# EFFICIENT CLOSED-LOOP OPTIMAL CONTROL OF PETROLEUM RESERVOIRS UNDER UNCERTAINTY

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF PETROLEUM ENGINEERING

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Pallav Sarma

September 2006

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

_____

(Dr. Khalid Aziz) Principal Co-Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

_____

(Dr. Louis J. Durlofsky) Principal Co-Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

_____

(Dr. Jef Caers)

Approved for the University Committee on Graduate Studies.

# Abstract

Practical realtime production optimization problems typically involve large, highly complex reservoir models, with thousands of unknowns and many constraints. Further, our understanding of the reservoir is always highly uncertain, and this uncertainty is reflected in the models. As a result, performance prediction and production optimization, which are the ultimate goals of the entire modeling and simulation process, are generally suboptimal. The key ingredients to successful realtime reservoir management would involve efficient optimization and uncertainty propagation algorithms combined with efficient model updating (history matching) algorithms for data assimilation and uncertainty reduction in realtime.

This work discusses a closed-loop approach for efficient realtime production optimization that consists of three key elements – adjoint models for efficient parameter and control gradient calculation, polynomial chaos expansions for efficient uncertainty propagation, and Karhunen-Loeve (K-L) expansions and Bayesian inversion theory for efficient realtime model updating (history matching). The control gradients provided by the adjoint solution are used by a gradient-based optimization algorithm to determine optimal control settings, while the parameter gradients are used for model updating. We also investigate an adjoint construction procedure that makes it relatively easy to create the adjoint and is applicable to any level of implicitness of the forward model. Polynomial chaos expansions provide optimal encapsulation of information contained in the input random fields and output random variables. This approach allows the forward model to be used as a black box but is much faster than standard Monte Carlo techniques. The K-L representation of input random fields allows for the direct application of adjoint techniques for history matching and uncertainty propagation algorithms while assuring that the two-point geostatistics of the reservoir description are maintained.

We further extend the basic closed-loop algorithms discussed above to address two important issues. The first concerns handling non-linear path inequality constraints during optimization. Such constraints always exist in practical production optimization problems, but are quite difficult to maintain with existing optimal control algorithms. We propose an approximate feasible direction algorithm combined with a feasible line-search to satisfy such constraints efficiently. The second issue concerns the Karhunen-Loeve expansion, used for both the uncertainty propagation and model-updating problems. It is computationally very expensive and impractical for large-scale simulation models, and since it only preserves two-point statistics of the input random field, it may not always be suitable for arbitrary non-Gaussian random fields. We use Kernel Principal Component Analysis (PCA) to address these issues efficiently. This approach is much more efficient, preserves high-order statistics of the random field, and is differentiable, meaning that gradient-based methods (and adjoints) can still be utilized with this representation.

The benefits and efficiency of the overall closed-loop approach are demonstrated through realtime optimizations of net present value (NPV) for synthetic and real reservoirs under waterflood subject to production constraints and uncertain reservoir description. The closed-loop procedure is shown to provide a substantial improvement in NPV over the base case, and the results are seen to be very close to those obtained when the reservoir description is known apriori.

# Acknowledgments

When I arrived at Stanford almost five years ago, I had absolutely no intention of pursuing a PhD. All I wanted was to complete my MS, get a nice job, and live happily ever after. I was a “cool dude” (or at least I thought I was) during my undergraduate years. Studies were of secondary importance to me, exams were a waste of time, and the ultimate goal of the four years of slogging was only to land a nice, stable job. Needless to say, this shortsighted attitude of mine towards life has been turned upside down during these five years at Stanford, and this is what I am most grateful for.

First and foremost, I would like to thank Prof. Khalid Aziz, who is not only my co-advisor, but was also my advisor during my MS. As such, I have had the privilege of being mentored by him throughout these years at Stanford. The extent to which he has inspired me is unparalled, and his belief in me is one of the main factors that has driven me to pursue a PhD. He has always given me the necessary independence to pursue my thoughts and ideas while always providing a broad vision on possible avenues for further research, which undoubtedly are key factors towards doing original research.

I had the opportunity of being mentored not by one but two advisors, and I think it will be hard to find anybody other than Prof. Lou Durlofsky who better complements Prof. Aziz in regards to research. Lou has a more “hands on” attitude, and has always been deeply involved with my work, critically scrutinizing my ideas and work, and providing new ideas to test and contemplate. He has been the one to give me the necessary “push” whenever I tended to relax (which I do quite often), and this work and all the papers associated with it would certainly not be possible in his absence. I also thank him deeply for the numerous hours he put in for correcting this thesis and associated papers.

I would also like to thank the other members of my PhD committee, namely Prof. Jef Caers, Prof. Benjamin van Roy and Prof. Jerry Harris, who put in a lot of time and effort to read and comment on this thesis. Further, I would like to thank all the teachers from whom I had the privilege to learn something or the other, be it elementary math or reservoir simulation. I would especially mention Prof. Roland Horne, Prof. Andre Journel, Prof. Albert Tarantola, Prof. Ruben Juanes, Prof. Margot Gerritsen, Prof. David Luenberger, Prof. Michael Saunders, Prof. Robert Lindblom and Prof. Hamdi Tchelepi.

During my stay at Stanford, I had the opportunity of interning with the research teams of Schumberger, ExxonMobil and Chevron. I would like to thank Fikri Kuchuck, Garf Bowen, Bret Beckner and Wen Chen who made these internships possible, and gave me the privilege of working with some extremely smart people. Wen is similar to Khalid in many aspects, especially in his attitude towards research, and I thus feel very happy that I would be working with him in the near future.

I would also like to thank all my friends who have given me comfort and encouragement throughout my stint at Stanford. Further, I would like to thank my parents Bhubaneswar and Rajeswari, my sister Rupjyoti, my cousin brothers Biraj and Jiten, and my dear wife Nidhi, without whose support and encouragement, I would not be at Stanford today. I have to especially mention my mother who has always believed in me, and had the courage to send her only son this far; and my wife, who always supported me with a warm smile, notwithstanding the fact that we had to live in separate cites while I was pursuing this PhD.

Finally, I would like to thank the Department of Petroleum Engineering for providing me with ample financial assistance through fellowships and research assistantships throughout my stay at Stanford. I also thank Saudi Aramco, SUPRI-B, SUPRI-HW and the recently established Smart Fields Consortium and their members for providing financial assistance for this work.

Stanford has instilled in me a spirit to always inquire, and to never take anything for granted. I only hope that I will be able to do justice to this spirit in the long years to come.

