}^{-3}$	$\mathrm{kg} \cdot \mathrm{m}^{-3}$
ρ_α°	Density of phase α at reference pressure P_α°	$\mathrm{kg} \cdot \mathrm{m}^{-3}$	$\mathrm{kg} \cdot \mathrm{m}^{-3}$
θ	Radial angle of well open to flow	-	-

C.2 Commonly used Abbreviations

Abbreviation	Description
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BHP	Bottom hole pressure
DG-FEM	Discontinuous Galerkin finite element method
DIRK	Diagonally implicit Runge-Kutta
EnKF	Ensemble Kalman filter
EOR	Enhanced oil recovery
ERK	Explicit Runge-Kutta
ESDIRK	Explicit singly diagonally implicit Runge-Kutta
ESDIRK$k\hat{k}$	Explicit singly diagonally implicit Runge-Kutta method of order k with an embedded method of order $\hat{k}$ (ESDIRK12, ESDIRK23 and ESDIRK34), see page 60
FDM	Finite difference method
FEM	Finite element method
FIM	Fully implicit method
FIRK	Fully implicit Runge-Kutta
FVM	Finite volume method
IMEX	Implicit-explicit
IMPES	Implicit pressure explicit saturation
MPFA	Multi-point flux approximation
NMPC	Nonlinear model predictive control
NPV	Net present value
NR	Newton-Raphson
ODE	Ordinary differential equation
PDE	Partial differential equation
PI	Proportional-integral
PI97	Proportional-integral step size controller (original), see page 71
PI10	Proportional-integral step size controller (modified), see page 71

Abbreviation	Description
PV	Pore volume V_p (void space of the reservoir rock containing the reservoir fluids)
SD	Sigificant digit
SDIRK	Singly diagonally implicit Runge-Kutta
SIM	Sequential implicit method
SPU	Single-point upstream
SQP	Sequential quadratic programming
TPFA	Two-point flux approximation
WI	Well index

APPENDIX D

Paper A

Simulation of Subsurface Two-Phase Flow
in an Oil Reservoir

Authors:
Carsten Völcker, John Bagterp Jørgensen, Per Grove Thomsen and
Erling Halfdan Stenby

Published in:
Proceedings of *European Control Conference 2009 (ECC'09)*
Budapest, Hungary
23-26 August 2009

Simulation of Subsurface Two-Phase Flow in an Oil Reservoir

Carsten Völcker*, John Bagterp Jørgensen*‡, Per Grove Thomsen*, Erling H. Stenby†

***Abstract*— Off-shore subsurface oil fields are porous rocks with oil trapped in the pores. Conventional technologies for recovery of this oil in porous rocks leave more than 50% of the oil in the reservoir. Wells with adjustable downhole flow control devices coupled with modern control technology offer the potential to increase the oil recovery significantly. The valve settings could be computed by solution of a large scale constrained optimal control problem implemented in a receding horizon fashion. The major computational effort in this optimal control problem concerns solution of a very large system of differential equations describing the flow of oil and water in the porous rock. We present a two-phase immiscible flow model for the oil reservoir and describe a new explicit singly diagonally implicit Runge-Kutta (ESDIRK) method for computationally efficient solution of this model. The ESDIRK integrator is mass preserving, of high order, and equipped with integration error controllers. The ESDIRK methods are computationally competitive to the implicit Euler method normally used for solution of the oil reservoir two phase immiscible flow problem.**

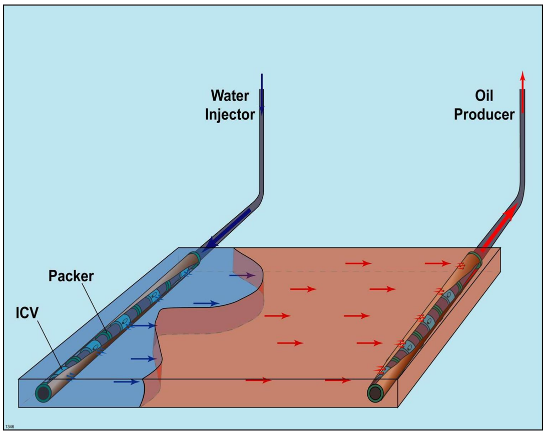


Fig. 1. An oil field with adjustable valves at the down-hole pipes [3].

I. Introduction

As the number of newly discovered major oil fields decrease, efficient exploration of already discovered oil fields becomes increasingly important. Automatic optimization and control of the operation of the oil recovery process constitute an important technology for increasing the oil recovery efficiency [1], [2].

Off-shore subsurface oil fields are porous rocks with oil in the capillaries of the rocks. Exploration of such oil reservoirs is typically conducted in three recovery phases. During *primary recovery* just after drilling, the pressure in the oil reservoir is so high that the oil is produced under the natural pressure in the reservoir. Primary recovery leaves about 70-85% of the hydrocarbons in the reservoir. To continue recovery of the remaining oil, water is injected at injection wells and oil is recovered at production wells. Water injection maintains high reservoir pressure and flow rates. As illustrated in Figure 1, it displaces the oil and pushes it toward the production wells. This part of the recovery is called *secondary recovery* or water flooding. If the reservoir pressure is above the bubble point pressure of the oil phase, secondary recovery occurs by *two-phase immiscible flow*. One phase is water and the other phase is oil. No mass transfer occurs between the phases. If the reservoir pressure drops below the bubble point pressure, the oil phase split into a liquid and a vapor phase in thermodynamic equilibrium. The water phase does not exchange mass with the other phases, but the hydrocarbon liquid and vapor phases exchange mass such that they are in thermodynamic equilibrium. More than 50% of the hydrocarbons remain in the reservoir after conventional water flooding. Strong surface tension which traps oil in small pores as well as high viscosity of the oil are some factors responsible for this low recovery. Another factor is the heterogeneity of the porous rock structure which imply different permeability in different locations of the reservoir such that large pockets of oil may be left by conventional uncontrolled water flooding. *Tertiary recovery* includes chemical flooding, steam flooding and in-situ combustion. It aims at recovering the remaining oil using chemical and thermal effects that reduce the surface tension and adhesion of oil to the rocks as well as the viscosity of the oil. These techniques are also called enhanced oil recovery.

Optimal control has been used for controlling two-phase immiscible subsurface porous flow using adjustable control valves at the down-hole pipes [3]–[9] as well as for enhanced oil recovery [10]–[12]. The constrained optimal control problem is usually solved by a gradient based optimization method. The gradients are computed from the adjoint equations of the optimal control problem. Several methods have been used for spatial discretization of the partial differential equation system, while temporal discretization is usually based on some variants of the implicit Euler method [13]–[15]. The implicit Euler method is a first-order method in the step size.

*Department of Informatics and Mathematical Modeling, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark {cv,jbj,pgt}@imm.dtu.dk, ‡ Corresponding author. †Department of Chemical and Biochemical Engineering, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark {ehs}@kt.dtu.dk. This research project is financially supported by the Danish Research Council for Technology and Production Sciences. FTP Grant no. 274-06-0284.

There is evidence, that high-order integration methods have the potential to produce faster computation when applied for temporal discretization of models describing multi-phase flows in porous media [16]. In this paper we develop a model for two-phase immiscible flow in a porous medium. A finite volume method is used for spatial discretization of the resulting system of partial differential equations. An explicit singly diagonally implicit Runge-Kutta (ESDIRK) method is used for temporal discretization. Newly developed ESDIRK methods with mass preserving properties, error estimators and automatic step size controllers are presented. The efficiency of the methods are demonstrated on simulation examples.

II. 1-Dimensional Model

In this Section we describe a one-dimensional model for two-phase immiscible flow in a porous medium. This model is important in its own right but also important for numerical studies of solution procedures [17], [18]. The two phases are oil and water. The model consists of two partial differential equations representing conservation of mass. Mass is transported by convection at a velocity determined by Darcy's law. Relative permeabilites are determined by a Corey expression and we assume zero capillary pressure. The fluid densities are described by an equation of state relating the densities to pressure.

A. Conservation Equations and Boundary Conditions

Consider the one-dimensional spatial domain $\Omega = \{x \in \mathbb{R} : 0 \leq x \leq L\}$ and the time domain $T = \{t \in \mathbb{R} : t \geq 0\}$. The boundary of the spatial domain is $\delta\Omega = \{x \in \mathbb{R} : x = 0 \wedge x = L\}$ and the interior of the spatial domain is $\Omega^o = \{x \in \mathbb{R} : 0 < x < L\}$. The mass of oil and water is conserved. Similarly $\delta T = \{t \in \mathbb{R} : t = 0\}$ and $T^o = \{t \in \mathbb{R} : t > 0\}$.

We consider a two-phase flow of oil and water with complete immiscibility. Let $C_w = C_w(t,x)$ and $C_o = C_o(t,x)$ be the concentrations of water and oil (kg water/oil per volume reservoir) in the reservoir as function of time $t \in T$ and position $x \in \Omega$. The mass balances for water and oil in the reservoir is expressed by the following system of partial differential equations

$$\frac{\partial C_w}{\partial t} = -\frac{\partial N_w}{\partial x} + q_w \quad t \in T^o,\, x \in \Omega^o \tag{1a}$$
$$\frac{\partial C_o}{\partial t} = -\frac{\partial N_o}{\partial x} + q_o \quad t \in T^o,\, x \in \Omega^o \tag{1b}$$

$N_w = N_w(t,x)$ and $N_o = N_o(t,x)$ are the fluxes of water and oil through the porous medium. The source/sink terms of water and oil are denoted $q_w = q_w(t,x)$ and $q_o = q_o(t,x)$. They are used to describe the flow from injection wells and the flow to production wells.

The initial concentrations of water and oil in the reservoir are specified

$$C_w(t,x) = C_{w0}(x) \quad t \in \delta T,\, x \in \Omega \tag{2a}$$
$$C_o(t,x) = C_{o0}(x) \quad t \in \delta T,\, x \in \Omega \tag{2b}$$

as well as the flux of water and oil at the boundaries

$$N_w(t,x) = 0 \quad t \in T,\, x \in \delta\Omega \tag{3a}$$
$$N_o(t,x) = 0 \quad t \in T, x \in \delta\Omega \tag{3b}$$

B. Constitutive Models

The concentrations of water and oil in the reservoir may be expressed as

$$C_w = \phi\rho_w(P_w)S_w \tag{4a}$$
$$C_o = \phi\rho_o(P_o)S_o \tag{4b}$$

ϕ is the porosity of the reservoir rock. The porosity is the volumetric fraction of void space that can be occupied by the reservoir fluids (water and oil). We assume the porosity to be constant. $\rho_w = \rho_w(P_w)$ and $\rho_o = \rho_o(P_o)$ are the densities of water and oil. These densities depend on the water pressure P_w and the oil pressure P_o, respectively. S_w and S_o are called the saturations of water and oil, respectively. They represent the volumetric fraction of water and oil in the void space. Consequently, as water and oil jointly fills the entire void space

$$S_w + S_o = 1 \tag{5}$$

Water and oil are transported by convection through the porous medium. Therefore, the water and oil fluxes can be expressed as

$$N_w = \rho_w(P_w)u_w(P_w, S_w) \tag{6a}$$
$$N_o = \rho_o(P_o)u_o(P_o, S_o) \tag{6b}$$

$u_w = u_w(P_w, S_w)$ and $u_o = u_o(P_o, S_o)$ are the linear velocities of each phase.

The flow through the porous medium is pressure driven and can be expressed using Darcy's law

$$u_w = -k\frac{k_{rw}(S_w)}{\mu_w}\frac{\partial P_w}{\partial x} \tag{7a}$$
$$u_o = -k\frac{k_{ro}(S_o)}{\mu_o}\frac{\partial P_o}{\partial x} \tag{7b}$$

$k = k(x)$ denotes the permeability of the porous medium. The permeabilities depend only on the spatial position in the reservoir. μ_w and μ_o are the viscosities of oil and water. The relative permeabilities of each phase $k_{rw} = k_{rw}(S_w)$ and $k_{ro} = k_{ro}(S_o)$ are nonlinear functions of the saturation of the associated phase.

The relative permeabilities are approximated by the Corey relations

$$s_w = \frac{S_w - S_{wc}}{1 - S_{wc} - S_{or}} \tag{8a}$$
$$s_o = \frac{S_o - S_{or}}{1 - S_{wc} - S_{or}} \tag{8b}$$
$$k_{rw}(S_w) = \begin{cases} 0 & 0 \leq S_w \leq S_{wc} \\ k_{rw0}s_w^{n_w} & S_{wc} < S_w < 1 - S_{or} \\ k_{rw0} & 1 - S_{or} \leq S_w \leq 1 \end{cases} \tag{8c}$$
$$k_{ro}(S_o) = \begin{cases} 0 & 0 \leq S_o \leq S_{or} \\ k_{ro0}s_o^{n_o} & S_{or} < S_o < 1 - S_{wc} \\ k_{ro0} & 1 - S_{wc} \leq S_o \leq 1 \end{cases} \tag{8d}$$

k_{rw0}, k_{ro0}, n_w and n_o are determined experimentally for each particular porous medium. S_{wc} is the critical water saturation and S_{or} is the residual oil saturation. s_w and s_o are the reduced saturations.

Assuming that the fluid compressibilities c_w and c_o are constant over the pressure range of interest, the fluid densities can be expressed using the following equation of states

$$\rho_w = \rho_{w0} e^{c_w(P_w - P_{w0})} \tag{9a}$$

$$\rho_o = \rho_{o0} e^{c_o(P_o - P_{o0})} \tag{9b}$$

$\rho_{w0} = \rho_w(P_{w0})$ and $\rho_{o0} = \rho_o(P_{o0})$ are the densities at the reference pressures P_{w0} and P_{o0}.

The pressure in the wetting fluid (the water in this case) is less than in the non-wetting fluid. The pressure difference is given by the capillary pressure, which is assumed to be a function of the water saturation

$$P_{cow}(S_w) = P_o - P_w \tag{10}$$

In highly permeable and highly porous media the capillary effects are small. In dense formations with very small pores the capillary pressure introduces a diffusive term into (1) [19]. The irreducible saturations S_{wc} and S_{or} can partly be explained by the capillary effects in the porous medium, so the capillary pressure is to some extend taken into account through the relative permeabilities. In this model we assume zero capillary pressure

$$P_{cow}(S_w) = 0 \tag{11}$$

C. Well Models

The injector and producer wells are all located at point locations. Let $\mathcal{I}$ denote an index set for the injectors and $\mathcal{P}$ denote an index set for the producers. Then the injectors are located at $x_{\mathcal{I},j}$ for $j \in \mathcal{I}$ and the producers are located at $x_{\mathcal{P},j}$ for $j \in \mathcal{P}$. Let $\delta_{\mathcal{I},j} = \delta(x - x_{\mathcal{I},j})$ and $\delta_{\mathcal{P},j} = \delta(x - x_{\mathcal{P},j})$ denote Dirac's delta function.

Then the set of injector wells may be modeled as

$$q_w = q_w(t,x) = \rho_w(P_w) q_j \delta_{\mathcal{I},j} \quad j \in \mathcal{I} \tag{12a}$$

$$q_o = q_o(t,x) = 0 \quad j \in \mathcal{I} \tag{12b}$$

in which q_j is the volumetric injection rate of water at injection well $j \in \mathcal{I}$. The set of producer wells are modeled as

$$q_w = -\alpha_j w_j \rho_w(P_w) \frac{k_{rw}(S_w)}{\mu_w} (P_w - P_{well,j}) \delta_{\mathcal{P},j} \tag{13a}$$

$$q_o = -\alpha_j w_j \rho_o(P_o) \frac{k_{ro}(S_o)}{\mu_o} (P_o - P_{well,j}) \delta_{\mathcal{P},j} \tag{13b}$$

in which $j \in \mathcal{P}$. $P_{well,j}$ is the pressure at the well. $\alpha_j \in [0,\,1]$ is the position of the control valve and w_j is the well-index for production well $j \in \mathcal{P}$ [13]–[15].

D. State Transformation

Let $P = P(t,x) = P_o(t,x)$ be the oil pressure and $S = S(t,x) = S_w(t,x)$ be the water saturation. In the two-phase immiscible flow model (1)-(11), we may use (S,P) as state variables instead of (C_w, C_o). (4), (5), (10) and (11) may be used to compute S and P given C_w and C_o.

This imply that we may state the initial conditions (2) as initial saturations and pressures

$$S(t,x) = S_0(x) \quad t \in \delta T,\ x \in \Omega \tag{14a}$$

$$P(t,x) = P_0(x) \quad t \in \delta T,\ x \in \Omega \tag{14b}$$

III. General 3-Dimensional Model

Consider the time domain $T = \{t \in \mathbb{R} : t \geq 0\}$ and the spatial domain $\Omega \subset \mathbb{R}^3$. Let $\nabla = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \end{bmatrix}^{\mathrm{T}}$. Using the state variables $S = S(t,x,y,z) = S_w(t,x,y,z)$ and $P = P(t,x,y,z) = P_o(t,x,y,z)$ for $t \in T$ and $(x,y,z) \in \Omega$, we may generalize the mass balances of the 1-D model to the general 3-D model

$$\frac{\partial C_w}{\partial t} = -\nabla \cdot \mathbf{N}_w + q_w \quad t \in T^o,\ (x,y,z) \in \Omega^o \tag{15a}$$

$$\frac{\partial C_o}{\partial t} = -\nabla \cdot \mathbf{N}_o + q_o \quad t \in T^o,\ (x,y,z) \in \Omega^o \tag{15b}$$

$\mathbf{N}_w = \begin{bmatrix} N_{w,x} & N_{w,y} & N_{w,z} \end{bmatrix}^{\mathrm{T}}$ and $\mathbf{N}_o = \begin{bmatrix} N_{o,x} & N_{o,y} & N_{o,z} \end{bmatrix}^{\mathrm{T}}$ are 3-dimensional flux vectors with the mass flux in each spatial direction as coordinates. The initial conditions with (S,P) as state variables are

$$S(t,x,y,z) = S_0(x,y,z) \quad t \in \delta T,\ (x,y,z) \in \Omega \tag{16a}$$

$$P(t,x,y,z) = P_0(x,y,z) \quad t \in \delta T,\ (x,y,z) \in \Omega \tag{16b}$$

The zero net influx condition at the spatial boundary can be expressed as

$$\mathbf{N}_w(t,x,y,z) \cdot \mathbf{n}(x,y,z) = 0 \quad t \in T,\ (x,y,z) \in \delta\Omega \tag{17a}$$

$$\mathbf{N}_o(t,x,y,z) \cdot \mathbf{n}(x,y,z) = 0 \quad t \in T,\ (x,y,z) \in \delta\Omega \tag{17b}$$

$\mathbf{n}(x,y,z)$ for $(x,y,z) \in \delta\Omega$ is a normal vector the boundary $\delta\Omega$.

The flux of water and oil is by convection. Therefore, the flux vectors are

$$\mathbf{N}_w = \rho_w(P_w) \mathbf{u}_w \tag{18a}$$

$$\mathbf{N}_o = \rho_o(P_o) \mathbf{u}_o \tag{18b}$$

in which $\mathbf{u}_w$ and $\mathbf{u}_o$ are linear velocities governed by Darcy's law

$$\mathbf{u}_w = -\mathbf{K} \frac{k_{rw}(S_w)}{\mu_w} \left(\nabla P_w - \rho_w(P_w) g \nabla z \right) \tag{19a}$$

$$\mathbf{u}_o = -\mathbf{K} \frac{k_{ro}(S_o)}{\mu_o} \left(\nabla P_o - \rho_o(P_o) g \nabla z \right) \tag{19b}$$

with $\mathbf{K} = \mathrm{diag}(k_x, k_y, k_z)$ being a diagonal permeability matrix. Often $\mathbf{K} = k\mathbf{I}$.

The fluid and rock properties are determined in the same was as for the 1-dimensional model. The source terms at injection wells and the sink terms at production wells are determined by a trivial extension of Dirac's delta-function to the 3-dimensional case, i.e. $\delta_j = \delta(x - x_j)\delta(y - y_j)\delta(z - z_j)$.