Dedicated to my family, without whom this work would not be possible

# Contents

# List of Tables

# List of Figures

# Chapter 1

# 1. Introduction

This work is an attempt to develop new algorithms and improve existing algorithms in order to establish an efficient and accurate framework for closed-loop production optimization of petroleum reservoirs under uncertainty. This chapter motivates the pressing need for maximizing oil recovery and asset value of reservoirs, discusses how closed-loop production optimization can be used as a means towards this end, and finally highlights a set of algorithms that can be utilized to create a framework for efficient realtime closed-loop management of reservoirs and production systems.

## 1.1. The Growing Energy Demand

Energy has played a pivotal role in the prosperity of mankind, and will in all probability continue to do so into the distant future. Worldwide economic growth is expected to be about 3% per year through 2030, a pace similar to the last 20 years [1]. This undiminishing growth and increasing personal income, notably in developing countries, will drive the global demand for energy. It is estimated that the energy demand will increase by 50% by the year 2030 [1], and will be close to 300 million barrels per day of oil equivalent (MBDOE) (see Figure 1-1).

Among the gamut of energy sources available to meet this demand, oil and gas have been the predominant sources satisfying almost 60% of the current energy demand [1]. This percentage is expected to stay relatively stable in the future, at least to 2030, as seen in Figure 1-2 [1], reflecting the advantages of oil and gas in availability, performance, cost, and convenience.

Although the oil and gas resource base is thought to be sufficient to meet the growing energy demand for many decades to come (see Figure 1-3), due to the non-renewable

nature of these resources, it will become increasingly harder to meet this ever-increasing demand for oil and gas. Most of the existing oilfields are already at a mature stage, and the discovery of large new oilfields is becoming a rarity. Figure 1-4 shows the new production that would be required in the future to meet this demand, assuming that no new oilfields are discovered and the current decline rate is sustained [1].

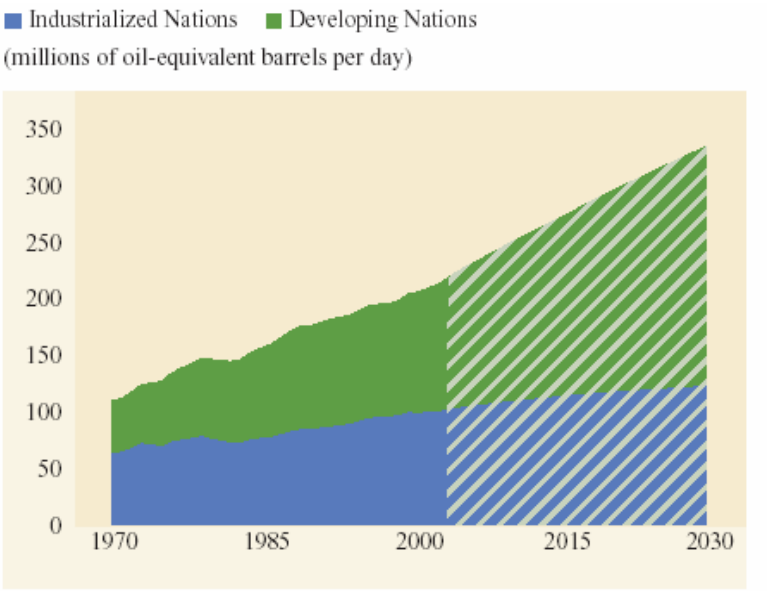


Figure 1-1 World energy demand projected to 2030, from [1].

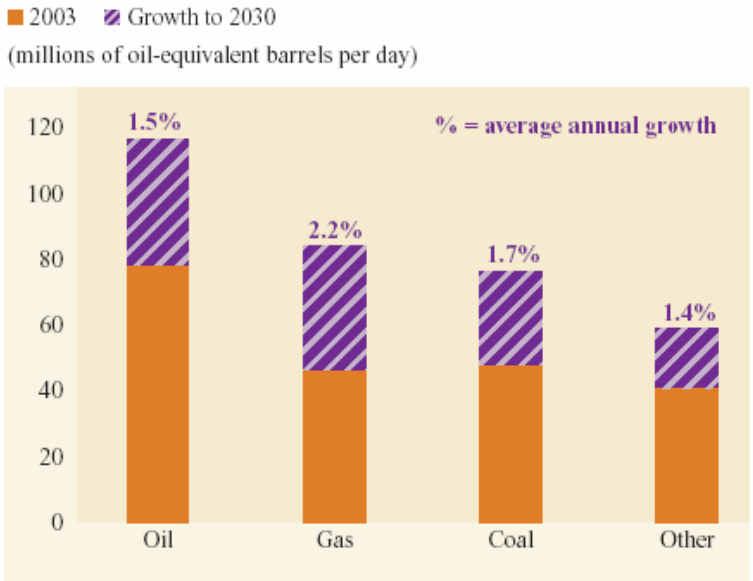


Figure 1-2 Oil and gas will remain the predominant sources of energy, from [1].

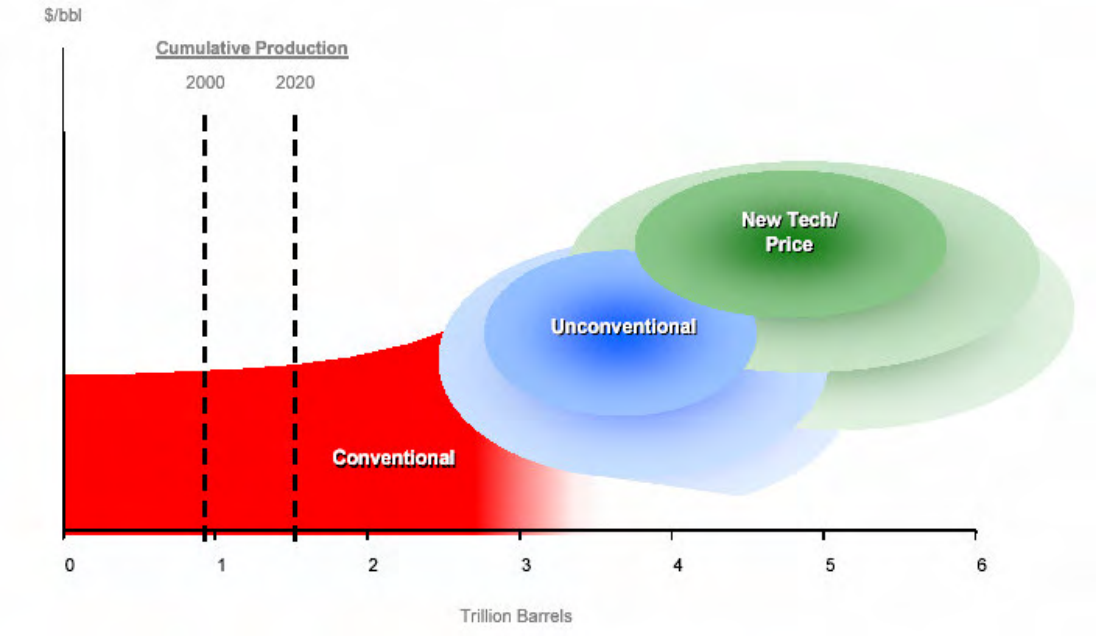


Figure 1-3 Estimated oil and gas reserves compared to production till date, from [2].

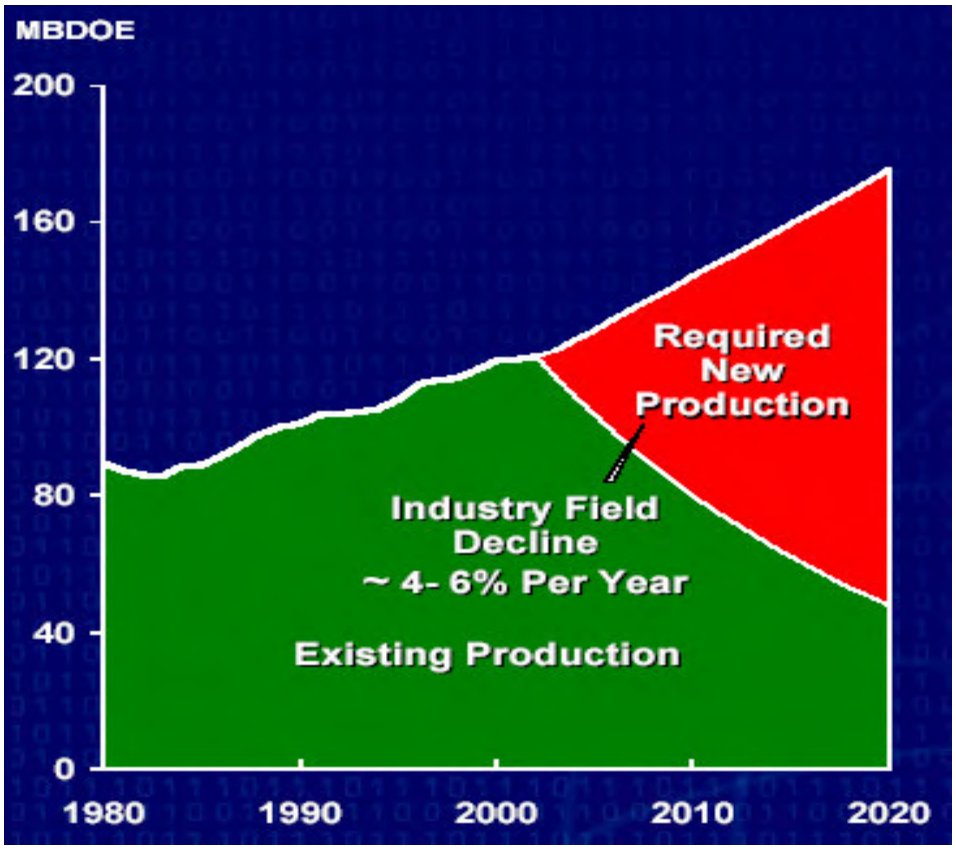


Figure 1-4 Required new production given the current production decline rate, from [1].

In order to meet this gap between demand and supply, it will become increasingly important to maximize recovery from existing reservoirs. The current industry average for the recovery factor is a meager 35%, and that too for reservoirs with favorable production conditions [3]. This number could be as low as 15% for complex reservoirs such as naturally fractured reservoirs [4]. Furthermore, along with maximizing recovery, it will also be essential to minimize capital expenditure and increase asset net present value (NPV) in order to assure that a reservoir achieves its maximum potential. One possible approach to tackle this problem is through a wide array of techniques collectively termed "Production Optimization."

## 1.2. The Production Optimization Process

Production optimization, within the context of this work, refers to long-term maximization of the performance of petroleum reservoirs by making optimal reservoir management and development decisions. The production optimization process is based on a sequence of activities that transform measured or collected data into optimal field management decisions, as seen in Figure 1-5.

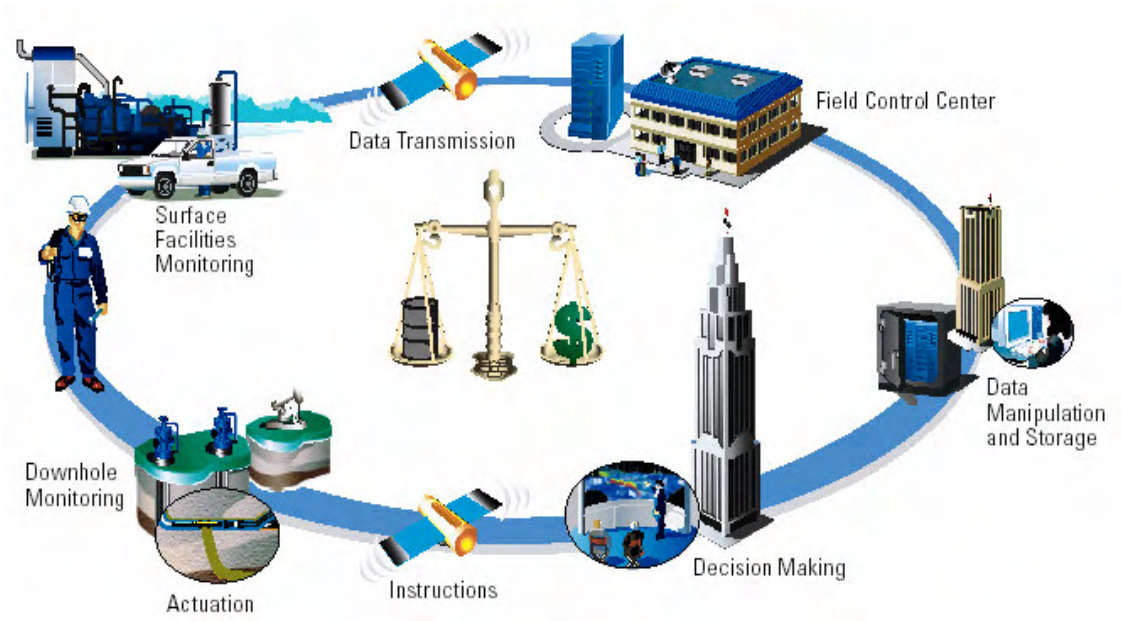


Figure 1-5 The production optimization process, from [3].