IV. Finite Volume Discretization

In this section we present a method for spatial discretization of the two-phase immiscible flow problem. Using Gauss' divergence theorem, (15) may be expressed in integral form

$$\frac{\partial}{\partial t}\int_{\Omega} C_w dV = -\int_{\delta\Omega} (\mathbf{N}_w \cdot \mathbf{n}) dS + \int_{\Omega} q_w dV \tag{20a}$$

$$\frac{\partial}{\partial t}\int_{\Omega} C_o dV = -\int_{\delta\Omega} (\mathbf{N}_o \cdot \mathbf{n}) dS + \int_{\Omega} q_o dV \tag{20b}$$

$\mathbf{n}$ is an outward pointing normal vector on the boundary. The model integral form then consists of (20) along with the initial conditions (16) and the boundary conditions (17). In this section we describe a finite volume method for spatial discretization of this model. For notational simplicity we consider the equation

$$\frac{\partial}{\partial t}\int_{\Omega} C dV = -\int_{\delta\Omega} (\mathbf{N} \cdot \mathbf{n}) dS + \int_{\Omega} q dV \tag{21}$$

and the structured grid of finite volumes

$$\begin{aligned}\Omega_{i,j,k} = \{(x,y,z) \in \mathbb{R}^3 : x_{i-} \le x \le x_{i+}, \\ y_{j-} \le y \le y_{j+},\ z_{k-} \le z \le z_{k+}\}\end{aligned} \tag{22}$$

with $x_{i-} = \sum_{l=1}^{i-1} \Delta x_l$, $x_i = x_{i-} + \frac{1}{2}\Delta x_i$, and $x_{i+} = x_{i-} + \Delta x_i$. Similar definitions apply in the y and z directions. $\Omega_{i,j,k}$ is a finite volume with mid-point in (x_i, y_j, z_k). $\delta\Omega_{i,j,k}$ is the boundary of the finite volume $\Omega_{i,j,k}$. The concentration in each control volume $\Omega_{i,j,k}$ is obtained using

$$\frac{\partial}{\partial t}\int_{\Omega_{i,j,k}} C dV = -\int_{\delta\Omega_{i,j,k}} (\mathbf{N} \cdot \mathbf{n}) dS + \int_{\Omega_{i,j,k}} q dV \tag{23}$$

$\forall (i,j,k)$ along with the initial and boundary conditions. The accumulation term is

$$\frac{\partial}{\partial t}\int_{\Omega_{i,j,k}} C dV = \frac{dC_{i,j,k}}{dt}(t) V_{i,j,k} \tag{24}$$

with $V_{i,j,k} = \Delta x_i \Delta y_j \Delta z_k$. The flux term on the boundary may be expressed as

$$\begin{aligned}&\int_{\delta\Omega_{i,j,k}} (\mathbf{N} \cdot \mathbf{n}) dS = \\ &\quad S_{x,j,k}(N_x(t, x_{i+}, y_j, z_k) - N_x(t, x_{i-}, y_j, z_k)) + \\ &\quad S_{y,i,k}(N_y(t, x_i, y_{j+}, z_k) - N_y(t, x_i, y_{j-}, z_k)) + \\ &\quad S_{z,i,j}(N_z(t, x_i, y_j, z_{k+}) - N_z(t, x_{i-}, y_j, z_{k-}))\end{aligned} \tag{25}$$

with $S_{x,j,k} = \Delta y_j \Delta z_k$, $S_{y,i,k} = \Delta x_i \Delta z_k$, and $S_{z,i,j} = \Delta x_i \Delta y_j$. Define the fluid transmissibility $\zeta_{i,j,k} = \rho(P_{i,j,k}) k_r(S_{i,j,k})/\mu$. The fluxes in the x-direction are

$$N_{x,i+,j,k} = -k_{x,i+,j,k}\zeta_{i+,j,k}\frac{P_{i+1,j,k} - P_{i,j,k}}{x_{i+1} - x_i} \tag{26a}$$

$$N_{x,i-,j,k} = -k_{x,i-,j,k}\zeta_{i-,j,k}\frac{P_{i,j,k} - P_{i-1,j,k}}{x_i - x_{i-1}} \tag{26b}$$

$$\Delta N_{x,i,j,k} = N_{x,i+,j,k} - N_{x,i-,j,k} \tag{26c}$$

To have flux continuity across interfaces, $k_{x,i+,j,k}$ and $k_{x,i-,j,k}$ are computed using the harmonic average for the permeabilities in the adjacent grid blocks

$$\frac{\Delta x_i + \Delta x_{i-1}}{k_{x,i-,j,k}} = \frac{\Delta x_i}{k_{x,i,j,k}} + \frac{\Delta x_{i-1}}{k_{x,i-1,j,k}} \tag{27a}$$

$$\frac{\Delta x_{i+1} + \Delta x_i}{k_{x,i+,j,k}} = \frac{\Delta x_{i+1}}{k_{x,i+1,j,k}} + \frac{\Delta x_i}{k_{x,i,j,k}} \tag{27b}$$

Due to the hyperbolic nature of the equation, the fluid transmissibilities, $\zeta_{i+,j,k}$ and $\zeta_{i-,j,k}$, are computed using upstream information. The single point upstream scheme is

$$\zeta_{i+,j,k} = \begin{cases}\zeta_{i+1,j,k} & P_{i+1,j,k} - P_{i,j,k} \ge 0 \\ \zeta_{i,j,k} & P_{i+1,j,k} - P_{i,j,k} < 0\end{cases} \tag{28a}$$

$$\zeta_{i-,j,k} = \begin{cases}\zeta_{i,j,k} & P_{i,j,k} - P_{i-1,j,k} \ge 0 \\ \zeta_{i-1,j,k} & P_{i,j,k} - P_{i-1,j,k} < 0\end{cases} \tag{28b}$$

Similar expressions are derived for the fluxes in the y- and z-directions. In the z-direction we incorporate the gravity term in the flow direction test.

The source/sink term of (23) becomes

$$\int_{\Omega_{i,j,k}} q dV = q_{i,j,k}(t) V_{i,j,k} \tag{29}$$

Consequently, we may express the model (23) as a system of ordinary differential equations in the form

$$\begin{aligned}\frac{dC_{i,j,k}}{dt} = &-\left(\frac{\Delta N_{x,i,j,k}}{\Delta x_i} + \frac{\Delta N_{y,i,j,k}}{\Delta y_j} + \frac{\Delta N_{z,i,j,k}}{\Delta z_k}\right) \\ &+ q_{i,j,k}\end{aligned} \tag{30}$$

V. ESDIRK

In this section we describe a method for temporal discretization of the two-phase immiscible flow model.

A. New Differential Equation Model

Explicit Singly Diagonally Implicit Runge-Kutta (ESDIRK) methods has previously been developed for system of ordinary differential equations [20]–[23]

$$\frac{d}{dt}x(t) = f(t, x(t)) \quad x(t_0) = x_0 \tag{31}$$

and systems of index-1 differential algebraic equations [23], [24]

$$M(t, x(t))\frac{d}{dt}x(t) = f(t, x(t)) \quad x(t_0) = x_0 \tag{32}$$

While the spatially discretized model for the immiscible two-phase flow problem may be formulated as (32), the main problem with such a formulation is that it is not guaranteed to preserve mass upon discretization in time. This is a major problem, as the differential equations was formulated based on conservation of mass. Process simulation problems in general are based on conservation of mass, energy and momentum. It is desirable to preserve such properties upon numerical discretization in time.

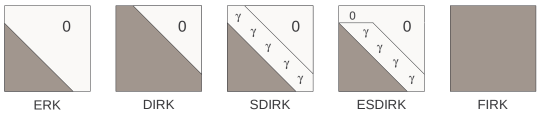


Fig. 2. The A-matrix in Runge-Kutta methods.

Therefore, we propose a new system of differential equation model for simulation of process systems. This model can be formulated as

$$\frac{d}{dt}g(x(t)) = f(t, x(t)) \quad x(t_0) = x_0 \tag{33}$$

$x(t)$ are the states, while $g(x(t))$ are the properties conserved. In the two-phase flow problem considered, $g(x(t))$ is a vector with oil and water concentrations in each grid-cell of the reservoir. $x(t)$ is a vector with water saturation and oil pressure in each grid cell of the reservoir. The right hand side function, $f(t, x(t))$, has the usual interpretation. Consequently, the spatially discretized immiscible two-phase flow problem (30) has the structure of (33).

In general process system models based on conservation of mass, energy, and momentum can directly be formulated as the model (33). Upon discretization in time this model preserves mass, energy, and momentum. This is not in general the case, if these process systems models are expressed as (32) using the chain rule, i.e. $\frac{d}{dt}g(x(t)) = \frac{\partial g}{\partial x}(x(t))\frac{dx}{dt}(t) = M(t, x(t))\frac{d}{dt}x(t)$ with $M(t, x(t)) = \frac{\partial g}{\partial x}(x(t))$.

B. Integration Method

Consider Runge-Kutta methods with s stages in each time step. Such Runge-Kutta methods for integration of (33) can be expressed as

$$T_i = t_n + c_i h \qquad i = 1, \dots, s \tag{34a}$$

$$g(X_i) = g(x_n) + h_n \sum_{i=1}^{s} a_{ij} f(T_j, X_j) \quad i = 1, \dots, s \tag{34b}$$

with X_i being a numerical approximation to $x(T_i)$. h_n is the step length of the current step.

Different Runge-Kutta methods are obtained depending on the structure of the matrix $A = [a_{ij}]$. If the A-matrix is strictly lower triangular an explicit Runge-Kutta (ERK) method is obtained, while a full A-matrix yield a fully implicit Runge-Kutta (FIRK) method. A number of variations in between these two extreme cases exist. This is illustrated in Figure 2. ERK methods are computationally fast but cannot be applied to stiff systems such as the two-phase problem. FIRK methods require simultaneous solution of (34b) and is computationally demanding. However, these method have excellent stability properties and can be applied for solution of stiff systems of differential equations as well as index-1 differential algebraic equations.

Explicit Singly Diagonally Implicit Runge-Kutta (ESDIRK) methods has a lower triangular A-matrix. They have an explicit first stage and a single coefficient, $a_{ii} = \gamma$ for $i = 2, \dots, s$, on the remaining diagonals of the A-matrix. The explicit first stage implies that $T_1 = t_n$ and $X_1 = x_n$. The subsequent state values, X_i, at $T_i = t_n + h_n c_i$ for $i = 2, \dots, s$ may be solved sequentially. Hence, the state values X_i are obtained by sequential solution of

$$R(X_i) = g(X_i) - h_n \gamma f(T_i, X_i) - \psi_i = 0 \;\; i = 2, \dots, s \tag{35}$$

with

$$\psi_i = g(x_n) + h_n \sum_{j=1}^{i-1} a_{ij} f(T_j, X_j) \quad i = 2, \dots, s \tag{36}$$

using some modified Newton's method. The Jacobian of the residual, $R(X_i)$, is

$$\begin{aligned} J(X_i) = \frac{\partial R}{\partial X_i}(X_i) &= \frac{\partial g}{\partial x}(X_i) - h_n \gamma \frac{\partial f}{\partial x}(T_i, X_i) \\ &\approx \frac{\partial g}{\partial x}(x_m) - h_m \gamma \frac{\partial f}{\partial x}(t_m, x_m) \\ &= J(x_m) = LU \end{aligned} \tag{37}$$

The Jacobian is only re-evaluated and factorized when the modified Newton step

$$LU \Delta X_i = R(X_i) \tag{38a}$$

$$X_i := X_i - \Delta X_i \tag{38b}$$

for solution of (35) converges too slowly or even diverge.

The coefficients, $b = [b_i]$, in the quadrature equation

$$g(x_{n+1}) = g(x_n) + h_n \sum_{i=1}^{s} b_i f(T_i, X_i) \tag{39}$$

are constructed such this equation is equal to the last stage, i.e. $b_i = a_{s,i}$ for $i = 1, \dots, s$ and $c_s = 1$. This implies that the next time and state value may be computed as

$$t_{n+1} = t_n + h_n = T_4 \tag{40a}$$

$$x_{n+1} = X_4 \tag{40b}$$

The ESDIRK methods [23] has an embedded error estimator

$$\hat{e}_{n+1} = g(x_{n+1}) - g(\hat{x}_{n+1}) = h_n \sum_{i=1}^{s} d_i f(T_i, X_i) \tag{41}$$

with the asymptotic relation $\hat{e}_{n+1} = \phi(t_n, x_n) h_n^{p+1}$ where p is the order of the basic ESDIRK method. It should be noted that $\hat{e}_{n+1}$ is an error estimate of the conserved quantities $g(x_{n+1})$ and not the states x_{n+1} themselves. Measures of the error such as

$$\hat{r}_{n+1} = \frac{1}{n_x} \sqrt{\sum_{i=1}^{n_x} \left(\frac{|(\hat{e}_{n+1})_i|}{\text{abs}_i + |(g(x_{n+1}))_i| \cdot \text{rel}_i} \right)^2} \tag{42a}$$

$$\hat{r}_{n+1} = \max_{i \in \{1, \dots, n_x\}} \frac{|(\hat{e}_{n+1})_i|}{\text{abs}_i + |(g(x_{n+1}))_i| \cdot \text{rel}_i} \tag{42b}$$

may be controlled adjusting the time step according to [25], [26]

$$h_{n+1} = \frac{h_n}{h_{n-1}} \left(\frac{\varepsilon}{\hat{r}_{n+1}} \right)^{k_2/k} \left(\frac{\hat{r}_n}{\hat{r}_{n+1}} \right)^{k_1/k} h_n \tag{43}$$

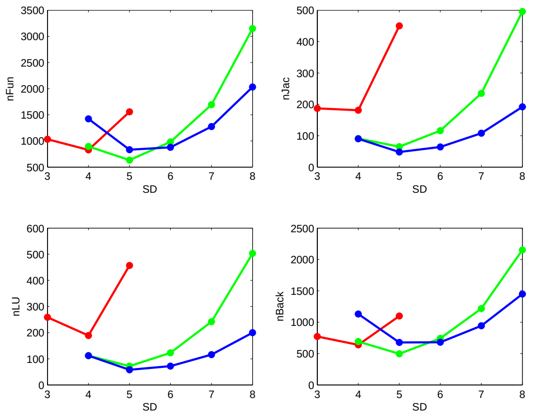


Fig. 3. Performance comparison of three ESDIRK methods as function of the number of significant digits (SD) in the solution. The performance is measured by number of function evaluations, jacobian evaluations, LU factorizations, and back substitutions. ESDIRK12 = red, ESDIRK23 = green, ESDIRK34 = blue.

with $k = p + 1$, $k_1 = k_2 = 1$, $\varepsilon = 0.8$.

The parameters $\{[a_{ij}], [c_i], [d_i], p, k_1, k_2, \varepsilon\}$ defining an ESDIRK method are provided in [20]–[23]. They are computed such that the integration method satisfy the Runge-Kutta order conditions and such that the integration method is A- and L-stable. We have implemented ESDIRK12, ESDIRK23, and ESDIRK34.

VI. Simulation Example

We have defined a 1-dimensional two phase immiscible flow problem with 1000 grid blocks. The performance of the ESDIRK methods have been compared in terms of statistics describing the computational efficiency. The results are plotted in Figure 3. It is evident that the higher ESDIRK methods are much more efficient that ESDIRK12 which corresponds to an implicit Euler method with a trapezoidal error estimator.

VII. Conclusion

In optimal control of the two phase immiscible flow problem is an example of a large-scale optimal control problem. The major computational effort for solution of this problem is spent solving the differential equations. We have developed new mass preserving adaptive ESDIRK methods for solution of the differential equations and demonstrated that high order methods are more efficient than low order methods such as the implicit Euler method. The methods are adaptive and equipped with an error estimator.

References

[1] R. G. Smith and G. Maitland, "The road ahead to real-time oil and gas reservoir management," *Trans IChemE*, vol. 76, pp. 539–552, 1998.

[2] H. Bieker, O. Slupphaug, and T. Johansen, "Real time production optimization of offshore oil and gas production systems: A technology survey," in *SPE Intelligent Energy Conference and Exhibition*, no. SPE 99446. SPE, 2006.

[3] D. Brouwer and J.-D. Jansen, "Dynamic optimization of waterflooding with smart wells using optimal control theory," *SPE Journal*, pp. 391–402, 2004.

[4] G. Nævdal, D. R. Brouwer, and J.-D. Jansen, "Waterflooding using closed-loop control," *Computational Geosciences*, vol. 10, pp. 37–60, 2006.

[5] J. Kraaijevanger, P. Egberts, J. Valstar, and H. Buurman, "Optimal waterflood design using the adjoint method," in *SPE Reservoir Simulation Symposium*, no. SPE 105764. Houston, Texas: SPE, 2007.

[6] M. Zandvliet, O. Bosgra, J. Jansen, P. Van den Hof, and J. Kraaijevanger, "Bang-bang control and singular arcs in reservoir flooding," *Journal of Petroleum Science and Engineering*, vol. 58, pp. 186–200, 2007.

[7] J.-D. Jansen, O. H. Bosgra, and P. M. Van den Hof, "Model-based control of multiphase flow in subsurface oil reservoirs," *Journal of Process Control*, vol. 18, pp. 846–855, 2008.

[8] P. Valvatne, J. Serve, L. Durlofsky, and K. Aziz, "Efficient modeling of nonconventional wells with downhole inflow control devices," *Journal of Petroleum Science and Engineering*, vol. 39, pp. 99–116, 2003.

[9] P. Sarma, L. J. Durlofsky, K. Aziz, and W. H. Chen, "Efficient real-time reservoir management using adjoint-based optimal control and model updating," *Computational Geosciences*, vol. 10, pp. 3–36, 2006.

[10] Z. Fathi and W. F. Ramirez, "Use of optimal control theory for computing optimal injection policies for enhanced oil recovery," *Automatica*, vol. 22, pp. 33–42, 1986.

[11] ——, "Optimization of an enhanced oil recovery process with boundary controls - a large-scale non-linear maximization," *Automatica*, vol. 23, pp. 301–310, 1987.

[12] W. Ramirez, *Application of Optimal Control Theory to Enhanced Recovery*, ser. Developments in Petroleum Science. New York: Elsevier, 1987, vol. 21.

[13] Z. Chen, G. Huan, and Y. Ma, *Computational Methods for Multiphase Flows in Porous Media*. Philadelphia, PA: SIAM, 2006.

[14] Z. Chen, *Reservoir Simulation. Mathematical Techniques in Oil Recovery*. Philadelphia, PA: SIAM, 2007.

[15] K. Aziz, L. Durlofsky, and H. Tchelepi, *Notes on Petroleum Reservoir Simulation*. Stanford, California: Department of Petroleum Engineering, School of Earth Sciences, Stanford University, 2005.

[16] C. E. Kess and C. T. Miller, "Higher order time integration methods for two-phase flow," *Advances in Water Resources*, vol. 25, pp. 159–177, 2002.

[17] H. A. Tchelepi and K. Aziz, "Advances in reservoir simulation," in *8th International Forum on Reservoir Simulation*. Iles Borromees, Stresa, Italy: International Forum on Reservoir Simulation, 2005.

[18] M. G. Gerritsen and L. J. Durlofsky, "Modeling fluid flow in oil reservoirs," *Annual Review of Fluid Mechanics*, vol. 37, pp. 211–238, 2005.

[19] R. A. Berenblyum, A. A. Shapiro, K. Jessen, E. H. Stenby, and F. M. Orr, "Black oil streamline simulator with capillary effects," in *SPE Annual Technical Conference and Exhibition*, no. SPE 84037. Denver, Colorado: SPE, 2003.

[20] R. Alexander, "Design and implementation of DIRK integrators for stiff systems," *Applied Numerical Mathematics*, vol. 46, pp. 1–17, 2003.

[21] M. R. Kristensen, J. B. Jørgensen, P. G. Thomsen, and S. B. Jørgensen, "An ESDIRK method with sensitivity analysis capabilities," *Computers and Chemical Engineering*, vol. 28, pp. 2695–2707, 2004.

[22] A. Kværnø, "Singly diagonally implicit runge-kutta methods with an explicit first stage," *BIT Numerical Mathematics*, vol. 44, pp. 489–502, 2004.

[23] J. B. Jørgensen, M. R. Kristensen, and P. G. Thomsen, "A family of ESDIRK integration methods," *manuscript for SIAM Journal of Scientific Computing*, 2008.

[24] M. R. Kristensen, J. B. Jørgensen, P. G. Thomsen, M. L. Michelsen, and S. B. Jørgensen, "Sensitivity analysis in index-1 differential algebraic equations by ESDIRK methods," in *16th IFAC World Congress 2005*. Prague, Czech Republic: IFAC, 2005.

[25] G. Söderlind, "Automatic control and adaptive time-stepping," *Numerical Algorithms*, vol. 31, pp. 281–310, 2002.

[26] ——, "Time-step selection algorithms: Adaptivity, control, and signal processing," *Applied Numerical Mathematics*, vol. 56, pp. 488–502, 2006.

APPENDIX E

Paper B

Adaptive Stepsize Control in Implicit Runge-Kutta Methods for Reservoir Simulation

Authors:
Carsten Völcker, John Bagterp Jørgensen, Per Grove Thomsen and Erling Halfdan Stenby

Published in:
Proceedings of *9th International Symposium on Dynamics and Control of Process Systems (DYCOPS 2010)*
Leuven, Belgium
5-7 July 2010

Adaptive Stepsize Control in Implicit Runge-Kutta Methods for Reservoir Simulation [⋆]

Carsten Völcker [*] John Bagterp Jørgensen [*] Per Grove Thomsen [*] Erling Halfdan Stenby [**]

[*] *Department of Informatics and Mathematical Modeling, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark (e-mail:* `{cv,jbj,pgt}@imm.dtu.dk`*)*
[**] *Department of Chemical and Biochemical Engineering, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark (e-mail:* `ehs@kt.dtu.dk`*)*

Abstract: This paper concerns predictive stepsize control applied to high order methods for temporal discretization in reservoir simulation. The family of Runge-Kutta methods is presented and in particular the explicit singly diagonally implicit Runge-Kutta (ESDIRK) methods are described. A predictive stepsize adjustment rule based on error estimates and convergence control of the integrated iterative solver is presented. We try to improve the predictive stepsize control by smoothing the stepsize sequence through combining the control of error with the control of convergence.

Keywords: Reservoir simulation, Runge-Kutta methods, convergence control, stepsize selection.

1. INTRODUCTION

Reservoir simulators are computer programs that solve the equations for heat and mass flow in porous media. Numerical integration is one of the basic steps involved in the simulation process. The number and type of equations to be solved depend on the geological characteristics of the reservoir, the characteristics of the oil and the oil recovery process to be modeled. Choosing the appropriate method of integration involves deciding on factors such as the order of the integration scheme, stability properties and concern on computational efficiency. ESDIRK methods have been applied successfully for solution of convection-diffusion-reaction problems, see Kennedy and Carpenter (2003). This class of methods is computationally efficient, and both A- and L-stable and stiffly accurate ESDIRK methods of various order with an embedded method for error estimation have been derived by Kværnø (2004) and Jørgensen et al. (2008). In addition, a robust adaptive stepsize selection is essential to an efficient numerical integration. An adaptive stepsize selection aims to keep the error estimate bounded i.e. close to a user-specified tolerance by adjusting the timestep. Gustafsson (1992) suggested a strategy for stepsize selection based on the rigorous error estimates provided by embedded Runge-Kutta methods.
We have applied the controller by Gustafsson and Söderlind (1997) to three different ESDIRK methods used for solving a two-phase reservoir model. Although the control strategy has proven efficient we observed that certain steps were rejected due to irregularities in the stepsize selection. We found that a different interaction between the error and the convergence control in the stepsize selection process may solve this problem. The idea is to combine the control of error with control of convergence in the inner iterations in a simple logic that minimizes the number of rejected steps and thereby improves the efficiency.

2. DIFFERENTIAL EQUATION MODEL

In this section we briefly outline the two-phase flow problem and we present the typical formulation of a system of ordinary differential equations (ODE) based on conservation laws.

2.1 The two-phase flow problem

We consider immiscible two-phase flow of oil and water in porous media. Let $P_o = P_o(t, x)$ be the pressure of oil and $S_w = S_w(t, x)$ be the saturation of water, as function of time $t \geq 0$ and position $x \subset \mathbb{R}^2$, and let $C_w = C_w(P_o, S_w)$ and $C_o = C_o(P_o, S_w)$ be the mass concentrations of water and oil respectively. Then the mass balances for water and oil in the reservoir is expressed by the following system of partial differential equations

$$\frac{\partial}{\partial t} C_w = -\nabla \cdot F_w + Q_w \tag{1a}$$

$$\frac{\partial}{\partial t} C_o = -\nabla \cdot F_o + Q_o \tag{1b}$$

$F_w = F_w(P_o, S_w)$ and $F_o = F_o(P_o, S_w)$ are the fluxes of water and oil through the porous media. The source/sink

[⋆] This research project is financially supported by the Danish Research Council for Technology and Production Sciences. FTP Grant no. 274-06-0284.

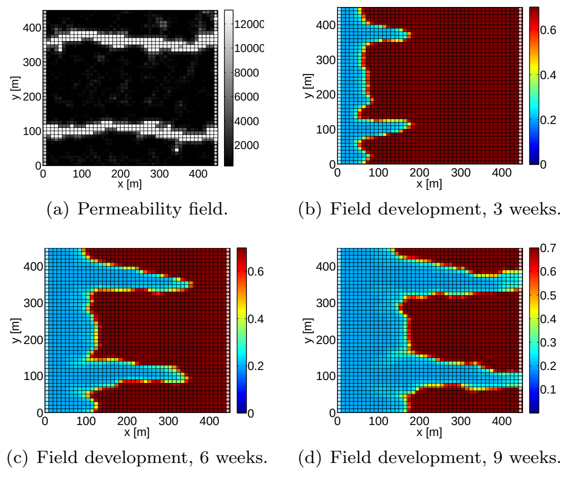

(a) Permeability field. (b) Field development, 3 weeks. (c) Field development, 6 weeks. (d) Field development, 9 weeks.

Fig. 1. Field development after 9 weeks of water injection.

terms of water and oil are denoted $Q_w = Q_w(P_o, S_w)$ and $Q_o = Q_o(P_o, S_w)$. They are used to describe the flow from injection wells and the flow to production wells. A more profound description can be found in Chen (2007) and Völcker et al. (2009). We use a standard 2-D problem defined by Brouwer and Jansen (2004), depicted in Figure 1.

2.2 An ODE system in general

Many process simulation problems in general are based on conservation of mass, energy and momentum. It is desirable to preserve such properties upon numerical integration in time. As proposed by Völcker et al. (2009) a general formulation of such an ODE system may be

$$\frac{d}{dt}g(x(t)) = f(t, x(t)) \quad x(t_0) = x_0 \tag{2}$$

where $x(t)$ denotes the system states, $g(x(t))$ are the properties conserved, while the right-hand side function $f(t, x(t))$ has the usual interpretation.

3. INTEGRATION METHODS

In this section different classes of Runge-Kutta methods are outlined. In particular ESDIRK methods are described.

3.1 Runge-Kutta Integration

An s-stage Runge-Kutta method for integration of (2) can be expressed as

$$T_i = t_n + h_n c_i \qquad i \in \mathbb{S}_1 \tag{3a}$$

$$g(X_i) = g(x_n) + h_n \sum_{j=1}^{s} a_{ij} f(T_j, X_j) \quad i \in \mathbb{S}_1 \tag{3b}$$

$$g(x_{n+1}) = g(x_n) + h_n \sum_{j=1}^{s} b_j f(T_j, X_j) \tag{3c}$$

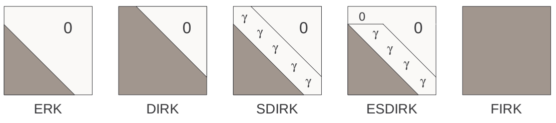


Fig. 2. The A-matrix of Runge-Kutta methods.

where T_i and X_i are the internal stage values being numerical approximations to $x(T_i)$. x_{n+1} is the step computed at $t_{n+1} = t_n + h_n$. The set $\mathbb{S}_i$ denotes the internal stages $i, i+1, \ldots, s$.

Different classes of Runge-Kutta methods can be obtained depending on the structure of the matrix $A = [a_{ij}]$. This is illustrated in Figure 2. Explicit Runge-Kutta (ERK) methods have a strictly lower triangular A-matrix which allows (3b) to be solved explicitly without iterations. Therefore, ERK methods are computationally fast but cannot be applied to stiff problems because of poor stability properties. All implicit methods are characterized by an A-matrix that is not strictly lower triangular and the state values X_i are computed iteratively by solution of (3b). Fully implicit Runge-Kutta (FIRK) methods, identified by a full A-matrix, have excellent stability properties making them usefull for solving stiff systems of ODE's. However, the excellent stability properties comes with high computational cost in solving (3b) simultaneously at each iteration step. To achieve some of the stability properties of the FIRK methods but at lower computational cost, various methods in between the ERK and the FIRK methods have been constructed.

3.2 ESDIRK Methods

ESDIRK methods have a lower triangular A-matrix. By construction they retain the stability properties of FIRK methods but at significant lower computational cost. Because $c_1 = 0$ and $a_{11} = 0$ the first stage in ESDIRK methods is explicit implying that the first state value equals the last step $(T_1, X_1) = (t_n, x_n)$. The subsequent stages are singly diagonally implicit such that the state values X_i at $T_i = t_n + h_n c_i$ for $i \in \mathbb{S}_2$ may be solved sequentially by solution of the residual

$$R(T_i, X_i) = g(X_i) - h_n \gamma f(T_i, X_i) - \psi_i = 0 \quad i \in \mathbb{S}_2 \tag{4}$$

with the term

$$\psi_i = g(x_n) + h_n \sum_{j=1}^{i-1} a_{ij} f(T_j, X_j) \quad i \in \mathbb{S}_2 \tag{5}$$

using Newton-Raphson's iterative method. The Jacobian $J_R(T_i, X_i) = \frac{\partial}{\partial X_i} R(T_i, X_i)$ of the residual is

$$J_R(T_i, X_i) = J_g(X_i) - h_n \gamma J_f(T_i, X_i) \quad i \in \mathbb{S}_2 \tag{6}$$

where $J_g(X_i) = \frac{d}{dX_i} g(X_i)$ and $J_f(T_i, X_i) = \frac{\partial}{\partial X_i} f(T_i, X_i)$ are the Jacobiants of the right- and left-hand sides of (2) respectively. We only consider methods assumed to be stiffly accurate by construction i.e. $c_s = 1$ and $a_{sj} = b_j$ for $j \in \mathbb{S}_1$. This implies that the quadrature function (3c) corresponds to the last internal stage in (3b). Consequently the next step equals the last state value $(t_{n+1}, x_{n+1}) = (T_s, X_s)$. The Butcher tableau for stiffly accurate ESDIRK methods is represented in (7).

$$\begin{array}{c|ccccc}
0 & 0 & & & & \\
c_2 & a_{21} & \gamma & & & \\
c_3 & a_{31} & a_{32} & \gamma & & \\
\vdots & \vdots & & & \ddots & \\
1 & b_1 & b_2 & b_3 & \cdots & \gamma \\
\hline
x_{n+1} & b_1 & b_2 & b_3 & \cdots & \gamma \\
\hat{x}_{n+1} & \hat{b}_1 & \hat{b}_2 & \hat{b}_3 & \cdots & \hat{b}_s \\
\hline
e_{n+1} & d_1 & d_2 & d_3 & \cdots & d_s
\end{array} \tag{7}$$

4. ERROR AND CONVERGENCE CONTROL

In this section we describe how to estimate the integration error, how the error is related to the user specified tolerances and how to control the convergence of the iterative solver.

4.1 Integration error

The ESDIRK method stated in (7) is equipped with an embedded Runge-Kutta method

$$g(\hat{x}_{n+1}) = g(x_n) + h_n \sum_{j=1}^{s} \hat{b}_j f(T_j, X_j) \tag{8}$$

computing the embedded solution $\hat{x}_{n+1}$. The embedded method is of different order, which then provides an estimate of the local truncation error

$$e_{n+1} = g(x_{n+1}) - g(\hat{x}_{n+1}) = h_n \sum_{j=1}^{s} d_j f(T_j, X_j) \tag{9}$$

corresponding to the numerical solution x_{n+1}. The integration error (9) is controlled adjusting the timestep by monitoring the root mean square of the error-tolerance relation

$$r_{n+1} = \frac{1}{\sqrt{m}} \left|\left| \frac{e_{n+1}}{atol + |g(x_{n+1})| rtol} \right|\right|_2 \tag{10}$$

where *atol* and *rtol* are componentwise user specified absolute and relative error tolerances and m is the dimension of the solution vector. Only stepsizes for which $r_{n+1} \leq 1$ are accepted.

4.2 Convergence control

The solution of (4) is done iteratively by a modified Newton-Raphson's method i.e. the Jacobian of the residual is not evaluated/factorized at each timestep. There is always a trade-off between the rate of convergence of the equation solver and the frequency of Jacobian updates/factorizations. For reasons of robustness the convergence rate is measured by the residuals Houbak et al. (1985)

$$\alpha_i = \frac{(r_R)_i^{k-1}}{(r_R)_i^k} \quad i \in \mathbb{S}_2 \tag{11}$$

where the iteration error of the k^{th} iteration is computed as the root mean square of the residual-tolerance relation

$$(r_R)_i^k = \frac{1}{\sqrt{m}} \left|\left| \frac{(R(T_i, X_i))^k}{atol + |(g(X_i))^k| rtol} \right|\right|_2 \quad i \in \mathbb{S}_2 \tag{12}$$

using the same componentwise absolute and relative error tolerances as in (10). If for some k during the iterations $\alpha \geq 1$ the iteration sequence is terminated and the stepsize is restricted. In case of convergence the iterations are successfully stopped when $(r_R)_i^k \leq \tau$. As noticed in Hairer and Wanner (1996) the choice of τ affects the efficiency of the algorithm. A large value of τ may lead to one or more large components in the integration error (10) with too many rejected steps as a result. We have chosen $\tau = 0.1$ as a compromise between robustness and computational speed.

5. STEPSIZE SELECTION

This section is divided into a brief description of the stepsize selection rule adopted, a description of the modifications that we suggest in order to simplify and stabilize the control algorithm and finally an outline of the complete controller is presented.

5.1 Predictive control

The integration error is controlled using a predictive controller for stepsize selection as presented by Gustafsson (1992). The controller must keep the estimate (9) of the local truncation error bounded and minimize the computational work in the solution process by trying to keep $r_{n+1} = 1$ by maximizing the stepsize. Based on empirical evidence Gustafsson (1992) suggested a proportional integral (PI) stepsize adjustment rule on the form

$$h_r = \frac{h_n}{h_{n-1}} \left(\frac{r_n}{r_{n+1}} \right)^{k_1/\hat{k}} \left(\frac{\epsilon}{r_{n+1}} \right)^{k_2/\hat{k}} h_n \tag{13}$$

where k_1 and k_2 are the gain parameters of the proportional and the integral parts respectively and $\hat{k}$ is the order of the embedded Runge-Kutta method, while ϵ is the desired tolerance (including a safety factor). Gustafsson (1992) suggests $k_1 = k_2 = 1$ corresponding to deadbeat control and a safety factor of 0.8.

5.2 Modified controller

The core stepsize adjustment rule (14) must be implemented along with a number of extensions and various safety nets and the original framework from which we propose our modifications can be found in Gustafsson (1992). Additionally a modification suggested by Gustafsson and Söderlind (1997) is described and implemented. The modified PI controller that we suggest is presented in Algorithm 5.1.

Since we are only considering stiffly accurate methods the order reduction for stiff systems can be avoided, see Prothero and Robinson (1974). Consequently the strategy described by Gustafsson (1992) for estimating $\hat{k}$ after successive rejects can be omitted. This does not make any noticeable change in the controller performance but simplifies the algorithm a great deal.
Besides the frequency of Jacobian updates/factorizations the stepsize is the only available control variable affecting the convergence rate of the equation solver. In order to assure convergence in the equation solver the stepsize has to be restrained in some situations. If convergence is too slow i.e. if $\alpha > \alpha_{ref}$ Gustafsson (1992) suggest the stepsize to be chosen as

$$h_\alpha = \frac{\alpha_{ref}}{\alpha} h_n \tag{14}$$

to obtain $\alpha = \alpha_{ref}$ in the next step. The stepsize suggested by (15) must be coordinated with the requirements from the error control. If $\alpha > \alpha_{ref}$ the stepsize in Gustafsson (1992) is implemented as

$$h_{n+1} = \min(h_r, h_\alpha) \tag{15}$$

restraining the stepsize if $h_\alpha < h_r$. The strategy adopted by (15) and (16) may be too aggressive in the sense that the corresponding error estimate (10) becomes very low compared to ϵ. Hence the subsequent stepsize estimated by the asymptotic controller

$$h_r = \left(\frac{\epsilon}{r_{n+1}}\right)^{1/\hat{k}} h_n \tag{16}$$

will be too large making the error estimate and thereby the stepsize fluctuate wildly. We try to avoid this by modifying (15) to

$$h_\alpha = \left(\frac{\alpha_{ref}}{\alpha}\right)^{1/\hat{k}} h_n \tag{17}$$

which means that the deviation of the convergence rate from α_{ref} is not necessarily corrected in one step. If a step has been rejected and restricted by slow convergence, then in combination (17) is filtered by the relation between the previous accepted step and the current accepted step

$$h_r = \frac{h_n}{h_{n-1}} \left(\frac{\epsilon}{r_{n+1}}\right)^{1/\hat{k}} h_n \tag{18}$$

which further reduces the stepsize following a convergence restricted step. If the current step is accepted we neglect the condition $\alpha > \alpha_{ref}$ on (16). In addition (14) is always used estimating the next stepsize, whenever a step is accepted. Consequently we allow the convergence of the equation solver to gain more influence on the stepsize selection. While Gustafsson (1992) suggests $0.2 \lesssim \alpha_{ref} \lesssim 0.5$ as set-points for the convergence rate we chose $\alpha_{ref} = 0.6$. This value favours robustness and a minimum amount of work needed to complete the integration fairly equal.
Slow convergence in the equation solver and in particular rejected steps because of convergence failure is very costly. This can to some extend be controlled by the stepsize but also by the frequency of Jacobian updates/factorizations. Considering (6) we see that stepsize changes invokes a refactorization of the Jacobian but not necessarily a Jacobian reevaluation - if on the other hand the Jacobian is updated a factorization is always called for. Good convergence can be obtained by both updating and factorizing the Jacobian at every stepsize change. For large systems though this may be the dominating part of the computations and large savings can be made by utilizing a strategy for reusing the same Jacobian for several timesteps. Gustafsson (1992) monitors the relative stepsize change since the last factorization was done and suggests

$$|h_{n+1} - h_{LU}|/h_{LU} > \alpha_{LU} \tag{19}$$

as a refactorization strategy. The strategy is preventive in the sense that it tries to avoid convergence failures by factorizing whenever planning to do a stepsize change that is likely to jeopardize convergence. Should poor convergence be experienced despite a factorization based on current data, say $\alpha > \alpha_{Jac}$, then a reevaluation of the Jacobian is called for. Gustafsson and Söderlind (1997) suggests the combination

$$\alpha - |h_{n+1} - h_{LU}|/h_{LU} > \alpha_{Jac} \tag{20}$$

as decision for when to compute a new Jacobian. Besides monitoring the convergence rate of the equation solver this strategy also trades Jacobian updates with factorizations and function evaluations. The value of α_{ref} sets an upper limit on α_{Jac} and α_{LU}, see Gustafsson (1992). In the two-phase flow problem the administration of the Jacobian is expensive compared to one iteration. This argues for large values of α_{Jac} and α_{LU}. To be more specific it is more costly to update the Jacobian than factorizing it, consequently we have chosen $\alpha_{Jac} = 0.5$ and $\alpha_{LU} = 0.3$. Because of the large value of α_{Jac} we allow a fairly large maximum number of iterations in the equation solver, setting $k_{\max} = 20$.

5.3 The complete controller

The complete modified PI controller for an implicit Runge-Kutta method is outlined in Algorithm 5.1. The controller includes three main parts:

- A stepsize selection rule based on both the error-tolerance relation and the convergence of the equation solver.
- An update/factorization strategy for the Jacobian that supervises the convergence and the iteration error of the equation solver.
- A strategy for handling convergence failures.

6. CHOICE OF METHODS

In this section the two-phase flow problem is used as a benchmark. We compare and discuss the performance of the controller by Gustafsson and Söderlind (1997) and the controller suggested in Algorithm 5.1 when applied to three different ESDIRK methods.
In this section ESDIRK$k\hat{k}$ refers to an ESDIRK method of

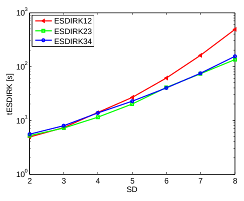
Algorithm 5.1: The complete modified PI controller for an implicit Runge-Kutta method.

```
if iterations converged then
    $h_r \leftarrow \left(\frac{\epsilon}{r}\right)^{1/\hat{k}} h$
    if step accepted then
        if step restricted then
            $h_r \leftarrow \frac{h}{h_{acc}} h_r$
        else
            $h_r \leftarrow \frac{h}{h_{acc}} \left(\frac{r_{acc}}{r}\right)^{1/\hat{k}} h_r$
        $r_{acc} \leftarrow r$
        $h_{acc} \leftarrow h$
    $h \leftarrow \min\left(h_r, \left(\frac{\alpha_{ref}}{\alpha}\right)^{1/\hat{k}} h\right)$
    if $\alpha - |h - h_{LU}|/h_{LU} > \alpha_{Jac}$ then
        Form new Jacobian and factorize iteration matrix.
        $h_{LU} \leftarrow h$
    else if $|h - h_{LU}|/h_{LU} > \alpha_{LU}$ then
        Factorize iteration matrix.
        $h_{LU} \leftarrow h$
else
    if new Jacobian then
        if $\alpha > \alpha_{ref}$ then
            $h \leftarrow \left(\frac{\alpha_{ref}}{\alpha}\right)^{1/\hat{k}} h$
        else
            $h \leftarrow h/2$
        Step restricted.
    else
        Form new Jacobian.
    Factorize iteration matrix.
    $h_{LU} \leftarrow h$
```

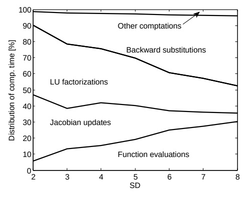
order k with an embedded method for error estimation of order $\hat{k}$. PI97 denotes the controller by Gustafsson and Söderlind (1997) and PI09 refers to the controller presented in Algorithm 5.1. For the work-precision diagrams we used a fixed absolute tolerance of 10^{-8} and the relative tolerances from 10^{-2} to 10^{-8}, denoted as significant digits (SD).

6.1 Choice of ESDIRK method

As can be seen from the work-precision diagram in Figure 3(a), the computational cost of ESDIRK12 increases dramatically with the requirement in SD's. This is due to the small stepsizes, which yields an increased workload of the equation solver trying to retain $(r_R)_i^k \leq \tau$ (12). This is seen in Figure 4(a), where the number of function evaluations reflects the number of iterations. ESDIRK23 and ESDIRK34 are better at maintaining an appropriate distribution of the workload as the requirements of the number in SD's increases. The distribution of workload of the two methods are almost identical and only the distribution of ESDIRK23 is depicted in Figure 3(b).
Except for SD $=$ 2 we observe from the work-precision diagram of the three methods, that ESDIRK23 is the most computationally efficient method for temporal discretization of problems like the two-phase flow.