Optimization is achieved by comparing measured data with predicted performance and executing a sequence of activities as iterative loops to ensure that the reservoir delivers to its maximum potential. Examples of decisions that constitute the process of production optimization are as follows: (1) where and how many wells should be drilled? (2) what types of wells should be drilled? (3) which reservoir layers should be completed at each well? (4) how should the production/injection schedules be determined for each well?

It is clear from the above that production optimization includes controlling wells in order to maximize (or minimize) some performance criteria. The ability to control wells provides the ability to control fluid flow behavior within the reservoir, thereby enabling maximization or minimization of any criteria by which production performance can be measured. Examples of such criteria could be maximizing oil production (or recovery factor), maximizing net present value, minimizing field watercut, etc. Since wells and their controllability is a key element of the production optimization process, a brief discussion about conventional wells and their evolution towards "smart well" technology is provided below.

### 1.3. Smart Well Technology

A conventional well is a vertical or a slightly deviated well, and has traditionally been the most common type of well drilled (see Figure 1-6). Although conventional wells are relatively inexpensive and easy to implement, a drawback is that their contact area with the reservoir is usually quite small, thus providing a minimum level of reservoir exposure. Further, they do not allow a high degree of controllability, thereby not providing much opportunity for optimization.

Horizontal, highly deviated and multilateral wells are generally referred to as nonconventional or advanced wells (NCWs, see Figure 1-6). A nonconventional well may be as simple as a horizontal well or a vertical/horizontal wellbore with one sidetrack or as complex as a horizontal, extended reach well with multiple laterals.

The drilling of nonconventional wells has become standard practice only during the past decade. A single NCW may be more cost effective than multiple vertical wells in terms of overall drilling and completion costs [6]. In addition, NCWs are well suited for the efficient exploitation of complex reservoirs since they act to increase drainage area and are capable of reaching attic hydrocarbon reserves or reservoir compartments [6]. Consequently, by drilling these wells, capital expenditures and operating costs can be reduced. Compared to conventional wells, these wells provide the same or better reservoir exposure but with fewer wells, hence improving production and injection strategies. However, even a standard NCW does not provide much controllability in realtime.

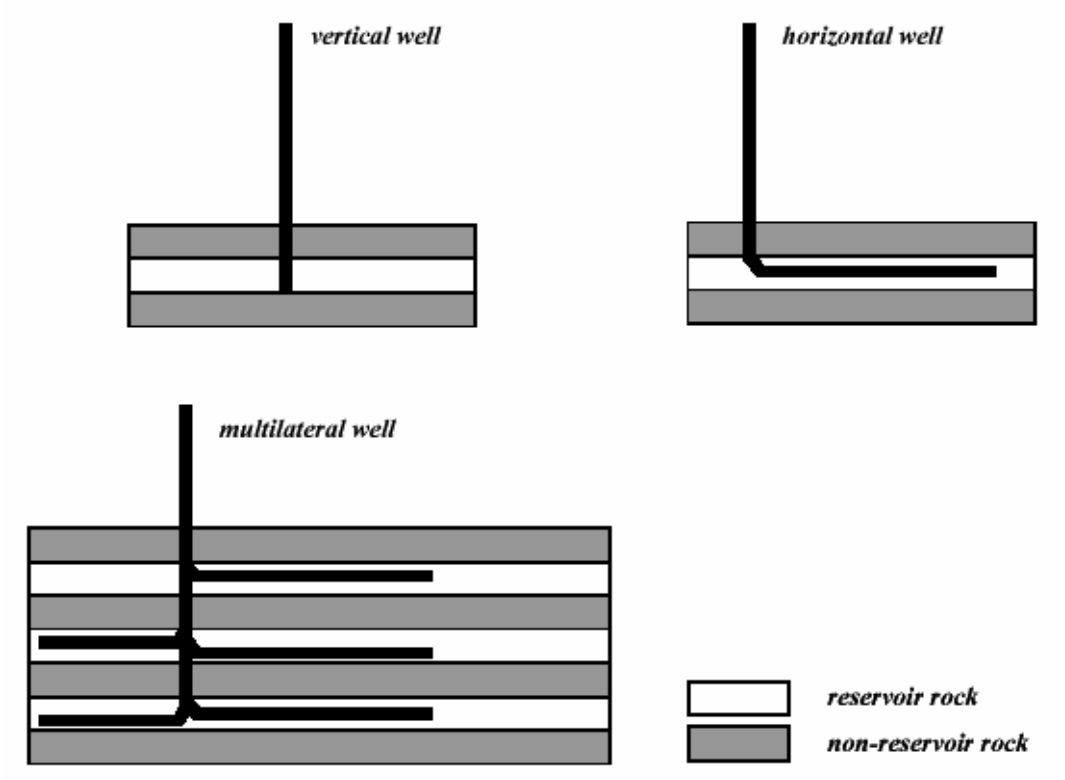


Figure 1-6 Schematic of different types of wells, from [5].

In the last decade, the need to maximize recovery and minimize costs has resulted in the further development of technology to improve measurement and control of production processes through wells. A well equipped with such technology is called a "smart" (or intelligent) well [5,6]. Smart wells essentially have smart completions,

which can be defined as completions with instrumentation (special sensors and valves) installed on the production tubing which allow continuous in-situ monitoring and adjustment of fluid flow rates and pressures (see Figure 1-7). The sensors provide permanent downhole measurements of properties such as temperature, pressure, resistivity, etc., which can lead to a better understanding of the reservoir, thereby enabling more accurate modeling and optimization. Control valves provide the flexibility of controlling each branch or section of a multilateral well independently. In the case of a monobore well (such as a horizontal well), valves transform the wellbore into a multi-branch well, again providing control flexibility for each segmented branch. This controllability has two benefits, first being that it allows control of fluid movements within the reservoir, and second being the ability to react to unforeseen circumstances, thus providing the ability to maximize oil production, recovery factor or any other performance index, in the presence of uncertainty.