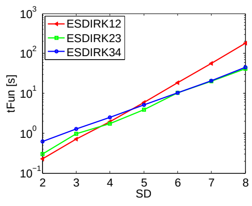


(a) Computational cost of the three ESDIRK methods.

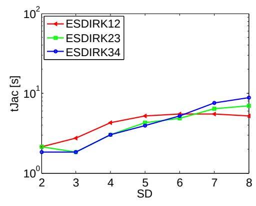


(b) Distribution of computational cost in ESDIRK23.

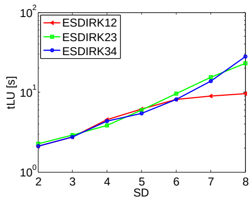
Fig. 3. Total computational cost of the three ESDIRK methods applied with the PI09 controller and the distribution of the computational cost in ESDIRK23.



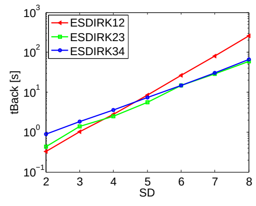
(a) Computational cost of function evaluations.



(b) Computational cost of Jacobian updates.



(c) Computational cost of LU factorizations.

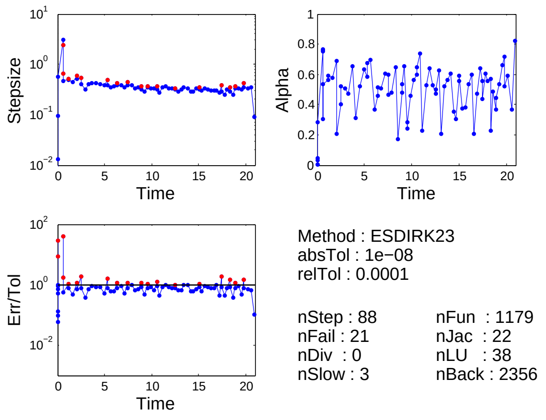


(d) Computational cost of back substitutions.

Fig. 4. Performance comparison of the three ESDIRK methods applied with the PI09 controller.

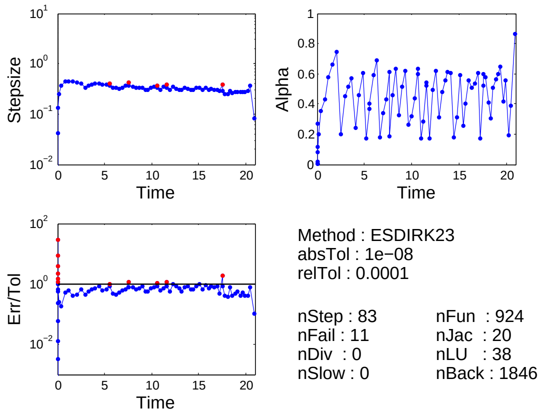
6.2 Choice of controller

The stepsize sequences for the PI97 and the PI09 controllers are depicted in Figure 5 and 6 respectively. As expected, we observe a reduction in rejected steps (nFail and nSlow) and fewer iterations done by the equation solver (nFun). In the PI97 controller, convergence is only allowed to restrict the stepsize, if $\alpha > \alpha_{ref}$. The PI09 controller allows convergence to restrict the stepsize by combining (16) and (18), hence the relation between α and α_{ref} is taken into account in each stepsize selection. Due to this improved interaction between the error and the convergence control in the stepsize selection process, large fluctuations of the stepsize, when advancing in time, is avoided. Consequently a smoother stepsize sequence is obtained and the need for heuristics to restrain large stepsize changes no longer applies.
As seen in Figure 7, it is difficult to make a general conclusion of the difference in computational cost for the two controllers. Typically we require 3 to 4 SD's in reservoir simulation, as a consequence we suggest applying the PI09 controller, when solving problems like the two-phase flow.



```
Method : ESDIRK23
absTol : 1e−08
relTol : 0.0001

nStep : 88        nFun  : 1179
nFail : 21        nJac  : 22
nDiv  : 0         nLU   : 38
nSlow : 3         nBack : 2356
```

Fig. 5. Performance of ESDIRK23 applied with the PI97 controller computing the solution in Figure 1(b).

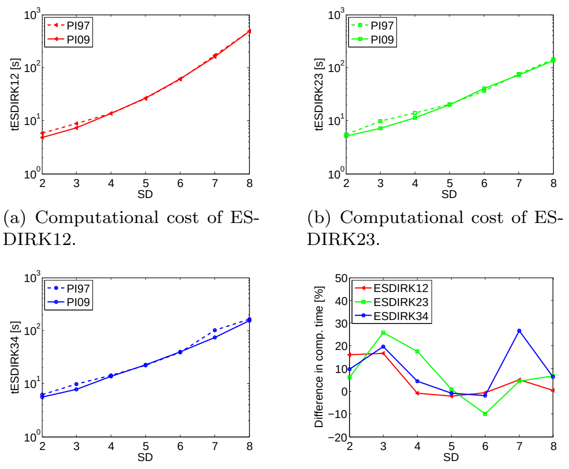


```
Method : ESDIRK23
absTol : 1e−08
relTol : 0.0001

nStep : 83        nFun  : 924
nFail : 11        nJac  : 20
nDiv  : 0         nLU   : 38
nSlow : 0         nBack : 1846
```

Fig. 6. Performance of ESDIRK23 applied with the PI09 controller computing the solution in Figure 1(b).

7. CONCLUSION

In this paper we combined the control of error with the convergence control of the equation solver in a simple logic that decreases the number of rejected steps and produces a smoother stepsize sequence. In some cases, better convergence of the equation solver is obtained i.e. fewer iterations is needed in order to meet the required tolerance. For large scale systems, which is typical in reservoir simulation, it may be necessary to solve the linearized equations iteratively. If this is the situation, the cost per iteration, both for the equation solver and the iterative solver of the linearized system, can be significant. Consequently, it is crucial for the solution of large scale systems to minimize the number of iterations per timestep, when performing implicit numerical integration.
In addition, the integration of the convergence control has the effect that extreme variations in stepsize are eliminated making the logics in the control algorithm free of heuristics.



(a) Computational cost of ESDIRK12.
(b) Computational cost of ESDIRK23.
(c) Computational cost of ESDIRK34.
(d) Difference in computational cost.

Fig. 7. Comparison of the PI97 and the PI09 controller applied to the three ESDIRK methods.

REFERENCES

Brouwer, D.R. and Jansen, J.D. (2004). Dynamic optimization of waterflooding with smart wells using optimal control theory spe-78278-pa. *The 2002 SPE European Petroleum Conference.*

Chen, Z. (2007). *Reservoir Simulation : Mathematical Techniques in Oil Recovery.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia.

Gustafsson, K. (1992). *Control of Error and Convergence in ODE Solvers.* Ph.D. thesis, Department of Automatic Control, Lund University, Sweden.

Gustafsson, K. and Söderlind, G. (1997). Control strategies for the iterative solution of nonlinear equations in ode solvers. *SIAM J. Sci. Comput.*, 18(1), 23–40.

Hairer, E. and Wanner, G. (1996). *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems.* Springer, 2nd edition.

Houbak, N., Nörsett, S., and Thomsen, P. (1985). Displacement or residual test in the application of implicit methods for stiff problems. *IMA Journal of Numerical Analysis*, 5(3), 297–305.

Jørgensen, J.B., Kristensen, M.R., and Thomsen, P.G. (2008). A family of esdirk integration methods. *SIAM Journal on Scientific Computing.*

Kennedy, C.A. and Carpenter, M.H. (2003). Additive runge-kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, 44(1-2), 139 – 181.

Kværnø, A. (2004). Singly diagonally implicit runge-kutta methods with an explicit first stage. *BIT Numerical Mathematics*, 44, 489 – 502.

Prothero, A. and Robinson, A. (1974). On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of Computation*, 28(125), 145–162.

Völcker, C., Jørgensen, J.B., Thomsen, P.G., and Stenby, E.H. (2009). Simulation of subsurface two-phase flow in an oil reservoir. *The European Control Conference 2009.*

APPENDIX F

Paper C

Explicit Singly Diagonally Implicit Runge-kutta Methods and Adaptive Stepsize Control for Reservoir Simulation

Authors:
Carsten Völcker, John Bagterp Jørgensen, Per Grove Thomsen and Erling Halfdan Stenby

Published in:
Proceedings of *12th European Conference on the Mathematics of Oil Recovery (ECMOR XII)*
Oxford, UK
6-9 September 2010

ECMOR XII

Explicit Singly Diagonally Implicit Runge-kutta Methods and Adaptive Stepsize Control for Reservoir Simulation

Carsten Völcker[*] John Bagterp Jørgensen[*]
Per Grove Thomsen[*] Erling Halfdan Stenby[**]

[*] *Department of Informatics and Mathematical Modeling, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark (e-mail:* {cv,jbj,pgt}@imm.dtu.dk*)*
[**] *Department of Chemical and Biochemical Engineering, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark (e-mail:* ehs@kt.dtu.dk*)*

Abstract

The implicit Euler method, normally refered to as the fully implicit (FIM) method, and the implicit pressure explicit saturation (IMPES) method are the traditional choices for temporal discretization in reservoir simulation. The FIM method offers unconditionally stability in the sense of discrete approximations, while the IMPES scheme benefits from the explicit treatment of the saturation. However, in tems of controlling the integration error, the low order of the FIM method leads to small integration steps, while the explicit treatment of the saturation may restrict the stepsizes for the IMPES scheme. Current reservoir simulators apply timestepping algorithms that are based on safeguarded heuristics, and can neither guarantee convergence in the underlying equation solver, nor provide estimates of the relations between convergence, integration error and stepsizes.
We establish predictive stepsize control applied to high order methods for temporal discretization in reservoir simulation. The family of Runge-Kutta methods is presented and in particular the explicit singly diagonally implicit Runge-Kutta (ESDIRK) method with an embedded error estimate is described. A predictive stepsize adjustment rule based on error estimates and convergence control of the integrated iterative solver is presented. We try to improve the predictive stepsize control through an extended communication between the convergence rate, the error control and the stepsize.

Keywords: Reservoir simulation, implicit Runge-Kutta methods, ESDIRK, Newton-Raphson, convergence control, error control, stepsize selection.

Introduction

Reservoir simulators are computer programs that solve the equations for heat and mass flow in porous media. Numerical integration is one of the basic steps involved in the simulation process. The number and type of equations to be solved depend on the geological characteristics of the reservoir, the characteristics of the reservoir fluids, and the oil recovery process to be modeled. Choosing the appropriate method of integration involves deciding on factors such as the order of the integration scheme, stability properties, and concern on computational efficiency. ESDIRK methods have been applied successfully for solution of convection-diffusion-reaction problems (Kennedy and Carpenter (2003)). This class of methods is computationally efficient, and both A- and L-stable stiffly accurate ESDIRK methods of various order, with an embedded method for error estimation, have been derived by Kværnø (2004) and Jørgensen et al. (2008). In addition, a robust adaptive stepsize selection is essential to an efficient numerical integration. An adaptive stepsize selection aims to keep the error estimate bounded i.e. close to a user-specified tolerance by adjusting the timestep. The nonlinear residual equations arising in fully implicit methods has to be solved iteratively. Thus in implicit integration the convergence in the equation solver has to be monitored. Gustafsson (1992) suggested a strategy for stepsize selection based on the error estimates provided by embedded Runge-Kutta methods.

We have applied the controller by Gustafsson and Söderlind (1997) to three different ESDIRK methods used for solving a two-phase reservoir model. Although the control strategy has proven efficient we observed that certain steps were rejected due to irregularities in the stepsize selection. We found that an extended use of previous accepted stepsizes in terms of filtering, a less aggressive stepsize suggestion from the convergence controller, and different interaction between the error and the convergence control in the stepsize selection process may solve this problem. The idea is to combine the control of error with the control of convergence in the equation solver such that we obtain a smoother stepsize sequence. This minimizes the number of rejected steps and produces a smoother stepsize sequence and thereby improves the efficiency of the implicit integration.

Differential Equation Model

In this section we briefly outline the two-phase flow problem and we present the general formulation of a system of differential equations based on conservation laws.

The Two-Phase Flow Problem

We consider immiscible two-phase flow of oil and water in porous media. Let $P_o = P_o(t, x)$ be the pressure of oil and $S_w = S_w(t, x)$ be the saturation of water, as function of time $t \geq 0$ and position $x \in \mathbb{R}^3$, and let $C_w = C_w(P_o, S_w)$ and $C_o = C_o(P_o, S_w)$ be the mass concentrations of water and oil respectively. Then the mass balances for water and oil in the reservoir is expressed by the following system of partial differential equations

$$\frac{\partial}{\partial t} C_w = -\nabla \cdot F_w + Q_w \tag{1a}$$
$$\frac{\partial}{\partial t} C_o = -\nabla \cdot F_o + Q_o \tag{1b}$$

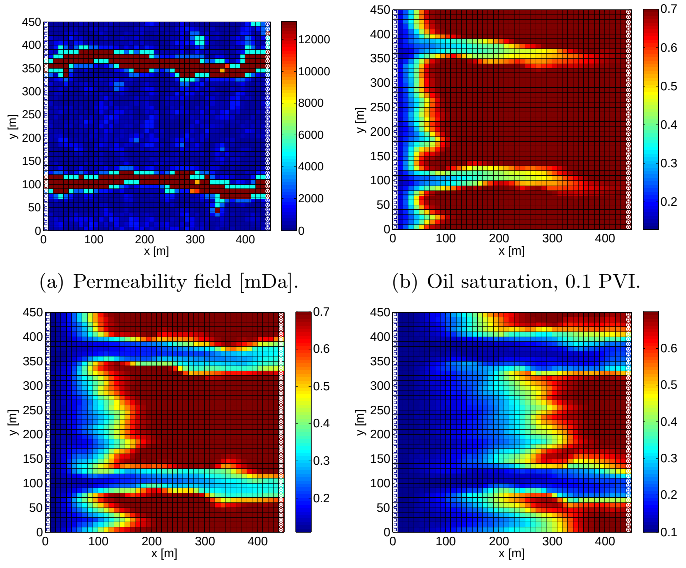
$F_w = F_w(P_o, S_w)$ and $F_o = F_o(P_o, S_w)$ are the fluxes of water and oil through the porous media. The source/sink terms of water and oil are denoted $Q_w = Q_w(P_o, S_w)$ and $Q_o = Q_o(P_o, S_w)$. They are used to describe the flow from injection wells and the flow to production wells. A more profound description can be found in Chen (2007) and Völcker et al. (2009). We use a standard 2-D problem defined by Brouwer and Jansen (2004), depicted in Figure 1.

General Formulation

Many process simulation problems in general are based on conservation of mass, energy and momentum. It is desirable to preserve such properties upon numerical integration in time. As proposed by Völcker et al. (2009) a mass preserving general formulation of such an ODE system may be

$$\frac{d}{dt} g(x(t)) = f(t, x(t)) \quad x(t_0) = x_0 \tag{2}$$

in which $x(t) \in \mathbb{R}^m$ denotes the system states, $g(x(t)) \in \mathbb{R}^m$ are the properties conserved, while the right-hand side function $f(t, x(t)) \in \mathbb{R}^m$ has the usual interpretation. The spatially discretized two-phase flow problem considered has the structure of (2).



(a) Permeability field [mDa]. (b) Oil saturation, 0.1 PVI. (c) Oil saturation, 0.3 PVI. (d) Oil saturation, 1 PVI.

Figure 1 *Water flooding example, 1 pore volume injected (PVI) over 360 days.*

Integration Methods

In this section different classes of Runge-Kutta methods are outlined. In particular ESDIRK methods are described.

Runge-Kutta Integration

An s-stage Runge-Kutta method for integration of (2) can be expressed as

$$T_i = t_n + h_n c_i \qquad i \in \mathbb{S}_1 \tag{3a}$$

$$g(X_i) = g(x_n) + h_n \sum_{j=1}^{s} a_{ij} f(T_j, X_j) \qquad i \in \mathbb{S}_1 \tag{3b}$$

$$g(x_{n+1}) = g(x_n) + h_n \sum_{j=1}^{s} b_j f(T_j, X_j) \tag{3c}$$

where X_i are the internal stage values being numerical approximations to $x(T_i)$. x_{n+1} is the step computed at $t_{n+1} = t_n + h_n$. The set $\mathbb{S}_i$ denotes the internal stages $i, i+1, \ldots, s$. The s-stage Runge-Kutta method (3) may be denoted in terms of its Butchter tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline x_{n+1} & b_1 & b_2 & \cdots & b_s \end{array} \tag{4}$$

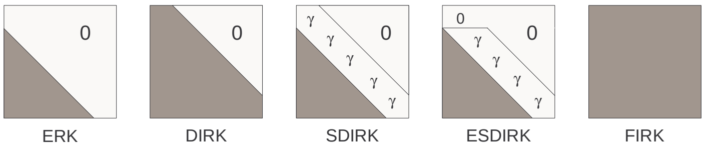


Figure 2 *The A-matrix of Runge-Kutta methods.*

from which different classes of Runge-Kutta methods can be obtained, depending on the structure of the matrix $A = [a_{ij}]$. This is illustrated in Figure 2. Explicit Runge-Kutta (ERK) methods have a strictly lower triangular A-matrix which allows all internal stages (3b) to be solved explicitly. Therefore, ERK methods are computationally fast and straightforward to implement but may suffer from stability limitations making them unsuitable for stiff problems (Hairer and Wanner (1996)). The four remaining classes of Runge-Kutta methods are all implicit, that is, the value of the internal stages are no longer computed explicitly from the values of the previous stages. Implicit methods are characterized by an A-matrix that is not strictly lower triangular and the stage values X_i are computed iteratively by solution of (3b). Fully implicit Runge-Kutta (FIRK) methods, identified by a full A-matrix, have excellent stability properties making them usefull for solving stiff systems of ODE's. However, the excellent stability properties comes with high computational cost in the sense that each integration step involves the solution of ms coupled nonlinear equations. To achieve some of the stability properties of the FIRK methods but at lower computational cost, various methods in between the ERK and the FIRK methods have been constructed. Diagonally implicit Runge-Kutta (DIRK) methods, singly diagonally implicit Runge-Kutta (SDIRK) methods and ESDIRK methods all have a lower triangular A-matrix. Instead of solving ms nonlinear equations simultaneously, like in the FIRK method, the internal stages in the DIRK, SDIRK and ESDIRK methods are decoupled in such a way that the solution of s systems of m nonlinear equations may be conducted sequentially.

ESDIRK Methods

ESDIRK methods have a lower triangular A-matrix. By construction they retain the stability properties of FIRK methods but at significant lower computational cost. Because $c_1 = 0$ and $a_{11} = 0$ the first stage in ESDIRK methods is explicit implying that the first stage value equals the last step, i.e. $(T_1, X_1) = (t_n, x_n)$. The subsequent stages are diagonally implicit such that the stage values X_i at $T_i = t_n + h_n c_i$ for $i \in \mathbb{S}_2$ may be solved sequentially by solution of the residual

$$R(T_i, X_i) = g(X_i) - h_n \gamma f(T_i, X_i) - \psi_i = 0 \quad i \in \mathbb{S}_2 \tag{5}$$

with the term

$$\psi_i = g(x_n) + h_n \sum_{j=1}^{i-1} a_{ij} f(T_j, X_j) \quad i \in \mathbb{S}_2 \tag{6}$$

using Newton-Raphson's iterative method. The Jacobian $J_R(T_i, X_i) = \frac{\partial}{\partial X_i} R(T_i, X_i)$ of the residual (5) is

$$J_R(T_i, X_i) = J_g(X_i) - h_n \gamma J_f(T_i, X_i) \quad i \in \mathbb{S}_2 \tag{7}$$

in which $J_f(T_i, X_i) = \frac{\partial}{\partial X_i} f(T_i, X_i)$ and $J_g(X_i) = \frac{d}{dX_i} g(X_i)$ are the Jacobiants of the right- and left-hand sides of (2) respectively. Since ESDIRK methods are singly diagonally, the Jacobian may be reused in the sense of a modified Newton-Raphson. The identical diagonal elements in the A-matrix implies that (7) only needs to be updated/factorized once per integration step. We only consider methods assumed to be stiffly accurate by construction, i.e. $c_s = 1$ and $a_{sj} = b_j$ for $j \in \mathbb{S}_1$. This implies that the quadrature function (3c) corresponds to the last internal stage in (3b). Consequently the next step equals the last stage value, i.e. $(t_{n+1}, x_{n+1}) = (T_s, X_s)$. The Butcher tableau for stiffly accurate ESDIRK methods is represented in (8).

$$\begin{array}{c|cccccc}
0 & 0 & & & & & \\
c_2 & a_{21} & \gamma & & & & \\
c_3 & a_{31} & a_{32} & \gamma & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & & \\
c_{s-1} & a_{s-1,1} & a_{s-1,2} & a_{s-1,3} & \cdots & \gamma & \\
1 & b_1 & b_2 & b_3 & \cdots & b_{s-1} & \gamma \\
\hline
x_{n+1} & b_1 & b_2 & b_3 & \cdots & b_{s-1} & \gamma
\end{array} \tag{8}$$

Error and Convergence Measures

In this section we describe how to estimate the integration error, how the error is related to the user specified tolerance and how to estimate the convergence rate of the iterative solver.