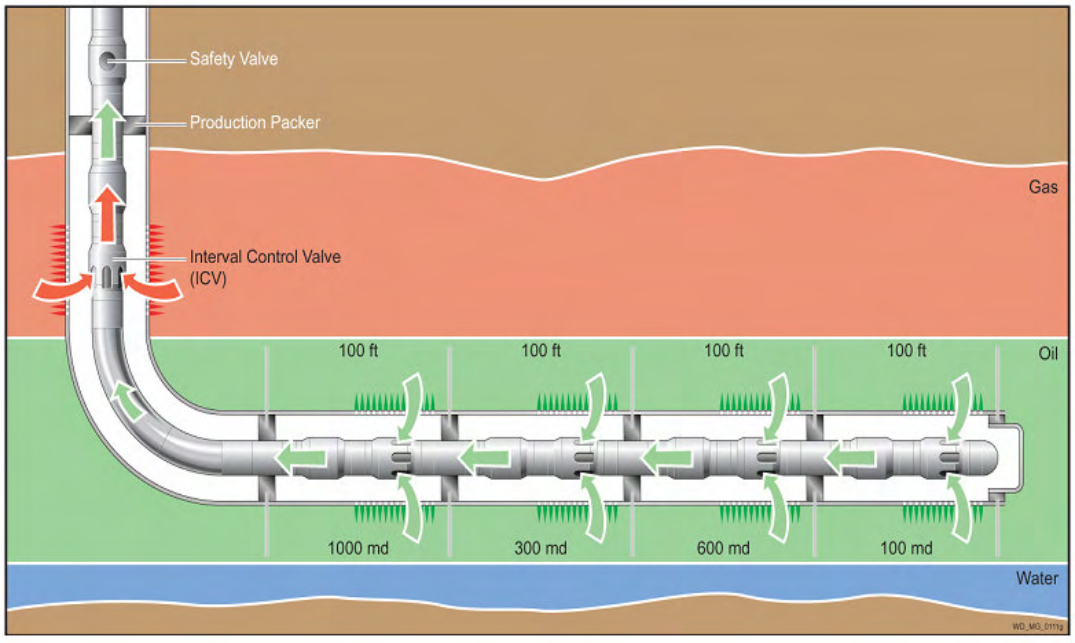


Figure 1-7 Schematic of a smart well, from [7].

Compared to traditional wells, this tremendous increase in monitoring capability and controllability of smart wells truly enables realtime production optimization. The

benefits of these wells have been demonstrated in the industry by various authors, and references can be found in Yeten [6].

### 1.4. Closed-loop Optimal Control

In order that the maximum benefit from the enhanced monitoring capacity and controllability of smart wells be realized, an integrated monitoring and control approach known as model-based closed-loop optimal control may be applied [8]. This realtime model-based reservoir management process can be explained with reference to Figure 1-8. In the figure, the "System" box represents the real system over which some cost function, designated $J(\mathbf{u})$, is to be optimized. In a typical application, $J(\mathbf{u})$ might be net present value or cumulative oil produced. The system consists of the reservoir, wells and surface facilities. Here $\mathbf{u}$ is a set of controls including, for example, well rates and bottom hole pressures (BHP), which can be controlled in order to maximize or minimize $J(\mathbf{u})$. It should be understood that the optimization process results in control of future performance to maximize or minimize $J(\mathbf{u})$, and thus the process of optimization cannot be performed on the real reservoir, but must be carried out on some approximate model. The "Low-order model" box represents the approximate model of the system, which in our case is the simulation model of the reservoir and facilities. This simulation model is a dynamic system that relates the controls $\mathbf{u}$ to the cost function $J(\mathbf{u})$. Since our knowledge of the reservoir is generally uncertain, the simulation model and its output are also uncertain.

The closed-loop process starts with an optimization loop (marked in blue in Figure 1-8) performed over the current simulation model to maximize or minimize the cost function. This optimization must be performed, in general, on an uncertain simulation model. The optimization provides optimal settings of the controls $\mathbf{u}$ for the next control step. These controls are then applied to the real reservoir (as input) over the control step, which impacts the outputs from the reservoir (such as watercuts, BHPs, etc.). These measurements provide new information about the reservoir, and therefore

enable the reservoir model to be updated (and model uncertainty to be reduced). This is called the model updating loop, marked in red in Figure 1-8. The optimization can then be performed on the updated model over the next control step, and the process repeated over the life of the reservoir.

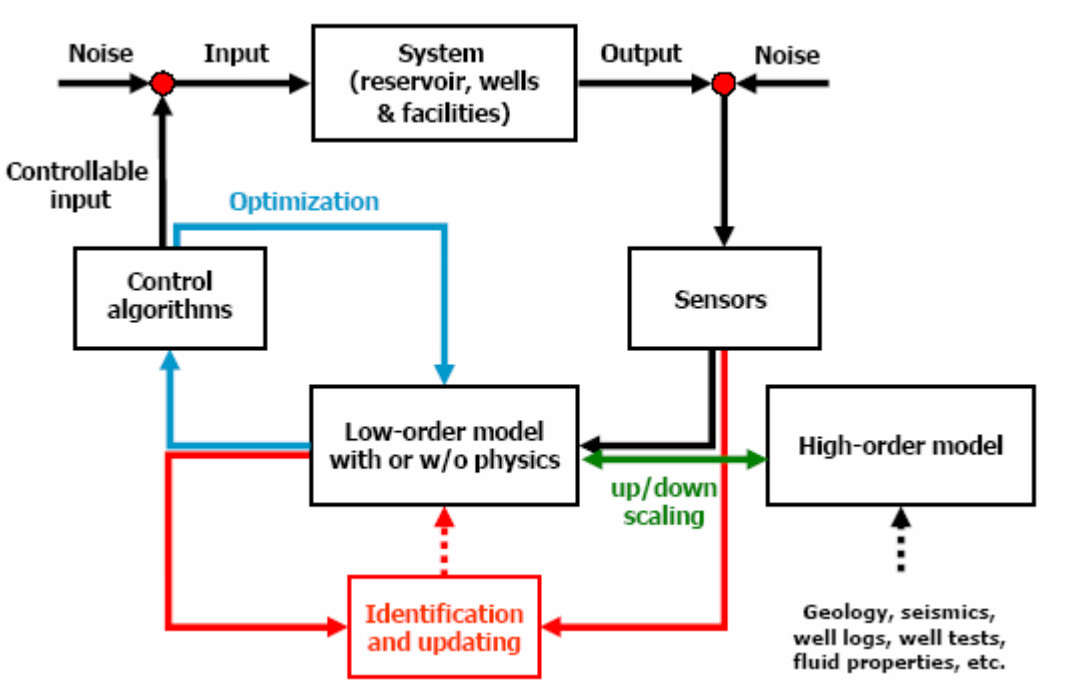


Figure 1-8 Schematic of the Closed-loop Optimal Control approach, from [8].

Many of the key ideas behind closed-loop reservoir management have been known to the oil industry for some time, although different names and forms have been used to describe them [8]. However, most of the earlier work on closed-loop control was geared towards short-term or instantaneous production optimization, and references for such approaches can be found in [9]. Although relatively little information is required to apply these techniques, long-term production performance is not really optimized as the effect of future events is not taken into account during the optimization process. In order to truly maximize production performance, a long-term closed-loop control approach is required, wherein, at each control step, optimization is performed throughout the remaining life of the reservoir in order that the effect of all future events may be taken into account to determine the current optimal controls. It has only been recently that closed-loop long-term production optimization has

generated some interest, and this is the main focus of this work. References to earlier work in long-term closed-loop reservoir management are cited in the following chapters as appropriate.

### 1.5. Research Objectives and Approach

As indicated above, the closed-loop approach for efficient realtime optimization consists of three key components: efficient optimization algorithms, efficient model updating algorithms, and efficient techniques for uncertainty propagation. Efficiency is essential because the closed-loop approach requires running the reservoir simulation model many times, and even a single evaluation of simulation models of real reservoirs can take many hours. The objective of this work is thus to address these key issues by developing new algorithms (and improving existing algorithms) that require a minimal number of evaluations of the simulation model, and integrating them together to realize an efficient closed-loop production optimization framework. This framework should not only be applicable to synthetic "university" reservoir models, but also to real large-scale reservoir models. The remainder of this section provides an outline of the thesis and describes these developments individually. Some references to earlier work are provided here; many others are contained within the following chapters. Note also that most of the material in the following chapters has already been published or submitted for publication (references provided).

In Chapter 2, we develop and apply a gradient-based algorithm for production optimization using optimal control theory [10]. The choice of gradient-based algorithms over other algorithms is due to their efficiency, which as discussed above, is essential for practicality, even though they only provide locally optimal solutions. The approach is to use the underlying simulator as the forward model and its adjoint for the calculation of gradients. An adjoint model is required because practical production optimization problems typically involve large, highly complex reservoir models, thousands of unknowns and many nonlinear constraints, which makes the numerical calculation of gradients impractical. Direct coding of the adjoint model is,

however, complex and time consuming, and the code is dependent on the forward model in the sense that it must be updated whenever the forward model is modified.

We investigate an adjoint procedure that avoids these limitations. For a fully implicit forward model and specific forms of the cost function, all information necessary for the adjoint run is calculated and stored during the forward run itself. The adjoint run then requires only the appropriate assembling of this information (and backward integration) to calculate the gradients. This makes the adjoint code relatively easy to construct and essentially independent of the forward model. This also leads to enhanced efficiency, as no calculations are repeated (generalization to arbitrary levels of implicitness is discussed in the Appendix). The forward model used in this work is the General Purpose Research Simulator (GPRS), a highly flexible compositional/black oil research simulator developed by Cao [11] and others at Stanford University. Through two examples, we demonstrate that the linkage proposed here provides a practical strategy for optimal control within a general purpose reservoir simulator. These examples illustrate production optimization with conventional wells and well configurations representative of smart wells. The efficient treatment of nonlinear constraints is considered in detail in Chapter 5.

In Chapter 3, we discuss a simplified implementation of the closed-loop approach that combines the optimal control algorithm from Chapter 2 with an efficient model updating algorithm for realtime production optimization [12]. Although model updating (automatic history matching) has been a topic of active research for the past few decades, existing algorithms have only had limited success in applications to real reservoirs. This is primarily due to the scarcity of data/measurements and nonlinearity of the forward model, which makes this an inherently ill-posed problem. Therefore, additional sources of information such as prior knowledge in terms of geological constraints have to be utilized in order to generate a reliable set of model parameters and reduce the uncertainty envelope.

Again, gradient-based algorithms are applied due to their efficiency. However, standard gradient-based algorithms have the inherent problem that geological constraints cannot be satisfied, potentially leading to poor predictive capacity of the history matched model. In this work, Bayesian inversion theory [13] is used in combination with an optimal representation of the unknown parameter field in terms of a Karhunen-Loeve expansion [14]. This representation essentially transforms the correlated input random field into a much smaller set of independent random variables, assuring that the two-point geostatistics of the reservoir description are maintained, while also allowing for the direct application of efficient adjoint techniques. The standard Karhunen-Loeve representation is limited in that it cannot capture higher order statistics. This issue is addressed in detail in Chapter 6. Most of the adjoint code used for the optimal control problem can be reused for the updating problem. The benefits and efficiency of the overall closed-loop approach are demonstrated through realtime optimizations of NPV for synthetic reservoirs under waterflood subject to production constraints and uncertain reservoir description. For two example cases, the closed-loop optimization methodology is shown to provide a substantial improvement in NPV over the base case, and the results are seen to be quite close to those obtained when the reservoir description is known a priori.

In the simplified closed-loop discussed above, uncertainty propagation was not considered. Neglecting uncertainty propagation essentially means that the closed-loop process is applied using a single realization of the uncertain parameters, for example, the maximum likelihood estimate. Such a procedure can be expected to provide near-optimal results in many cases, though the treatment of uncertainty will of course be important in many applications. In Chapter 4, efficient uncertainty propagation algorithms are integrated with the closed-loop algorithm to realize a complete closed-loop algorithm for realtime production optimization [16].

Traditionally, Monte-Carlo simulation has been a popular method for uncertainty propagation due to its simplicity and ease of implementation [17]. However, its main

drawback, rendering it infeasible for this application, is that many (e.g., hundreds) forward model evaluations are required for accurate results. On the other hand, techniques such as the perturbation method and stochastic finite elements are much more efficient, but such techniques generally contain underlying assumptions and require access to model equations, implying that a "black box" approach is not possible [17]. In this work, we apply polynomial chaos expansions [17, 18] within the probabilistic collocation method [19] for uncertainty propagation. Polynomial chaos expansions provide optimal encapsulation of information contained in the input random fields and output random variables. The method is similar in efficiency to stochastic finite elements, but allows the forward model to be used as a black box as in Monte-Carlo methods. As a result, implementation is straightforward and the method can be readily integrated with the optimal control and model updating algorithms. Again, the efficiency of the overall closed-loop approach is demonstrated through realtime optimizations of net present value (NPV) for synthetic reservoirs under waterflood.