Integration Error

The ESDIRK method stated in (8) may be equipped with an embedded Runge-Kutta method

$$g(\hat{x}_{n+1}) = g(x_n) + h_n \sum_{j=1}^{s} \hat{b}_j f(T_j, X_j) \tag{9}$$

computing the embedded solution $\hat{x}_{n+1}$. The Butcher tableau for embedded ESDIRK methods takes the form

$$\begin{array}{c|cccccc}
0 & 0 & & & & & \\
c_2 & a_{21} & \gamma & & & & \\
c_3 & a_{31} & a_{32} & \gamma & & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & & \\
c_{s-1} & a_{s-1,1} & a_{s-1,2} & a_{s-1,3} & \cdots & \gamma & \\
1 & b_1 & b_2 & b_3 & \cdots & b_{s-1} & \gamma \\
\hline
x_{n+1} & b_1 & b_2 & b_3 & \cdots & b_{s-1} & \gamma \\
\hat{x}_{n+1} & \hat{b}_1 & \hat{b}_2 & \hat{b}_3 & \cdots & \hat{b}_{s-1} & \hat{b}_s \\
\hline
e_{n+1} & d_1 & d_2 & d_3 & \cdots & d_{s-1} & d_s
\end{array} \tag{10}$$

The embedded method is of different order, which then provides an estimate of the local truncation error

$$e_{n+1} = g(x_{n+1}) - g(\hat{x}_{n+1}) = h_n \sum_{j=1}^{s} d_j f(T_j, X_j) \tag{11}$$

corresponding to the numerical solution x_{n+1}. It should be noted that e_{n+1} is an error estimate of the conserved quantities $g(x_{n+1})$ and not the states x_{n+1} themselves. Measures of the error such as

$$r_{n+1} = \frac{1}{\sqrt{m}} \left\| \frac{|e_{n+1}|}{\text{abstol} + |g(x_{n+1})| \cdot \text{reltol}} \right\|_2 \tag{12a}$$

(12b)

$$r_{n+1} = \left\| \frac{|e_{n+1}|}{\text{abstol} + |g(x_{n+1})| \cdot \text{reltol}} \right\|_\infty \tag{12c}$$

may be controlled adjusting the timestep in such a way that only stepsizes for which the error-tolerance relation $r_{n+1} \leq 1$ are accepted.

Convergence Rate

The solution of the residual function (5) is done iteratively by modified Newton-Raphson iterations. The controller needs to supervise the equation solver and make decisions like: when should the Jacobian be evaluated/factorized, and what restrictions should be put on the stepsize to assure convergence? In practice the convergence rate is measured. For reasons of robustness the convergence rate is measured by the residuals (Houbak et al. (1985))

$$\alpha = \max_{i,k} \frac{(r_R)_i^{k-1}}{(r_R)_i^k} \qquad i \in \mathbb{S}_2 \tag{13}$$

in which the iteration error of the k^{th} iteration is computed as the residual-tolerance relation

$$(r_R)_i^k = \frac{1}{\sqrt{m}} \left\| \frac{|(R(T_i, X_i))^k|}{\text{abstol} + |(g(X_i))^k| \cdot \text{reltol}} \right\|_2 \qquad i \in \mathbb{S}_2 \tag{14a}$$

(14b)

$$(r_R)_i^k = \left\| \frac{|(R(T_i, X_i))^k|}{\text{abstol} + |(g(X_i))^k| \cdot \text{reltol}} \right\|_\infty \qquad i \in \mathbb{S}_2 \tag{14c}$$

using the same componentwise absolute and relative error tolerances as in (12). For robustness the iteration sequence should be contractive. If for some k during the iterations $\alpha \geq 1$ the iteration sequence is terminated. In the event of termination either a Jacobian update/factorization is called for or the stepsize is restricted. In case of convergence, the equation solver is successfully stopped when $(r_R)_i^k \leq \tau$. As noticed in Hairer and Wanner (1996) the choice of τ affects the efficiency of the algorithm. A large value of τ may lead to one or more large components in the integration error (12) with too many rejected steps as a result. We have chosen $\tau = 0.1$ as a compromise between robustness and computational speed.

Stepsize Selection

This section is divided into a brief description of the stepsize selection rule adopted and a description of the convergence control and the Jacobian evaluation/factorization strategy.

Predictive Error Control

The objective of adaptive timestepping is to produce numerical solutions for which the integration error (11) is kept within the error tolerance. This implies choosing stepsizes small enough such that $r_{n+1} \leq \epsilon$, and at the same time large enough in the sense of minimizing the computational cost in the solution process. The asymptotic stepsize selection rule is as follows

$$h_r = \left(\frac{\epsilon}{r_{n+1}}\right)^{1/\hat{k}} h_n \tag{15}$$

in which $\hat{k}$ is the order of the embedded Runge-Kutta method. The asymptotic stepsize selection rule uses nothing else than current information about the stepsize and the error in order to achieve $r_{n+1} = \epsilon$. Occasionally, the error estimate may be unusually small or large, thus advocating (15) to produce very small or large stepsize changes. This can to some extend be avoided by including some limitations on such changes. However, information about previous timesteps should also be exploited in the controller to further improve robustness and the quality of the predicted timestep. Gustafsson (1992) suggests a proportional integral (PI) stepsize adjustment rule on the form

$$h_r = \frac{h_n}{h_{n-1}} \left(\frac{r_n}{r_{n+1}}\right)^{k_1/\hat{k}} \left(\frac{\epsilon}{r_{n+1}}\right)^{k_2/\hat{k}} h_n \tag{16}$$

where k_1 and k_2 are the gain parameters of the proportional and the integral parts respectively, $\hat{k}$ is the order of the embedded Runge-Kutta method, while ϵ is the set point of the error-tolerance relation. In theory $\epsilon = 1$ is an acceptable threshold. However, in the practical implementation we have chosen 0.8 as a safeguard. Gustafsson (1992) suggests $k_1 = k_2 = 1$ corresponding to deadbeat control.

Convergence Control

In some situations the stepsize has to be restrained in order to assure convergence in the equation solver. When using modified Newton-Raphson the convergence is normally secured by frequent Jacobian updates/factorizations. Thus the Jacobian is an approximation computed at one solution point, and the distance between consecutive solution points may be large enough to jeopardize convergence. Besides frequent Jacobian updates/factorizations the stepsize is the only available control variable affecting the convergence rate in the equation solver. Decreasing the distance between the different solution points by stepsize reduction may secure convergence. If convergence is too slow, i.e. if $\alpha > \alpha_{ref}$, Gustafsson (1992) suggests the stepsize to be chosen as

$$h_\alpha = \frac{\alpha_{ref}}{\alpha} h_n \tag{17}$$

to achieve $\alpha = \alpha_{ref}$ in the next step. The stepsize suggested by (17) must be coordinated with the requirements from the error control. Convergence may be a more restrictive constraint than accuray, and it even may occur that convergence is poor in spite of a Jacobian based on current data. If this is the case and if $\alpha > \alpha_{ref}$ at the same time, then the stepsize is implemented as

$$h_{n+1} = \min(h_r, h_\alpha) \tag{18}$$

restricting the stepsize by convergence if $h_\alpha < h_r$. The effeciency of an implicit integration method depends highly on the convergence in the equation solver, i.e. the value of the set point α_{ref}. If the two conditions described above are satisfied simultaneously, the controller invokes (18) in an attempt to obtain $\alpha = \alpha_{ref}$. Gustafsson (1992) recommends, based on an idea due to Söderlind (1986), any value $0.2 < \alpha_{ref} < 0.5$ as reference for the convergence rate, with robustness favoring the lower values. More precisely he suggests the set point $\alpha_{ref} = 0.4$, which is the value that we use in our implementation.
Slow convergence in the equation solver can to some extend be avoided by stepsize reductions, but also by frequent of Jacobian evaluations/factorizations. Considering (7) we see that stepsize changes may invoke a refactorization of the Jacobian but not necessarily a Jacobian reevaluation. On the other hand, if the Jacobian is updated a factorization is always called for. Good convergence can be obtained by both updating and factorizing the Jacobian at every stepsize change. For large systems though this may be the dominating part of the computations and large savings can be made by utilizing a strategy for reusing the same Jacobian for several timesteps. The relative stepsize change is monitored since the last factorization was done by the relation

$$|h_{n+1} - h_{LU}|/h_{LU} > \alpha_{LU} \tag{19}$$

as a refactorization strategy. The strategy anticipates possible convergence failures and refactorizes the Jacobian whenever a suitable, planned stepsize change is likely to jeopardize convergence. Should poor convergence be experienced despite a factorization based on current data, say $\alpha > \alpha_{Jac}$, then a reevaluation of the Jacobian is called for. Gustafsson and Söderlind (1997) suggests the combination

$$\alpha - |h_{n+1} - h_{LU}|/h_{LU} > \alpha_{Jac} \tag{20}$$

as decision for when to compute a new Jacobian. Besides monitoring the convergence rate of the equation solver this strategy also trades Jacobian updates with factorizations and function evaluations. The maximum reasonable value of α_{ref} sets an upper limit on α_{Jac} and α_{LU}. In other words, it is of no use having a Jacobian update/factorization strategy that accepts a convergence rate worse than what is known to be effecient.
In the two-phase flow problem the administration of the Jacobian is expensive compared to one iteration. Though we have chosen $\alpha_{Jac} = 0.2$ and $\alpha_{LU} = 0.2$. This choice seems like a good balance between the computational load of the equation solver compared to the total computational cost of Jacobian evaluations/factorizations, thus minimizing the total computational time spent in the solution process. The maximum number of iterations allowed in the equation solver is set to 10, i.e. $k_{\max} = 10$.

Modifying the Control Algorithm

In this section we present our modifications of the control algorithm and finally an outline of the complete controller is presented.

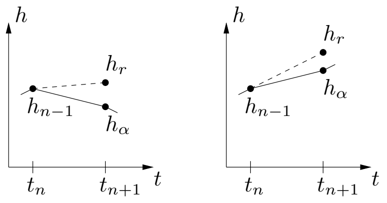


Figure 3 *Stepsizes restricted by convergence, i.e.* $h_\alpha < h_r$.

Controller Modifications

The core stepsize adjustment rule (16) must be implemented along with a number of extensions and various safety nets and the original framework from which we propose our modifications can be found in Gustafsson (1992). The modified PI controller that we suggest is presented in Algorithm 1.

Since we are only considering stiffly accurate ESDIRK methods, the order reduction for stiff systems can be avoided (Prothero and Robinson (1974)). Consequently, the strategy described by Gustafsson (1992) for estimating $\hat{k}$ after successive rejects can be omitted. This does not make any noticeable change in the controller performance but simplifies the algorithm a great deal.

The strategy adopted by (17) and (18) may be too aggressive. Gustafsson (1992) denotes a stepsize as restricted in two situations: if $\alpha > \alpha_{ref}$ in spite of a Jacobian based on current data, and if the iterations do not converge either because $\alpha \geq 1$ or because the maximum number of iterations is reached. Whenever a stepsize is restricted by convergence, i.e. $h_\alpha < h_r$, the asymptotic stepsize selection rule (15) is utilized. Convergence restricting a stepsize leads to an error estimate (12) much below ϵ, thus advocating (15) to produce a very large stepsize. Compared to the sequence of stepsizes that corresponds to past solution points, this stepsize may be too large, making the error estimates fluctuate wildly. We adopt some changes in both the error and the convergence control in order to anticipate this behaviour. The stepsize selection based on convergence (17) is modified to

$$h_\alpha = \left(\frac{\alpha_{ref}}{\alpha}\right)^{1/\hat{k}} h_n \tag{21}$$

such that the deviation of the convergence rate from α_{ref} is not necessarily corrected in one step. In addition the asymptotic stepsize selection rule (15) is filtered by the relation between the previous accepted stepsize and the current accepted stepsize

$$h_r = \frac{h_n}{h_{n-1}} \left(\frac{\epsilon}{r_{n+1}}\right)^{1/\hat{k}} h_n \tag{22}$$

such that it supports the trend of either a decreasing or an increasing restricted stepsize. This is illustrated in Figure 3: on the left showing an accepted decreasing restricted stepsize, and on the right showing an accepted increasing restricted stepsize.

In order to secure convergence Gustafsson (1992) suggests two conditions on deciding if a stepsize has to be restricted. If the Jacobian is based on current data and at the same time $\alpha > \alpha_{ref}$, then the stepsize should be restricted by (21) and (18). In most cases the Jacobian is not updated before the condition $\alpha > \alpha_{ref}$ has occured. Furthermore, this update is only carried into effect on the consecutive stepsize, and a stepsize executed with a Jacobian based on current

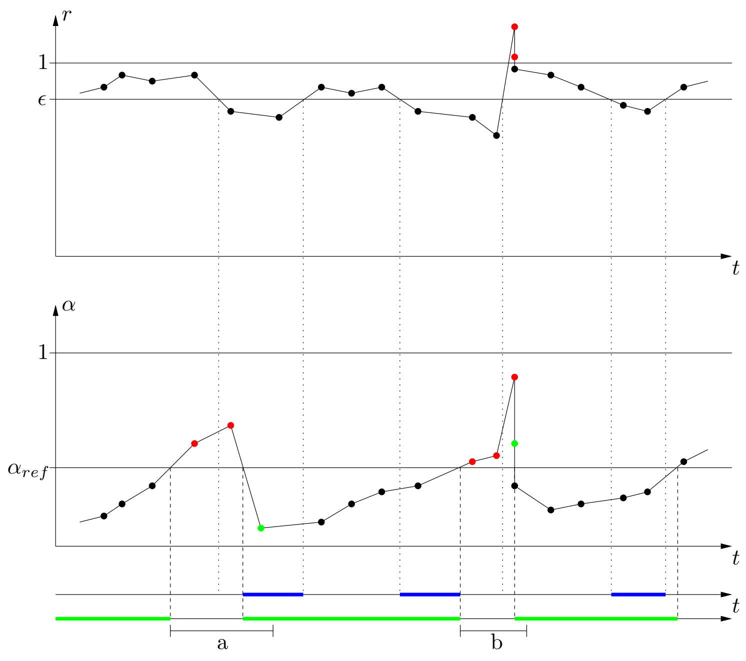


***Figure 4** A typical scenario showing the correlation between the error-tolerance relation* (12) *(upper plot) and the estimate of the convergence rate* (13) *(lower plot).*

data mostly results in good convergence, i.e. $\alpha \leq \alpha_{ref}$. Consequently, convergence restriction is not invoked very often and (21) may only have little effect on the stepsize selection process. The situation is depicted in region "a" in Figure 4. The red solution points satisfy the condition $\alpha > \alpha_{ref}$, but the Jacobian is an approximation computed at a previous solution point. The Jacobian at the green solution point is based on current data, which on the other hand leads to good convergence, i.e. $\alpha \leq \alpha_{ref}$. So none of the solution points in region "a" satisfies both conditions simultaneously. In region "b" in Figure 4 we try to illustrate a situation where both conditions are satisfied. The topmost red solution point in the lower plot represents a rejected step (as indicated by the corresponding upper red solution point in the upper plot where $r_{n+1} > 1$) that satisfies the condition $\alpha > \alpha_{ref}$ only. The controller calls for a Jacobian update/factorization and proceeds with the same stepsize. The step may be rejected once again, which leads us to the green solution point that actually satisfies both conditions. So the green solution point in region "b" indicates a situation where the stepsize is convergence restricted by (21) and (18). The two conditions suggested by Gustafsson (1992) only allows convergence to restrain the stepsize, and only when the Jacobian is based on current data. This implies that (21) is prevented to cause undesirable stepsize decrements, even if convergence is above the reference value, i.e. $\alpha > \alpha_{ref}$. This is a very important feature, in particular if the error is below the desired set point at the same time, i.e. $r_{n+1} < \epsilon$. It may be advantageous though if (21) had the opportunity to increase the stepsize as well. If the error estimate is low in comparison to the tolerance, then

$$\frac{\epsilon}{r_{n+1}} > 1 \tag{23}$$

which essentially provides the error control the potential to increase the stepsize. In situations like this it could be beneficial to allow convergence to increase the stepsize as well. This may be possible if

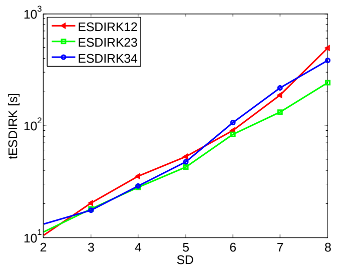
$$\frac{\alpha_{ref}}{\alpha} > 1 \tag{24}$$

at the same time. If we require both (23) and (24) to be satisfied before we let convergence increase the stepsize, then (21) will also become an active part in the stepsize selection in the blue sections in Figure 4. A stepsize increment must not jeopardize the integration error. So the stepsize suggested by (21) has to be coordinated with the one from the error control. Since (18) secures the integration error such that the error control tries to maintain the condition $r_{n+1} < \epsilon$, we may omit condition (23) and only require $\alpha < \alpha_{ref}$ to be satisfied. By doing so we let (21) increase the stepsize in the green sections in Figure 4.

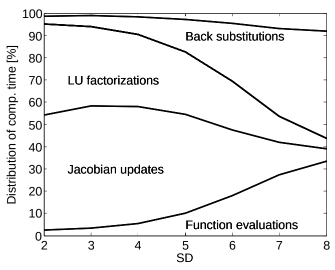
Algorithm 1: The complete modified PI controller for an implicit Runge-Kutta method.

```
if iterations converged then
    $h_r \leftarrow \left(\frac{\epsilon}{r}\right)^{1/\hat{k}} h$
    if step accepted then
        $h_r \leftarrow \frac{h}{h_{acc}} h_r$
        if step not restricted then
            $h_r \leftarrow \left(\frac{r_{acc}}{r}\right)^{1/\hat{k}} h_r$
        $r_{acc} \leftarrow r$
        $h_{acc} \leftarrow h$
    if new Jacobian and $\alpha > \alpha_{ref}$ or $\alpha < \alpha_{ref}$ then
        $h \leftarrow \min\left(h_r, \left(\frac{\alpha_{ref}}{\alpha}\right)^{1/\hat{k}} h\right)$
    else
        $h \leftarrow h_r$
    if $\alpha - |h - h_{LU}|/h_{LU} > \alpha_{Jac}$ then
        Form new Jacobian and factorize iteration matrix.
        $h_{LU} \leftarrow h$
    else if $|h - h_{LU}|/h_{LU} > \alpha_{LU}$ then
        Factorize iteration matrix.
        $h_{LU} \leftarrow h$
else
    if new Jacobian then
        if $\alpha > \alpha_{ref}$ then
            $h \leftarrow \left(\frac{\alpha_{ref}}{\alpha}\right)^{1/\hat{k}} h$
        else
            $h \leftarrow h/2$
    else
        Form new Jacobian.
    Factorize iteration matrix.
    $h_{LU} \leftarrow h$
```

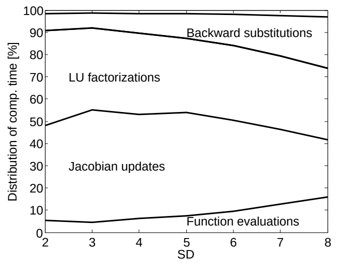
The modifications that we suggest to the original framework are as follows: the strategy for estimating $\hat{k}$ in case of successive rejects is omitted, the stepsize suggested by convergence (17) is made less aggressive by (21), the asymptotic stepsize selection rule (15) is filtered by (22),



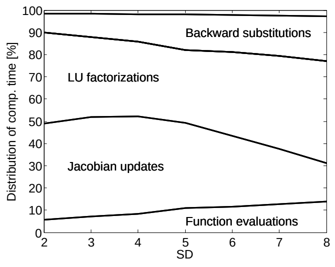
(a) Total computational cost.



(b) Workload distribution, ESDIRK12.



(c) Workload distribution, ESDIRK23.



(d) Workload distribution, ESDIRK34.

Figure 5 *P10 controller: total computational cost and distribution of workload (the uppermost patch in the diagrams is overhead time).*

and convergence is allowed to increase the stepsize if $\alpha < \alpha_{ref}$.

The Complete Controller

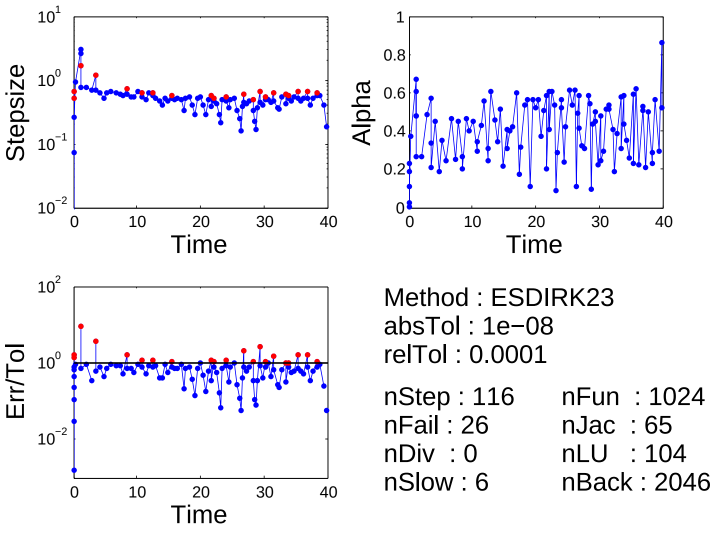
The complete modified PI controller for an implicit Runge-Kutta method is outlined in Algorithm 1. The controller includes four main parts: a stepsize selection based on error control, a strategy that coordinates the stepsize suggested by (21) with the one from the error control, a Jacobian update/factorization strategy monitoring convergence in the equation solver, and a strategy for handling convergence failures.

Choice of methods

In this section the two-phase flow problem (1) is used as a benchmark. We compare and discuss the performance of the controller by Gustafsson and Söderlind (1997) and the controller suggested in Algorithm 1, when applied to three different ESDIRK methods.
In this section ESDIRK$k\hat{k}$ refers to an ESDIRK method of order k with an embedded method for error estimation of order $\hat{k}$. PI97 denotes the controller by Gustafsson and Söderlind (1997) and PI10 refers to the controller presented in Algorithm 1. For the work-precision diagrams we used a fixed absolute tolerance of 10^{-8} and relative tolerances in the range from 10^{-8} to 10^{-2}. We denote the relative tolerances as significant digits (SD), such that SD $= 4$ corresponds to the relative tolerance 10^{-4}. The count of function evaluations, which some of the figures refers to, relates directly to the number of iterations done in the equation solver.

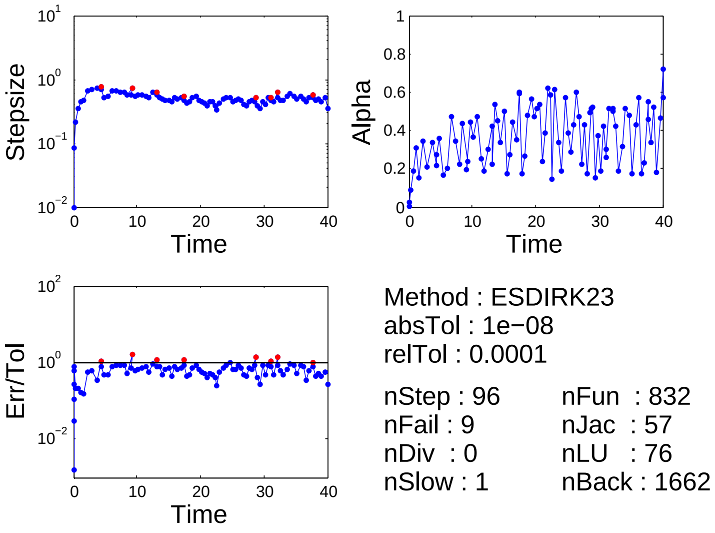
Choice of ESDIRK Method

From the work-precision diagram in Figure 5(a), we see that the total computational cost of ESDIRK12 increases in a more profound way than the other two methos. This is due to the



```
Method : ESDIRK23
absTol : 1e−08
relTol : 0.0001

nStep : 116      nFun  : 1024
nFail : 26       nJac  : 65
nDiv  : 0        nLU   : 104
nSlow : 6        nBack : 2046
```

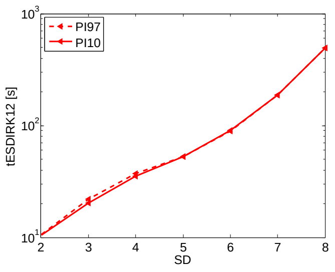
Figure 6 *PI97 controller: performance of ESDIRK23 when computing the solution shown in Figure 1(b).*



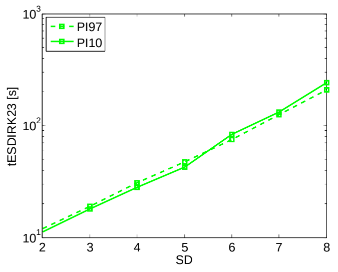
```
Method : ESDIRK23
absTol : 1e−08
relTol : 0.0001

nStep : 96       nFun  : 832
nFail : 9        nJac  : 57
nDiv  : 0        nLU   : 76
nSlow : 1        nBack : 1662
```

Figure 7 *PI10 controller: performance of ESDIRK23 when computing the solution shown in Figure 1(b).*

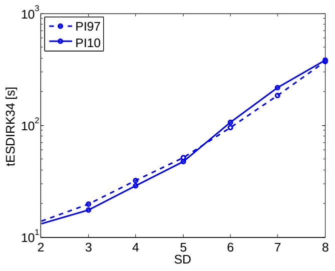
small stepsizes, which are necessary for the method in order to satisfy the required accuracy of the solution. This implies an increased workload of the equation solver when trying to retain $(r_R)_i^k \leq \tau$. In addition, small stepsizes leads to an increase in overhead time, which can be seen in Figure 5(b). As can be seen in Figure 5(c) and Figure 5(d), ESDIRK23 and ESDIRK34 are better at maintaining an appropriate distribution of the workload as the requirements in accuracy increases. Except for SD = 2, we can observe from the work-precision diagram of the three methods, that ESDIRK23 is overall the most computationally efficient method for temporal discretization of the two-phase flow problem (1).



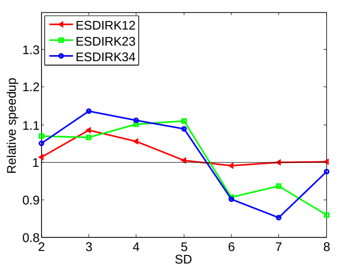
(a) Totsl computational cost, ESDIRK12.



(b) Computational cost of ESDIRK23.



(c) Computational cost of ESDIRK34.



(d) Relative speedup, $\frac{t_{\mathrm{PI10}}}{t_{\mathrm{PI97}}}$.

Figure 8 *Comparison of the PI97 and the PI10 controller applied to the three ESDIRK methods.*

Choice of Controller

The stepsize sequences for the PI97 and the PI10 controllers are depicted in Figure 6 and in Figure 7 respectively. It must be mentioned that `nStep`, `nFail` and `nSlow` refers to the number of timesteps used in order to obtain the solution, the number of steps rejected by the error control, i.e. $r_{n+1} > 1$, and the amount of steps rejected because the maximum number of iterations is reached. By comparing the performance of the two controllers, we see that fewer Jacobian evaluations/factorizations is required by the PI10 controller in the sense of maintaining good convergence in the equation solver. Thus fewer iterations is necessary in order to complete the integration. Because of the less aggressive stepsize change suggested by (21) in connection with the filtering of the asymptotic stepsize selection rule, a smoother stepsize sequence is obtained by the PI10 controller. This implies that the PI10 controller provides a solution with lesser rejected steps when compared to the performance of the PI97 controller. In Figure 6 several solution points for which $r_{n+1} \ll \epsilon$ ca be observed. For the same solution points the corresponding stepsizes are small. Due to the ability of (21) to increase the stepsize as well, this behaviour is only to a lesser extend observed in Figure 7. Hence the PI10 controller may produce larger stepsizes in situations where the integration error and the convergence rate are below their respective set points. In Figure 8 we compare the computational cost of the two controllers. We notice, that in comparison to the PI97 controller, a better performance is obtained in the range from 2 to 5 SD's for the PI10 controller. Since high accuracy is not required in reservoir simulation, we suggest the application of the stepsize controller presented in Algorithm 1 for implicit integration of dynamical systems like the two-phase flow problem.

Conclusion

We have established a predictive stepsize control applied to high order methods for temporal discretization of the two-phase flow problem. The family of Runge-Kutta methods has been presented and the implicit ESDIRK scheme with an embedded method for error estimation has been emphasized. A stepsize selection based on error estimates and convergence control of a modified Newton-Raphson method has been presented. The performance of the timestep control for the solution of the two-phase flow problem has been improved. The convergence controller is allowed more influnce on the stepsize selection process, thus improving the relation between the convergence and the error control. This leads to fewer iterations in the eqaution solver. Moreover, the controller produces a smoother stepsize sequence because of the extended use of information of previous stepsizes and the damping of the stepsize suggested by the convergence control.

References

Brouwer, D.R. and Jansen, J.D. [2004] Dynamic optimization of waterflooding with smart wells using optimal control theory spe-78278-pa. *The 2002 SPE European Petroleum Conference.*

Chen, Z. [2007] *Reservoir Simulation : Mathematical Techniques in Oil Recovery*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, ISBN 978-0-898716-40-5.

Gustafsson, K. [1992] *Control of Error and Convergence in ODE Solvers*. Ph.D. thesis, Department of Automatic Control, Lund University, Sweden.

Gustafsson, K. and Söderlind, G. [1997] Control strategies for the iterative solution of nonlinear equations in ode solvers. *SIAM J. Sci. Comput.*, **18**(1), 23–40, ISSN 1064-8275.

Hairer, E. and Wanner, G. [1996] *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer, 2nd edn., ISBN 3540604529.

Houbak, N., Nörsett, S. and Thomsen, P. [1985] Displacement or residual test in the application of implicit methods for stiff problems. *IMA Journal of Numerical Analysis*, **5**(3), 297–305, ISSN 02724979.

Jørgensen, J.B., Kristensen, M.R. and Thomsen, P.G. [2008] A family of esdirk integration methods. *SIAM Journal on Scientific Computing.*

Kennedy, C.A. and Carpenter, M.H. [2003] Additive runge-kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, **44**(1-2), 139 – 181, ISSN 0168-9274.

Kværnø, A. [2004] Singly diagonally implicit runge-kutta methods with an explicit first stage. *BIT Numerical Mathematics*, **44**, 489 – 502, ISSN 0168-9274.

Prothero, A. and Robinson, A. [1974] On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of Computation*, **28**(125), 145–162, ISSN 00255718.

Söderlind, G. [1986] *Folklore and fudge factors in the implementation of fixed-point and Newton iterations for nonlinear ODEs*. Talk presented at ”Workshop on the Numerical Solution of Stiff ODEs”, Sept. 15-17, NTH, Trondheim, Norway.

Völcker, C., Jørgensen, J.B., Thomsen, P.G. and Stenby, E.H. [2009] Simulation of subsurface two-phase flow in an oil reservoir. *The European Control Conference 2009.*

APPENDIX G

Paper D

NMPC for Oil Reservoir Production Optimization

Authors:
Carsten Völcker, John Bagterp Jørgensen, Per Grove Thomsen and
Erling Halfdan Stenby

Published in:
Proceedings of *21st European Symposium on Computer-Aided Process Engineering (ESCAPE21)*
Chalkidiki, Greece
29 May - 1 June 2011

21st European Symposium on Computer Aided Process Engineering – ESCAPE 21
E.N. Pistikopoulos, M.C. Georgiadis and A.C. Kokossis (Editors)

NMPC for Oil Reservoir Production Optimization

Carsten Völcker[a], John Bagterp Jørgensen[a], Per Grove Thomsen[a], Erling Halfdan Stenby[b]
[a]*Department of Informatics and Mathematical Modeling, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark (email: {cv,jbj,pgt}@imm.dtu.dk)*
[b]*Department of Chemistry, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark (email: ehst@kemi.dtu.dk)*

Abstract

In this paper, we use nonlinear model predictive control (NMPC) to maximize secondary oil recovery from an oil reservoir by controlling two-phase subsurface porous flow using adjustable down-hole control valves. The resulting optimal control problem is nonlinear and large-scale. We solve this problem numerically using a single shooting sequential quadratic programming (SQP) based optimization method. Explicit singly diagonally implicit Runge-Kutta (ESDIRK) methods are used for integration of the stiff system of differential equations describing the two-phase flow, and the adjoint method is used for sensitivity computations. We report computational experiences and oil recovery improvements for a standard test case.

Keywords: Reservoir simulation, water flooding, NMPC, single shooting, adjoints, SQP, ESDIRK, adaptive time stepping.

1. Introduction

As the discoveries of new significant oil fields decrease, efficient exploration of existing oil fields using automatic optimization and control of the recovery process is becoming increasingly important (Smith and Maitland (1998) and Jansen et al. (2008)). After drilling the wells the oil is usually produced under the natural pressure in the reservoir, but after some years of production the pressure in the reservoir has dropped below the hydrostatic pressure and this primary recovery phase ends. In the secondary recovery phase, water or gas is injected into the reservoir to maintain the pressure and to displace the oil from the injection wells towards the production wells, see Fig. 1. Even such techniques leaves most of the oil in the microscopic pores of the reservoir rock, and often the recovery factor stays below 50%. Sometimes a further increase in the recovery factor is possible during a tertiary recovery phase where enhanced oil recovery methods such as chemical flooding, steam flooding or in situ combustion are deployed. However, these techniques are relatively expensive and not economically feasible even at the current high level oil prices. Alternatively, the recovery factor in the secondary phase could be increased by real-time model based reservoir management, also known as closed-loop optimization. In the closed-loop approach the water flooding process is optimized by adjustment of smart wells containing down-hole measurement and control equipment. The measurements may be used for frequent updating of the reservoir model, whereas an optimal control strategy can be computed based on the regularly updated model. We focus on the optimal control of injection rates and bottom hole pressures (BHP) of injection and production wells respectively. The objective is to maximize net present value (NPV) of the water flooding process.

2. Control Strategy

The two-phase flow model is derived on the assumption of mass conservation. It is desirable to preserve such properties upon numerical integration in time. As suggested by the authors (Völcker et al. (2009)) a general formulation of such a system of differential equations may be

$$\frac{d}{dt} g(x(t)) = f(x(t), u(t));\ x(t_0) = x_0 \tag{2}$$

in which $g(x(t))$ are the properties conserved, $x(t)$ are the system states, $u(t)$ are the control variables, while the right-hand side $f(x(t),u(t))$ has the usual interpretation. Using (2) we formulate the water flooding problem as a continuous time Bolza problem

$$\begin{aligned} \min_{[x(t),u(t)]_{t_0}^{t_f}} \quad & \phi = \int_{t_0}^{t_f} J(x(t), u(t))dt \\ \text{s.t.} \quad & x(t_0) = x_0 \\ & \frac{d}{dt} g(x(t)) = f(x(t), u(t)) \\ & c(u(t)) \geq 0 \end{aligned} \tag{3}$$

where the dynamics of (2) is represented as the system constraints. We use zero order hold parameterization of the control variables, i.e. we assume that $u(t)$ is a piecewise constant function of time. Besides the system constraints we also consider the input constraints $c(u(t))$ both in terms of upper and lower bounds and in terms of limiting the rate of change of the control variables. For water flooding using multiple injectors and producers, the injection rates and the BHP's of injection wells and production wells respectively can be used to optimimize the flooding process. Reservoir models are large-scale by nature, so the number of system constraints will be in the order of magnitude of $10^5 - 10^6$. We solve (3) using the single shooting method, in which two steps, a system simulation and an update of the control variables, are performed sequentially in each optimization iteration. Because the control variables are fixed during the simulation step, we can regard the system states as functions of the control variables, such that they are uniquely identified by (2). In addition the initial state $x(t_0) = x_0$ is never changed, consequently, the system constraints can be satisfied by the solution of (2) exclusively. In this way the variable space of (3) is strongly reduced.

We consider the ESDIRK12 scheme for temporal discretization of (2). We control the error of the numerical solution by an adaptive step size selection as described by the authors (Völcker et al. (2010a) and Völcker et al. (2010b)). The ESDIRK12 scheme is using an advancing method of first order and is equipped with an embedded Runge-Kutta method of second order, used for error estimation. In order to determine if the optimal solution (the optimal set of controls) is found, we need the sensitivities of the objective function of (3) with respect to the control variables. In each simulation step we compute the system states of the temporal domain, thus we can use this information to compute the gradient by the adjoint method (Jørgensen (2007)). So in each optimization iteration we do one system simulation, then we compute the sensitivies by the adjoint method and finally we update the control variables. Since the state constraints are eliminated from (3) by the variable reduction, we can find the optimal solution of (3) using an SQP based optimization method. The control updates are done by solving the underlying QP, in which we use modified BFGS approximations of the Hessian matrix (Nocedal and Wright (2006)).

3. Numerical Experiment

We consider immiscible isothermal two-phase flow of oil and water in a porous media. A further description is found in e.g. Aziz and Settari (1971) and Völcker et al. (2009). The optimal control strategy is conducted on a synthetic horizontal 2D permeability field (Brouwer and Jansen (2004)). The field contains two narrow high permeable streaks in which the reservoir fluids flows very easily. The injected water has a tendency to flow through the streaks. Consequently, the producer segments positioned at the streaks produce large amounts of water while the field is developed. The fluid properties and the economical data, that we use in this numerical experiment, are found in Kraaijevanger et al. (2007) and Jansen et al. (2009). Except from the crude oil price, which is the current market value. The discount rate related to the NPV is set to zero. Implying that maximizing NPV essentially amounts to maximizing cummulative oil production and minimizing cummulative water production/injection (Sarma et al. (2008)). We constrain the injection rates of injectors and the BHP's of producers. The injection of water is limited to a total of 2 pore volumes (PV’s) injected over the complete period of production, which is one year.
We have applied two different development strategies. In the first approach in Fig. 4 we use fixed injection rates and BHP's (w/o control), such that 2 PV's are injected over a period of 360 days. In the second approach in Fig. 5 and 6 we apply optimized well rates and pressures (w control) leading to only 0.93 PV injected over the full period of 360 days. The control variables are updated every 30 days, and in each subinterval between the updates we simulate the dynamic system using implicit integration with predictive step size adjustment. In Fig. 2 we find an increase in NPV of approximately 10%, provided that the approach using fixed controls is stopped after 220 days of production, while we keep producing with the optimized strategy for all 360 days. In Fig. 3 we find, that the recovery factor (produced oil related to the total initial mass of oil in the reservoir) is 61% and the water cut (produced oil related to the total mass of produced reservoir fluids) is 70% after 220 days of production using the fixed control strategy. Whereas the application of optimized well rates and pressures results in a recovery factor of 60% and a water cut of 60% for all 360 days of production. So the employment of optimal control of the water flooding process leads to a decrease in recovery by 1%, yet still the NPV has increased. The increase is a consequence of the lower production cost that can be explained by the decrease of 10% of the water cut, and the reduction from 1,22 PV's injected using the fixed control strategy to 0,93 PV's injected using the optimized strategy.
Considering the updates of the control variables, as seen in Fig. 7 and 8, we notice the influence of the cost of water injection and the cost of water seperation. The injection rate is reduced as the reservoir fluids are produced and the injection of water has almost stopped when the reservoir is near depletion. This is emphasized by the close to zero gradient of the blue curve in Fig. 2 at 360 days. The BHP's of the producers are regulated in such a way that production of those wells that do not contribute to an increase in NPV is lowered or even completely shut down. Consequently, the injected water is redirected and thereby pushing the remaining oil saturated reservoir fluids towards active producer segments that still contributes positively. This behaviour is

seen in Fig. 5. At this stage the field still represents a significant economical value compared to Fig. 4, where no further increase in NPV is possible according to Fig. 2.

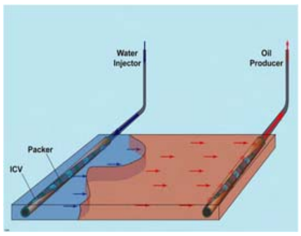


Fig. 1. Schematic view of horizontal wells in the water flooding problem.

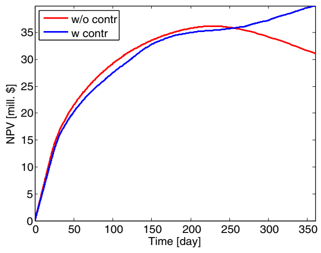


Fig. 2. Net present value.

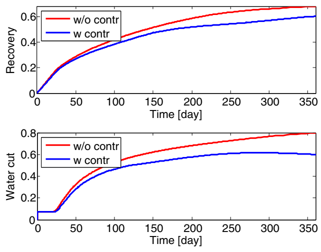


Fig. 3. Recovery factor and water cut.

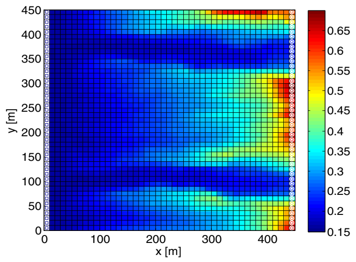


Fig. 4. Oil saturation, 220 days (w/o control).

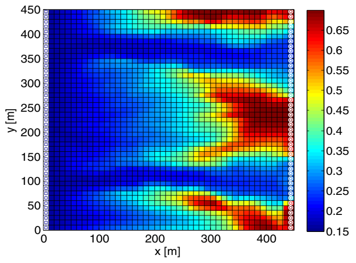


Fig. 5. Oil saturation, 220 days (w control).

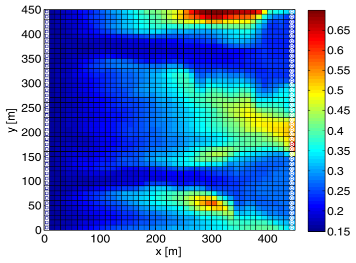


Fig. 6. Oil saturation, 360 days (w control).