In the examples discussed in the previous chapters, nonlinear control-state path constraints were not present during optimization. These are constraints that are nonlinear with respect to the controls and have to be satisfied at every time step, for example a maximum water injection rate constraint when BHPs are used as controls. However, practical production optimization problems are usually constrained with such nonlinear control-state path inequality constraints, and it is acknowledged that path constraints involving state variables are particularly difficult to handle [20]. Currently, one category of methods implicitly incorporates the constraints into the forward and adjoint equations to tackle this issue. However, these are either impractical for the production optimization problem, or require complicated modifications to the forward model equations (simulator) [21]. Thus, the usual approach is to formulate the above problem as a constrained nonlinear programming problem (NLP) where the constraints are calculated explicitly after the dynamic system is solved [21]. The most popular of this category of methods (for optimal control

problems) has been the penalty function method [20] and its variants, which are, however, extremely inefficient. All other constrained NLP algorithms require the gradient of each constraint, which is impractical for an optimal control problem with path constraints, as one adjoint has to be solved for each constraint at every iteration of each time step.

In Chapter 5, an approximate feasible direction NLP algorithm based on the objective function gradient and a combined gradient of the active constraints is proposed [21]. This approximate feasible direction is then converted into a true feasible direction by projecting it onto the active constraints by solving the constraints during the forward model evaluation itself. The approach has various advantages. First, only two adjoint evaluations are required at each iteration. Second, all iterates obtained are always feasible, as feasibility is maintained by the forward model itself, implying that any iterate can be considered a useful solution. Third, large step sizes are possible during the line search, which can lead to significant reductions in forward and adjoint model evaluations and large reductions in the objective function. Through two examples, it is demonstrated that the algorithm provides a practical and efficient strategy for production optimization with nonlinear path constraints.

It was shown in the Chapters 3 and 4 that the Karhunen-Loeve (K-L) expansion is required to create a differentiable parameterization of the input random fields (which are of dimension $N_C$, with $N_C$ the number of cells) of the simulation model, in order to perform uncertainty propagation and model updating with adjoints. This requires an eigen decomposition of the covariance matrix (of size $N_C \times N_C$) of the random field, which is usually created numerically from a number of realizations $N_R$ (large enough to capture the essence of the patterns present in the realizations). Eigen decomposition with standard techniques is a very expensive process of order $n^3$ complexity, where $n$ is the dimension of the matrix. Therefore, it is not practical to apply this technique directly to large-scale numerical models. In Chapter 6, the kernel formulation of the eigenvalue problem [22, 23] is applied, implying that instead of performing the eigen

decomposition of the original covariance matrix, an eigen decomposition of another matrix, the kernel matrix of size $N_R \times N_R$, can be performed to determine the eigen decomposition of the original covariance matrix. Since an $N_R$ of order $10^3$ is usually enough to capture the patterns of a typical random field even for an $N_C$ of order $10^{5\text{-}6}$, the kernel matrix decomposition is extremely fast compared to the original matrix decomposition, thereby making the technique feasible even for million or more cell problems.

Another issue with the K-L expansion (also called linear Principal Component Analysis, PCA) is that it only preserves two-point statistics of a random field, and is thus appropriate only for multi-Gaussian random fields. This may not be enough for complex geological scenarios, for example a channelized reservoir, for which the permeability field is far from Gaussian. Thus, in order to perform uncertainty propagation and model updating (with adjoints) with such random fields, an appropriate differentiable parameterization of the non-Gaussian random field is required in terms of a small number of independent random variables. Also in Chapter 6, a nonlinear form of PCA known as kernel PCA (with high order polynomial kernels) [23] is applied to parameterize the non-Gaussian, non-stationary random fields. This allows preserving arbitrary high-order statistics of the random field, is differentiable, meaning that gradient-based methods can be utilized, and is essentially as efficient as linear PCA. Further, a polynomial chaos expansion or a histogram transform can be employed to additionally preserve the marginal distribution of the random field. Results indicate that the proposed method is far superior to just using the K-L expansion for non-Gaussian random fields in terms of reproducing geological features. The kernel PCA representation is then applied to history match a waterflooding problem. This example demonstrates that kernel PCA can be used with gradient-based history matching to provide models that match production history while maintaining multi-point geostatistics consistent with the underlying training image.

The seventh chapter is a culmination of all the algorithms discussed heretofore, and in this chapter, the integration of all these algorithms within an efficient closed-loop optimization framework with application to a realistic reservoir is presented. The closed-loop algorithm is applied on a sector of the simulation model of a Gulf of Mexico reservoir being waterflooded. The objective is to maximize the expected NPV of the sector over a period of eight years by controlling the BHPs of the injectors and producers. The optimization is subject to production constraints (nonlinear path constraints) and an uncertain reservoir description. The closed-loop procedure is shown to provide substantial improvement in NPV over the base case, and the results are very close to those obtained when the reservoir description is known a priori. Through this example, it is verified that the algorithms presented in this thesis indeed provide an efficient and accurate realtime closed-loop optimization framework applicable to real reservoirs.

In Chapter 2, we applied a fully implicit forward model for use with the simplified adjoint construction procedure discussed therein. In the Appendix, it is shown that if the forward model is implemented with the general formulation approach [11], then a similar simplified adjoint construction approach is applicable for any level of implicitness (e.g., IMPES, IMPSAT) of the forward model. Again, all information necessary for the adjoint run is calculated and stored during the forward run itself. Further, the adjoint model is always consistent with the forward model and will have the same level of implicitness as the forward model with this approach.

# Chapter 2

# 2. Deterministic Optimization with Adjoints

As mentioned in the previous chapter, in production optimization problems, the objective is to maximize or minimize some cost function $J(\mathbf{u})$ such as net present value (NPV) of the reservoir, sweep efficiency, cumulative oil production, etc. by manipulating a set of controls $\mathbf{u}$ that include, for example, well rates and bottom hole pressures (BHPs). In this chapter, only a deterministic form of this production optimization problem will be considered; that is, for the purpose of this chapter, the properties of the reservoir simulation model (dynamic system) are assumed to be known deterministically. Methods to incorporate uncertainty will be discussed in subsequent chapters.