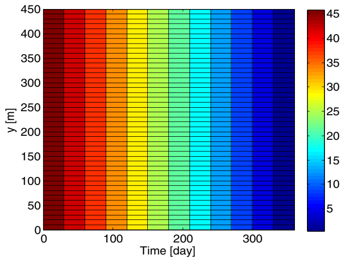


Fig. 7. Individual injection rates (m^3/day) of the 45 injector segments.

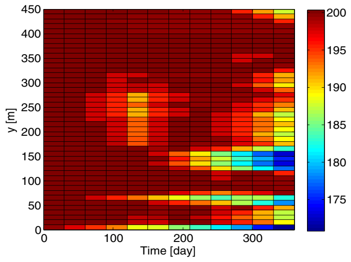


Fig. 8. Individual BHP's (atm) of the 45 producer segments.

4. Conclusion

In this paper, we applied a single shooting SQP based optimization method for production optimization of an oil reservoir. We use a fully implicit integration scheme equipped with an adpative step size control for efficient integration of the model equations, and we compute the sensitivities by the adjoint method. We demonstrate the method on a simple water flooding problem using linear input constraints and low frequency updates of the control variables. The numerical results show an increase of the economical value. A further increase of the economical value of existing oil fields demonstrates the potential of model based optimization as a good alternative to finding new ones.

5. References

Aziz, K. and Settari, A. (1971). *Petroleum Reservoir Simulation*. Applied Science Publishers Ltd, London, first edition.

Brouwer, D.R. and Jansen, J.-D. (2004). Dynamic optimization of waterflooding with smart wells using optimal control theory. *SPE Journal*, 9(4), 391-402.

Jansen, J.-D., Bosgra, O.H. and Van den Hof, P.M. (2008). Model-Based Control of Multiphase Flow in Subsurface Oil Reservoirs. *Journal of Process Control*, **18**, 2008, 846-855.

Jansen, J.-D., Brouwer, R. and Douma, S.G. (2009). Closed Loop Reservoir Management. *SPE Reservoir Simulation Symposium*, Society of Petroleum Engineers, The Woodlands, Texas, 2009.

Jørgensen, J.B. (2007). Adjoint Sensitivity Results for Predictive Control, State- and Parameter-Estimation with Nonlinear Models. *Proceedings of the European Control Conference 2007*, Kos, Greece, July 2-5, 2007, 3649-3656.

Kraaijevanger, J.F.B.M., Egberts, P.J., Valstar, J.R. and Buurman, H.W. (2007). Optimal Waterflood Design Using the Adjoint Method. *SPE Reservoir Simulation Symposium*, Society of Petroleum Engineers, Houston, Texas, USA, 2007.

Nocedal, J. and Wright, S.J. (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering, New York, USA, 2006, second edition.

Sarma, P., Chen, W.H., Durlofsky, L.J. and Aziz, K. (2008). Production Optimization with Adjoint Models under Nonlinear Control-State Path Inequality Constraints. *SPE Evaluation & Engineering*, **11(2)**, 2008, 326-339.

Smith, R.G. and Maitland, G. (1998). The Road Ahead to Real-Time Oil and Gas Reservoir Management. *Trans IchemE*, **76**, 1998, 539-552.

Völcker C., Jørgensen J.B., Thomsen, P.G. and Stenby E.H. (2009). Simulation of subsurface two-phase flow in an oil reservoir. *Proceedings of the European Control Conference 2009*, Budapest, Hungary, August 23-26, 2009, 1221-1226.

Völcker C., Jørgensen J.B., Thomsen, P.G. and Stenby E.H. (2010a). Adaptive Stepsize Control in Implicit Runge-Kutta Methods for Reservoir Simulation. *Proceedings of the 9th International Symposium on Dynamics and Control of Process Systems*, Leuven, Belgium, July 5-7, 2010.

Völcker C., Jørgensen J.B., Thomsen, P.G. and Stenby E.H. (2010b). Explicit Singly Diagonally implicit Runge-Kutta Methods and Adaptive Stepsize Control for Reservoir Simulation. *Proceedings of the 12th European Conference on the Mathematics of Oil Recovery*, Oxford, England, September 6-9, 2010.

APPENDIX H

Paper E

Oil Reservoir Production Optimization
using Optimal Control

Authors:
Carsten Völcker, John Bagterp Jørgensen and Erling Halfdan Stenby

Published in:
Proceedings of *50th IEEE Conference on Decision and Control and European Control Conference (CDC - ECC 2011)*
Orlando, Florida
12-15 December 2011

Oil Reservoir Production Optimization using Optimal Control

Carsten Völcker, John Bagterp Jørgensen and Erling Halfdan Stenby

***Abstract*— Practical oil reservoir management involves solution of large-scale constrained optimal control problems. In this paper we present a numerical method for solution of large-scale constrained optimal control problems. The method is a single-shooting method that computes the gradients using the adjoint method. We use an Explicit Singly Diagonally Implicit Runge-Kutta (ESDIRK) method for the integration and a quasi-Newton Sequential Quadratic Programming (SQP) algorithm for the constrained optimization. We use this algorithm in a numerical case study to optimize the production of oil from an oil reservoir using water flooding and smart well technology. Compared to the uncontrolled case, the optimal operation increases the Net Present Value of the oil field by 10%.**

I. Introduction

Petroleum reservoirs are subsurface formations of porous rocks with hydrocarbons trapped in the pores. Initially, the reservoir pressure may be sufficiently large to push the fluids to the production facilities. However, as the fluids are produced the pressure declines and production reduces over time. When the natural pressure becomes insufficient, the pressure must be maintained artificially by injection of water. Conventional technologies for recovery leaves more than 50% of the oil in the reservoir. Wells with adjustable downhole flow control devices coupled with modern control technology offer the potential to increase the oil recovery significantly. [1] introduces optimal control of smart wells. In these applications, downhole sensor equipment and remotely controlled valves are used in combination with large-scale subsurface flow models and gradient based optimization methods in a Nonlinear Model Predictive Control framework to increase the production and economic value of an oil reservoir [2]–[6]. Wether the objective is to maximize recovery or some financial measure like Net Present Value, the increased production is achieved by manipulation of the well rates and bottom-hole pressures of the injection and production wells. The optimal water injection rates and production well bottom-hole pressures are computed by solution of a large-scale constrained optimal control problem.

This research project is financially supported by the Danish Research Council for Technology and Production Sciences. FTP Grant no. 274-06-0284

C. Völcker and J.B. Jørgensen are with the Department of Informatics and Mathematical Modeling, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark `{cv,jbj}@imm.dtu.dk`

E.H. Stenby is with the Department of Chemistry, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark `ehst@kemi.dtu.dk`

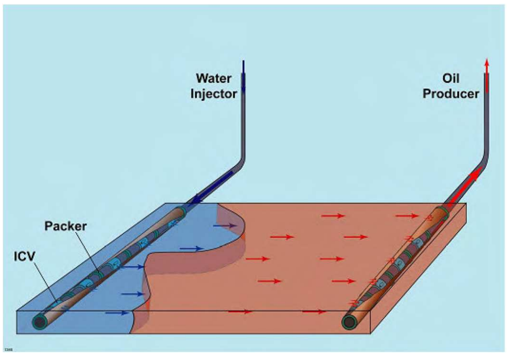


Fig. 1. Schematic view of horizontal wells in the water flooding problem [1].

In this paper, we focus on maximizing the economical value of an oil field and describe the gradient based method to compute the optimal control strategy. An Explicit Singly Diagonally Implicit Runge-Kutta (ESDIRK) method with an adaptive step size control is used for computationally efficient solution of the model [7], [8]. The gradients are computed by the adjoint method [9]. The adjoint equations associated with the integration scheme are solved by integrating backwards in time. The necessary information for the adjoint computation is computed and stored during the forward solution of the model. The backward adjoint computation assembles this information to compute the gradients [9], [10]. We demonstrate the optimal control strategy using a 2-dimensional water-flooding example as illustrated in Fig. 1.

The paper is organized as follows. Section II briefly introduces the two phase flow model. Section III states the general constrained optimal control problem using a novel representation of the system dynamics. The numerical methods for the constrained optimal control problem are described in Section IV. Section V describes specific details related to the objective function and the constraints for the water flooding production optimization problem. Section VI describes a numerical case study illustrating the method. Conlcusions are presented in Section VII.

II. Two-Phase Flow Model

In this section we briefly state the governing equations of an oil reservoir model. We consider isothermal two-phase flow of oil and water in a porous media. We assume complete immiscibility of the reservoir fluids, zero capillary pressure and we neglect gravity effects. Let $P = P(t,r)$ be the pressure in the reservoir and $S = S(t,r)$ be the saturation of water, as function of time $t \geq 0$ and position $r \in \Omega \subset \mathbb{R}^3$ with Ω being the domain of the reservoir. Let $C_w = C_w(P,S)$ and $C_o = C_o(P,S)$ be the mass concentrations of water and oil, respectively. Then the mass balances for water and oil in the reservoir are expressed by

$$\frac{\partial}{\partial t} C_w = -\nabla \cdot F_w + Q_w \tag{1a}$$

$$\frac{\partial}{\partial t} C_o = -\nabla \cdot F_o + Q_o \tag{1b}$$

$F_w = F_w(P,S)$ and $F_o = F_o(P,S)$ are the fluxes of water and oil through the porous media. The source/sink terms of water and oil are denoted $Q_w = Q_w(P,S)$ and $Q_o = Q_o(P,S)$. They describe the flow rate of water from the injection wells into the reservoir and the flow rates of oil and water from the reservoir into the production wells. [11]–[13] provide more detailed descriptions of the model.

III. Continuous-Time Optimal Control

Process models are based on conservation of mass, energy and momentum. It is desirable to preserve such properties upon numerical integration in time. Such problems related to flow in porous media can be represented by the system of differential equations [13]

$$\frac{d}{dt} g(x(t)) = f(x(t), u(t)) \tag{2}$$

with the initial condition $x(t_0) = x_0$. The left-hand side $g(x(t))$ are the properties conserved, $x(t)$ are the system states, $u(t)$ are the manipulated variables, while the right-hand side $f(x(t), u(t))$ has the usual interpretation. Considering (2) we formulate the water flooding problem as a continuous time Bolza problem

$$\min_{[x(t),u(t)]_{t_0}^{t_f}} \quad \int_{t_0}^{t_f} J(t, x(t), u(t)) dt \tag{3a}$$

$$\text{s.t.} \quad \frac{d}{dt} g(x(t)) = f(x(t), u(t)),\ x(t_0) = x_0 \tag{3b}$$

$$u_{min} \leq u(t) \leq u_{max} \tag{3c}$$

$$-u_{min}^{\Delta} \leq \frac{d}{dt} u(t) \leq u_{max}^{\Delta} \tag{3d}$$

The algorithm developed for solution of this problem is suitable for production optimization of oil reservoirs. We use a zero-order-hold parameterization for $u(t)$. This implies that the constraints (3d) should be interpreted as the movement constraints (6d).

IV. Numerical Methods

In this section, we describe a single-shooting algorithm for solution of (3). An ESDIRK method is used for the integration, the gradients are computed using the adjoint method, and the constrained optimization is performed using a quasi-Newton SQP method.

To convert the infinite-dimensional problem (3) into a numerically tractable finite-dimensional problem, we divide the temporal domain $[t_0, t_f]$ into K control steps and each control step into N_k time steps for the integration of the differential equations. We then define a set of control step indices $\mathcal{K}_i = \{i, i+1, \ldots, K-1\}$ and a set of time step indices $\mathcal{N}_k = \{0, 1, \ldots, N_k - 1\}$ for all $k \in \mathcal{K}_0$. The number of control steps is known in advance due to the zero-order-hold parametrization of the manipulated variables. A control step $k \in \mathcal{K}_0$ is defined as an interval between the times $t_{n_k=0}$ and $t_{n_k=N_k}$. Note that $t_{n_0=0} = t_0$ and $t_{n_{K-1}=N_{K-1}} = t_f$. For a given control interval k, the number of time steps are not known in advance as we use an adaptive step length controller in the numerical integrator [7], [8]. This indexing of the control steps and the time steps are illustrated in Fig. 2.

Using the ESDIRK12 scheme for temporal discretization of (2), we can compute the trajectory $\{\{x_{n_k+1}\}_{n_k=0}^{N_k-1}\}_{k=0}^{K-1}$ as the solution of the system of difference equations [13]

$$g(x_{n_k+1}) = g(x_{n_k}) - f(x_{n_k+1}, u_k) h_{n_k} \tag{4}$$

in which $x(t_{n_k}) = x_{n_k}$ and $u(t_{n_k}) = u_k$ for $n_k \in \mathcal{N}_k$ and $k \in \mathcal{K}_0$. For notational convenience we define the residual function

$$\begin{aligned} R_{n_k+1}(x_{n_k+1}, x_{n_k}, u_k) &= \\ g(x_{n_k+1}) - g(x_{n_k}) &- f(x_{n_k+1}, u_k) h_{n_k} = 0 \end{aligned} \tag{5}$$

for $n_k \in \mathcal{N}_k$ and $k \in \mathcal{K}_0$. The continuous-time optimal control problem (3) can be formulated as the following discrete-time optimal control problem

$$\min_{\{\{x_{n_k+1}\}_{n_k=0}^{N_k-1}, u_k\}_{k=0}^{K-1}} \quad \sum_{k=0}^{K-1} \sum_{n_k=0}^{N_k-1} J_{n_k}(x_{n_k}, u_k) \tag{6a}$$

$$\text{s.t.} \quad R_{n_k+1}(x_{n_k+1}, x_{n_k}, u_k) = 0 \tag{6b}$$

$$u_{min} \leq u_k \leq u_{max} \tag{6c}$$

$$-u_{min}^{\Delta} \leq \Delta u_k \leq u_{max}^{\Delta} \tag{6d}$$

where $\Delta u_k = u_k - u_{k-1}$ and

$$J_{n_k}(x_{n_k}, u_k) = \int_{t_{n_k}}^{t_{n_k+1}} J(x(t), u_k) dt,\ n_k \in \mathcal{N}_k,\ k \in \mathcal{K}_0 \tag{7}$$

A. Single-Shooting Optimization

The discrete-time optimal control problem (6) can be solved using single-shooting, multiple-shooting, and the simultaneous method. Reservoir models are large-scale and the number of states are easily in the order of magnitude of $10^5 - 10^6$ for realistic problems.

To keep the dimension of the optimization problem small and to be able to use adaptive temporal step size, we use the single-shooting method in this paper. In the single-shooting method, the manipulated variables, u, are fixed at each iteration and used to solve the difference equations (6b) numerically. Knowledge of the

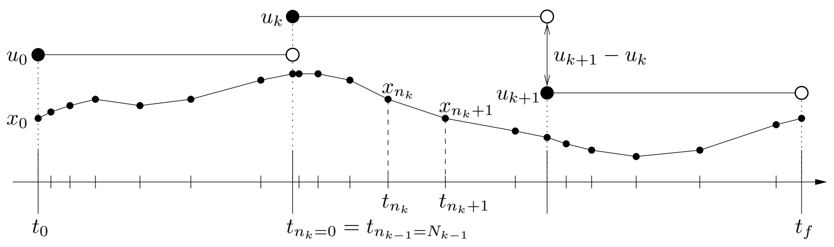


Fig. 2. The zero order hold parametrization and the relation between the control steps and the time steps. For a given time step t_{n_k} in a given control step k the optimal control problem can be described by the system states x_{n_k} and the control settings u_k.

initial state, x_0, the manipulated variables, $\{u_k\}_{k=0}^{K-1}$, and the requirement that the systems dynamics are satisfied determines the states $\{\{x_{n_k+1}\}_{n_k=0}^{N_k-1}\}_{k=0}^{K-1}$. In practical computations, the system constraints (6b) are satisfied by solving (5), i.e. by doing a system simulation. In this way, a single-shooting method for (6) can be stated as the optimization problem

$$\min_{\{u_k\}_{k=0}^{K-1}} \quad \psi(\{u_k\}_{k=0}^{K-1}, x_0) \tag{8a}$$

$$\text{s.t.} \quad u_{min} \le u_k \le u_{max} \tag{8b}$$

$$-u_{min}^{\Delta} \le \Delta u_k \le u_{max}^{\Delta} \tag{8c}$$

with the objective function

$$\psi(\{u_k\}_{k=0}^{K-1}, x_0) = \left\{ \sum_{k=0}^{K-1} \sum_{n_k=0}^{N_k-1} J_{n_k}(x_{n_k}, u_k) : R_{n_k+1}(x_{n_k+1}, x_{n_k}, u_k) = 0, n_k \in \mathcal{N}_k, k \in \mathcal{K}_0 \right\} \tag{9}$$

B. Sequential Quadratic Programming

We solve the reduced problem (8) using sequential quadratic programming (SQP) with line-search and modified BFGS approximations, B, of the Hessian of the Lagrangian [14]. In each iteration, we solve the convex quadratic program

$$\min_{\Delta u} \quad \frac{1}{2}\Delta u^T B \Delta u + \nabla_u \psi^T \Delta u \tag{10a}$$

$$\text{s.t.} \quad \nabla_u c(u)^T \Delta u \ge -c(u) \tag{10b}$$

in which $u = [u_0, u_1, \ldots, u_{K-1}]^T$. The optimal solution of (10), $\Delta u = \{\Delta u_k\}_{k=0}^{K-1}$, combined with a line-search method based on Powell's exact penalty function are used to determine the next iterate

$$u^{(i+1)} = u^{(i)} + \alpha \Delta u^{(i)} \tag{11}$$

α is the line search parameter.

C. Gradient Computation by the Adjoint Method

In computing the search direction, i.e. solving (10), we must compute the gradient $\nabla_{u_k}\psi$. The system states in dynamic optimization problems are dependent on the control variables, in the sense that any past change of the control variables has an influence on all subsequent system states. Consequently, the gradient information of (9) is not directly accessible. The necessary information for computing $\nabla_{u_k}\psi$ is obtained during the simulation step at each optimization iteration in the single-shooting approach. The adjoint method uses this information efficiently to compute the gradients.

Assume that the current iterate, $u^{(i)}$, satisfies the input constraints (8b-8c). The adjoint method can be derived using parts of the first order necessary conditions and the Lagrangian [9]

$$\mathcal{L}(\{\{x_{n_k+1}\}_{n_k=0}^{N_k-1}, u_k, \{\lambda_{n_k+1}\}_{n_k=0}^{N_k-1}\}_{k=0}^{K-1}) = \sum_{k=0}^{K-1} \sum_{n_k=0}^{N_k-1} [J_{n_k}(x_{n_k}, u_k) - \lambda_{n_k+1}^T R_{n_k+1}(x_{n_k+1}, x_{n_k}, u_k)] \tag{12}$$

When the Lagrange multipliers (adjoint variables) $\{\{\lambda_{n_k+1}\}_{n_k=0}^{N_k-1}\}_{k=0}^{K-1}$ and the state variables $\{\{x_{n_k+1}\}_{n_k=0}^{N_k-1}\}_{k=0}^{K-1}$ satisfy certain parts of the KKT conditions, we have

$$\psi(\{u_k\}_{k=0}^{K-1}) = \left\{ \mathcal{L}(\{\{x_{n_k+1}\}_{n_k=0}^{N_k-1}, u_k, \{\lambda_{n_k+1}\}_{n_k=0}^{N_k-1}\}_{k=0}^{K-1}) : R_{n_k+1}(x_{n_k+1}, x_{n_k}, u_k) = 0, n_k \in \mathcal{N}_k, k \in \mathcal{K}_0 \right\} \tag{13}$$

such that we can compute the sensitivity $\nabla_{u_k}\psi$ as the sensitivity $\nabla_{u_k}\mathcal{L}$. The KKT condition corresponding to the state derivative of (12) yields

$$\nabla_{x_{n_k}}\mathcal{L} = \nabla_{x_{n_k}} J_{n_k}(x_{n_k}, u_k) - \nabla_{x_{n_k}} R_{n_k+1}(x_{n_k+1}, x_{n_k}, u_k)\lambda_{n_k+1} - \nabla_{x_{n_k}} R_{n_k}(x_{n_k}, x_{n_k-1}, u_k)\lambda_{n_k} = 0 \tag{14}$$

for $n_k \in \mathcal{N}_k$ and $k \in \mathcal{K}_0$. Substituting the definition of the residuals (5) into (14) and taking derivatives gives

$$\left[\nabla_{x_{n_k}} g(x_{n_k}) - \nabla_{x_{n_k}} f(x_{n_k}, u_k) h_{n_k-1}\right] \lambda_{n_k} = \nabla_{x_{n_k}} J_{n_k}(x_{n_k}, u_k) + \nabla_{x_{n_k}} g(x_{n_k}) \lambda_{n_k+1} \tag{15}$$

from which we can compute the adjoint variables λ_{n_k} marching backwards. The Lagrange multiplier at the final time is $\lambda_{N_{K-1}} = 0$ since the cost-to-go function is zero. $\lambda_{N_{K-1}} = 0$ is used to initialize the backward march for computation of the adjoint variables λ_{n_k}. Special attention must be given when computing $\lambda_{N_{k-1}}$ at the transition between u_{k-1} and u_k for $k \in \mathcal{K}_1$. This is because the first term and the second term on the right-hand side in (14) both belong to control step u_k, while the third term belongs to control step u_{k-1}.

The partial derivatives of (12) with respect to the manipulated variables are

$$\begin{aligned}\nabla_{u_k}\mathcal{L} =& \nabla_{u_k}\mathcal{L} + \nabla_{u_k} J_{n_k}(x_{n_k}, u_k) - \\ & \nabla_{u_k} R_{n_k+1}(x_{n_k+1}, x_k, u_k)\lambda_{n_k+1}\end{aligned} \tag{16}$$

for $k \in \mathcal{K}_0$. Using (5) and $\psi = \mathcal{L}$ we arrive at the following expression for $\nabla_{u_k}\psi$

$$\begin{aligned}\nabla_{u_k}\psi = \nabla_{u_k}\mathcal{L} =& \nabla_{u_k}\mathcal{L} + \nabla_{u_k} J_{n_k}(x_{n_k}, u_k) + \\ & \nabla_{u_k} f(x_{n_k+1}, u_k) h_{n_k}\lambda_{n_k+1}\end{aligned} \tag{17}$$

for $k \in \mathcal{K}_0$. Consequently, the gradients $\nabla_u\psi$ may be computed using (17) in combination with solution of the adjoint equations (15) marching backwards.

V. Water Flooding Production Optimization

The objective of oil reservoir management is to maximize the economic value of the oil reservoir. Essentially, we want to produce as much oil as possible while keeping the operational cost at a minimum. We do this by maximizing the Net Present Value (NPV). Consequently the stage cost $J(t) = J(t, x(t), u(t))$ in (3) becomes

$$\begin{aligned}J(t) = -e^{-dt}\Bigg[&\sum_{j\in\mathcal{N}_{pro}} (r_{op}Q_{o,j}(t) - r_{wp}Q_{w,j}(t)) \\ & - \sum_{j\in\mathcal{N}_{inj}} r_{wi}Q_{w,j}(t)\Bigg]\end{aligned} \tag{18}$$

The factor e^{-dt} accounts for the time value of capital. The terms contributing to $J(t)$ are the value of the produced oil, the cost of separating water from the produced oil, and the cost of water injection. r_{op} is the oil price, r_{wp} is the water separation cost, and r_{wi} is the water injection cost. $Q_{o,j}(t)$ is the oil production and $Q_{w,j}(t)$ is the water production at production wells, $j \in \mathcal{N}_{pro}$, at time t. $Q_{w,j}(t)$ is water injection rate as the water injectors, $j \in \mathcal{N}_{inj}$. d is the continuous discount rate (cost of capital per unit time).

For water flooding using multiple injectors and producers, the well rates and pressures are adjusted by the optimal control problem (3) such that the NPV is maximized [1], [5].

The inequality constraints in (3) are bound constraints (3c) and rate-of-movement constraints (3d). The bound constraints corresponds to constraints on the water injection rates at the injectors and the bottom hole pressures (BHPs) at the production wells. The lower bounds on the water injection rates are zero, while the upper bound is computed such that no more than PV$_{max}$ pore volumes

TABLE I
Reservoir properties.

Symbol	Description	Value	Unit
ϕ	Porosity	0.2	-
c_r	Rock compressibility	0	Pa^{-1}
ρ_o	Oil density (at 1 atm)	800	kg/m^3
ρ_w	Water density (at 1 atm)	1000	kg/m^3
c_o	Oil compressibility	10^{-5}	Pa^{-1}
c_w	Water compressibility	10^{-5}	Pa^{-1}
μ_o	Oil viscosity (dynamic)	1	cP
μ_w	Water viscosity (dynamic)	1	cP
S_{or}	Residual oil saturation	0.15	-
S_{wc}	Connate water saturation	0.20	-
$k_{ro,wc}$	End-point rel. perm., oil	0.8	-
$k_{rw,or}$	End-point rel. perm., water	0.6	-
n_o	Corey exponent, oil	2.0	-
n_w	Corey exponent, water	1.5	-
P_{init}	Initial reservoir pressure	200	atm
S_{init}	Initial water saturation	0.3	-

of water are injected over the time horizon considered, $[t_0, t_f]$. These bound constraints implies that we will implicitly satisfy

$$0 \leq \sum_{k=0}^{K-1}\sum_{n_k=0}^{N_k-1}\sum_{j=1}^{\mathcal{N}_{inj}} Q^{wi}_{n_k,j} h_{n_k} \leq \mathrm{PV}_{max} \tag{19}$$

The reservoir fluids are trapped inside the pores of a porous medium. The total void space of a reservoir is defined by the fraction (the porosity) of the porous medium that is not occupied by rock. Ideally we would replace and thus produce all the reservoir fluids by injecting one PV of water into the reservoir. The BHP's in the production wells are restricted to be lower than the initial pressure of the reservoir. The lower bound of the BHP's is chosen such that the pressure in the well is high enough to push the produced fluids to the production facilities. The rate-of-change constraints of both the injection rates and BHP's are chosen such that the controller is able to change e.g. the injection rate from maximum to minimum within a predefined number of control steps.

VI. Numerical Case Study

In this section, we apply our algorithm for the constrained optimal control problem (3) to maximize the Net Present Value of a horizontal 2D reservoir using water flooding and smart well technology. The permeability field of the reservoir is illustrated in Fig. 3 [1]. The reservoir dimensions are 450 × 450 × 10 m and it is discretized into 45 × 45 × 1 grid blocks. One horizontal injector (white squares at $x = 5$ m) and one horizontal producer (white circles at $x = 445$ m) are divided into 45 segments each. With this setup each grid block that is penetrated by a well represents a well segment. Table I lists the geological and fluid properties of the reservoir. The economical data are listed in Table II [2] [5]. The discount rate is zero, $d = 0$ [4]. Table III provides the constraints of the injection rates and the BHP's as well as the maximum allowed number of PV's to be injected over the period of production.

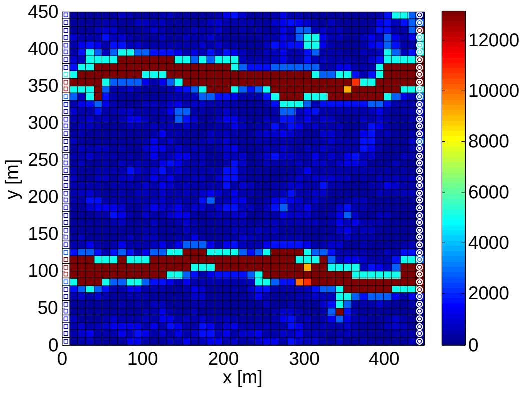


Fig. 3. Permeability field (mDa) with two high permeable streaks, 45 injector segments (white squares at $x = 5$ m) and 45 producer segments (white circles at $x = 445$ m) ([1]).

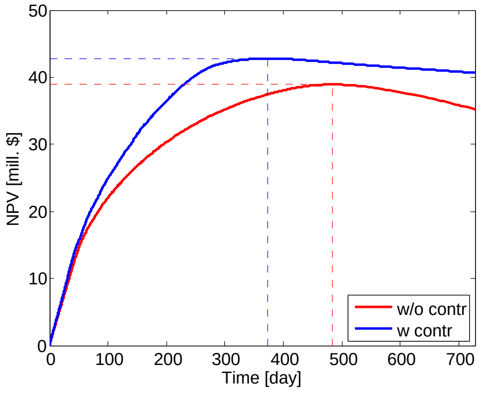


Fig. 5. NPV over 728 days of production. Red curve: Fixed injection rates and BHP's. Blue curve: Optimized injection rates and BHP's.

TABLE II

ECONOMIC DATA.

Symbol	Description	Value	Unit
r_{op}	Oil price	283,0	$\$/m^3$
r_{wp}	Cost of water separation	31,5	$\$/m^3$
r_{wi}	Cost of water injection	5,0	$\$/m^3$
d	Discount rate	0	-

We apply two different production strategies. In the first approach, we use fixed injection rates and fixed BHP's. In this case, 2 PV's are injected over a period of 728 days (2 years). In the second approach, we apply optimized well rates and pressures that we update every 28 days (4 weeks). This strategy leads to 1.00 PV injected over the optimal production period of 374 days. Fig. 4(a) illustrates the injected pore volumes as function of time. Fig. 4(b) illustrates that the recovery factor (produced oil related to the initial mass of oil in the reservoir) and the water cut (produced oil related to the total mass of produced reservoir fluids) as function of time. Fig. 5 shows the NPV as function of the time in which we develop the reservoir. Without control, the optimal development period is 484 days. In the case with optimized water injections and BHPs, the optimal development period is 374 days. NPV increases by approximately 10% by adjusting the water injections and BHPs compared to the uncontrolled case with a development period of 484 days. The recovery factor corresponding to optimal operation in the controlled case is 65%. In the uncontrolled case, the optimal recovery factor is 63%. The corresponding optimal water cuts are 62% in the controlled case and 72% in the uncontrolled case. Thus, the 10% increase in NPV for the controlled case is due to 2% increased oil recovery, a 10% decrease in produced water, and a reduction in injected water from 1.33 PV to 1.00 PV.

TABLE III

CONTROLLER SETTINGS.

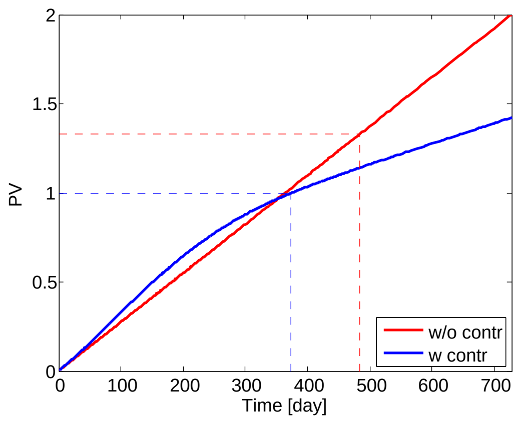
Symbol	Description	Value	Unit
Q^{wi}_{min}	Min. injection rate	0	m^3/day
Q^{wi}_{max}	Max. injection rate	50	m^3/day
BHP_{min}	Min. BHP in producers	150	atm
BHP_{max}	Max. BHP in producers	200	atm
ΔQ^{wi}_{min}	Max. rate of change	-3.85	m^3/day
ΔQ^{wi}_{max}	Max. rate of change	3.85	m^3/day
$\Delta \mathrm{BHP}_{min}$	Max. rate of change	-3.85	atm
$\Delta \mathrm{BHP}_{max}$	Max. rate of change	3.85	atm
PV_{max}	Max. PV's allowed	4	-
T	period of production	728	days

Fig. 7 illustrates the optimal water injection rates and the optimal BHPs for the controlled case. The water injection rates are increased in the injectors located at regions with low permeabilities. Similarly, the water injection rates are decreased for the injectors located in areas with high permeability. The BHPs are adjusted such that the back pressures are increased at locations with high water breakthrough. Fig. 8 illustrates the corresponding oil saturations of the reservoir at time 50, 125, 200, 374 days for the optimally controlled case. Fig. 6 shows the oil saturations for the uncontrolled case after 484 days of production.

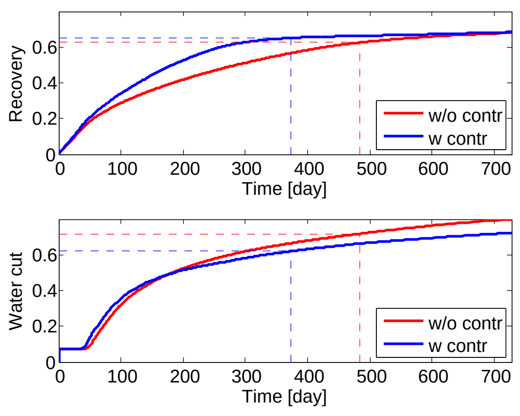
VII. CONCLUSION

We have implemented a numerical method for solution of large-scale constrained optimal control problems (3). The implementation uses a novel formulation of the system dynamics that is relevant to describe flow in porous media. We use Explicit Singly Diagonally Implicit Runge-Kutta (ESDIRK) methods for the integration along with adaptive temporal step sizes. The optimization is based on single-shooting, the SQP optimization algorithm with line-search and BFGS approximations of the Hessian, and the adjoint method for computation of the gradients.

We use this algorithm to maximize the Net Present Value of an oil reservoir case study. In this case study,

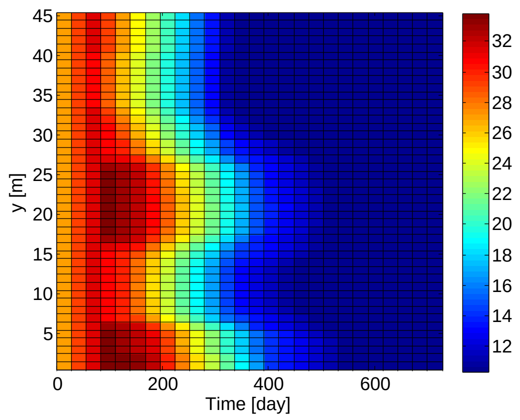


(a) PV injected.

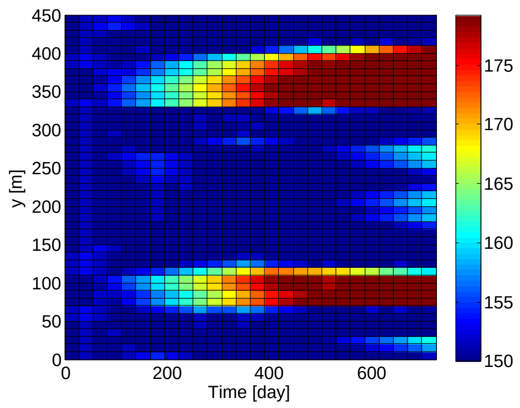


(b) Recovery factor (top) and water cut (bottom).

Fig. 4. Production data over 728 days of production. Red: Fixed injection rates and BHP's. Blue: Optimized injection rates and BHP's.



(a) Individual injection rates (m^3/day) of 45 injectors.



(b) Individual BHP's (atm) of 45 producers.

Fig. 7. Injection rates and BHP's over 728 days of production, updated each 28 days. The injectors and producers are depicted in Figure 3.

we use water flooding to produce the oil. The developed large-scale constrained optimal control algorithm computes the optimal profiles of water injection rates and the bottom hole pressures. Compared to the uncontrolled case, the Net Present Value in the controlled case increases by 10%. This figure demonstrates a significant economic potential of applying smart well technology along with constrained optimal control in oil reservoir management.

References

[1] D. Brouwer and J.-D. Jansen, "Dynamic optimization of waterflooding with smart wells using optimal control theory," *SPE Journal*, vol. 9, no. 4, pp. 391–402, 2004.

[2] J. F. B. M. Kraaijevanger, P. J. Egberts, J. R. Valstar, and H. W. Buurman, "Optimal waterflood design using the adjoint method," in *SPE Reservoir Simulation Symposium*. Houston, Texas, U.S.A.: Society of Petroleum Engineers, 2007.

[3] J.-D. Jansen, O. H. Bosgra, and P. M. Van den Hof, "Model-based control of multiphase flow in subsurface oil reservoirs," *Journal of Process Control*, vol. 18, no. 9, pp. 846–855, Oct. 2008.

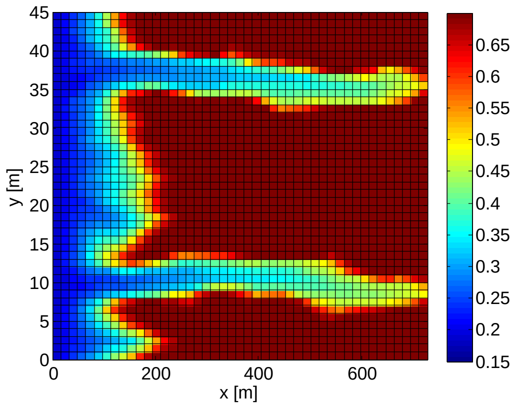
[4] P. Sarma, W. H. Chen, L. J. Durlofsky, and K. Aziz, "Production optimization with adjoint models under nonlinear control-state path inequality constraints," *SPE Reservoir Evaluation & Engineering*, vol. 11, no. 2, pp. 326–339, 2008.

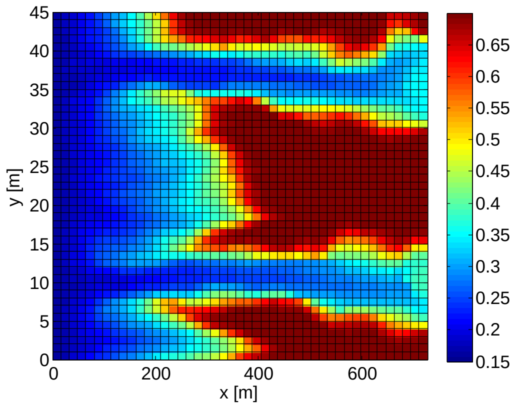
[5] J.-D. Jansen, R. Brouwer, and S. G. Douma, "Closed loop reservoir management," in *SPE Reservoir Simulation Symposium*. The Woodlands, Texas: Society of Petroleum Engineers, 2009.

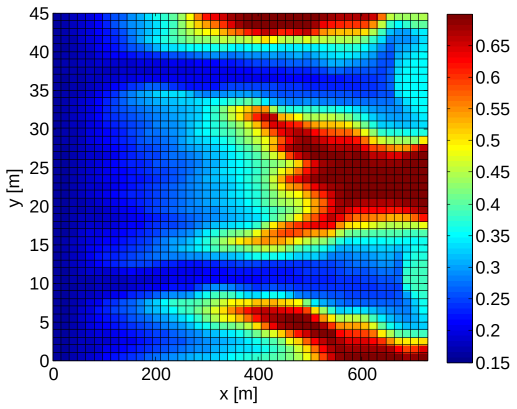
[6] E. Suwartadi, S. Krogstad, and B. A. Foss, "On state constraints of adjoint optimization in oil reservoir waterflooding," in *SPE/EAGE Reservoir Characterization and Simulation Conference*. Abu Dhabi, UAE: Society of Petroleum Engineers, 2009.

[7] C. Völcker, J. B. Jørgensen, P. G. Thomsen, and E. H. Stenby, "Adaptive stepsize control in implicit runge-kutta methods for reservoir simulation," *9th International Symposium on Dynamics and Control of Process Systems*, 2010.

[8] ——, "Explicit singly diagonally implicit runge-kutta methods


(a) 0.14 PV injected after 50 days.


(b) 0.41 PV injected after 125 days.


(c) 0.64 PV injected after 200 days.

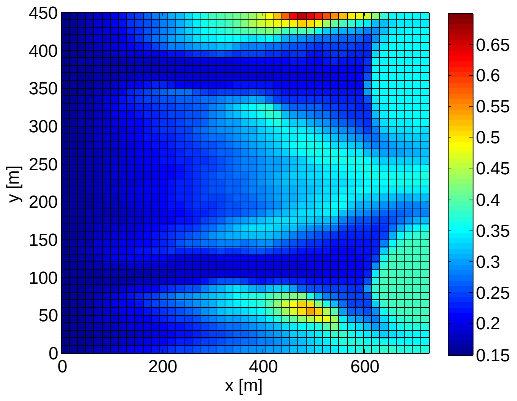

(d) 1.00 PV injected after 374 days.

Fig. 8. Oil saturations after 50, 125, 200 and 374 days of production using optimized injection rates and BHP's.

and adaptive stepsize control for reservoir simulation," *12th European Conference on the Mathematics of Oil Recovery*, 2010.

[9] J. B. Jørgensen, "Adjoint sensitivity results for predictive control, state- and parameter-estimation with nonlinear models," in *Proceedings of the European Control Conference 2007*, Kos, Greece, July 2-5 2007, pp. 3649–3656.

[10] P. Sarma, K. Aziz, and L. Durlofsky, "Implementation of adjoint solution for optimal control of smart wells," in *SPE Reservoir Simulation Symposium*. The Woodlands, Texas: 2005,. Society of Petroleum Engineers Inc., 2005.

[11] K. Aziz and A. Settari, *Petroleum Reservoir Simulation*, 1st ed. London: Applied Science Publishers Ltd, 1979.

[12] Z. Chen, *Reservoir Simulation : Mathematical Techniques in Oil Recovery*. Philadelphia: Society for Industrial and Applied Mathematics (SIAM), 2007.

[13] C. Völcker, J. B. Jørgensen, P. G. Thomsen, and E. H. Stenby, "Simulation of subsurface two-phase flow in an oil reservoir," in *Proceedings of the European Control Conference 2009*, Budapest, Hungary, August 23-26 2009, pp. 1221–1226.

[14] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, USA: Springer Series in Operations Research and Financial Engineering, 2006.

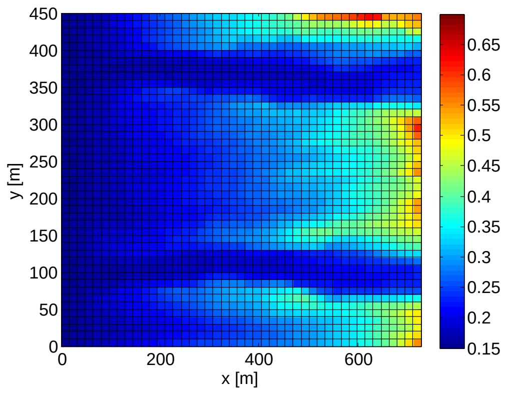


Fig. 6. Oil saturation 484 days of production using fixed injection rates and BHP's.