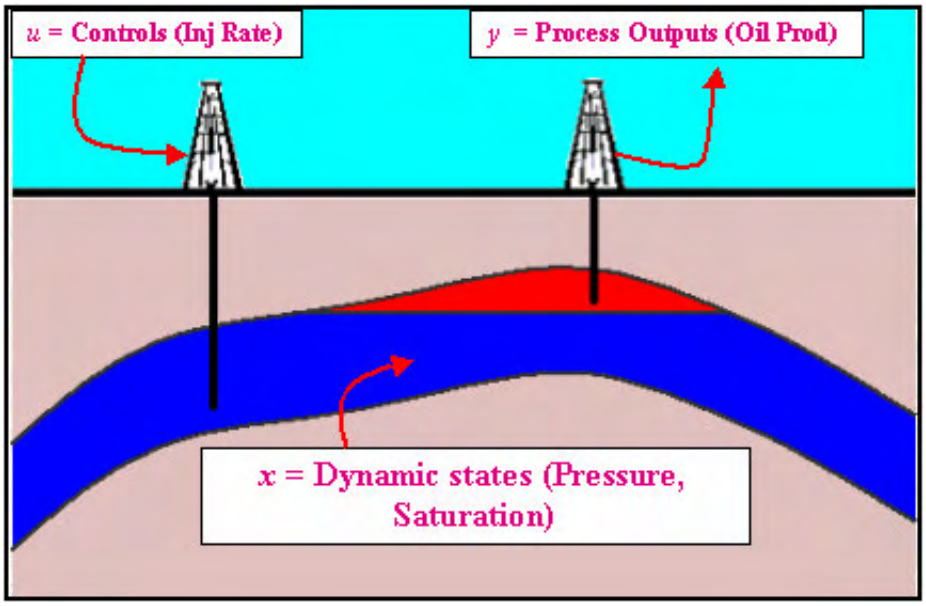


Figure 2-1 Schematic of simple production system

Consider the simple schematic of a reservoir shown in Figure 2-1, where the cost function is cumulative oil production and the control is the injection rate. Changing the

injection rate changes the dynamic states of the system (pressures, saturations), which changes the oil production rate, which in turn impacts the cost function. Thus, the controls $\mathbf{u}$ are related to $J(\mathbf{u})$ through the dynamic system. The dynamic system can also be thought of as a set of constraints that determine the dynamic state given a set of controls. Further, the controls $\mathbf{u}$ themselves may be subject to other constraints that dictate the feasible or admissible values of the controls, such as surface facility constraints or fracture pressure limits. It is these additional constraints that in many cases complicate the problem and the solution process. In this chapter, only linear constraints on $\mathbf{u}$ will be considered. Nonlinear constraints will be discussed in detail in Chapter 5.

The existing optimization algorithms can be broadly classified into two categories: stochastic algorithms like Genetic Algorithms [24] and Simulated Annealing [25], and gradient based algorithms like Steepest Descent [26] and Quasi-Newton algorithms [26]. The first category usually requires many forward model evaluations and does not guarantee monotonic minimization/maximization of the objective function, but is capable (in theory) of finding the global optimum with a sufficiently large number of simulation runs. On the other hand, the second category is generally very efficient, requires few forward model evaluations and also guarantees reduction of the objective function at each iteration, but only assures local optima for non-convex problems [26].

For practical problems, where the simulation grid can be of the order of $10^6$ cells, a single evaluation of the forward model may take many hours; implying that gradient based algorithms would be preferable for such problems. Furthermore, gradient-based algorithms might also be sufficient for the production optimization problem, as any increase in the objective function above the initial manually engineered model is always beneficial. In other words, finding a global optimum may not be necessary – the goal is to determine an operational scenario that represents an improvement over what would otherwise be done.

In order to use gradient-based optimization algorithms, the main requirement is an efficient technique to calculate the gradients of the cost function with respect to the controls. Since the dynamic system is too complicated to calculate the gradients analytically, the simplest approach is to approximate the gradients numerically. This method is very easy to implement, as the forward model is treated as a black box. However, doing so essentially renders the whole process highly inefficient, particularly in the presence of a large number of controls and updates, as one forward model evaluation (simulation) is required for each gradient to be calculated. In particular, considering the simple model mentioned before, in order to calculate the gradient of cumulative oil production with respect to water injection rate at a given time *t*, the injection rate over a time *dt* is perturbed slightly, and the model is evaluated again. The perturbation results in a perturbation of the oil rate (Figure 2-2) and therefore of the cumulative oil production, and the gradient is calculated with the simple forward difference formula as:

$$\frac{\partial J(u)}{\partial u} \approx \frac{J(u+du)-J(u)}{du} \tag{2.1}$$

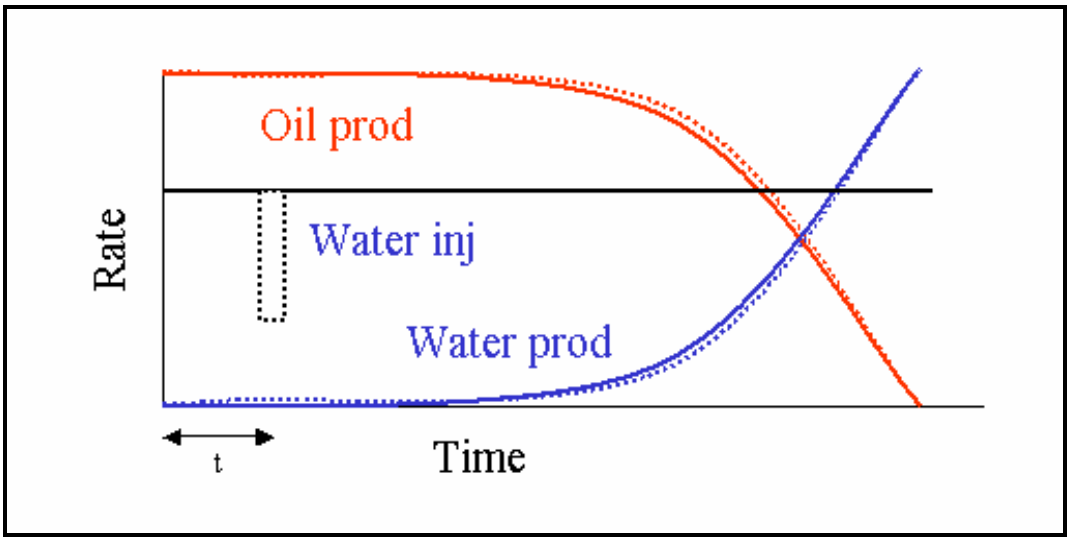


Figure 2-2 Perturbation of injection rate from numerical gradient calculation

Thus, if the set of controls consists of any well variable such as rate or BHP, then the total number of controls would be the product of the total number of wells and the

total number of control updates in time (control steps). As seen in Table 2-1, this number can be very large even for a moderate number of wells and control steps. Another issue with this approach is the selection of the magnitude of each perturbation.

| # Wells or Segments | # Control Updates in Time | Tot. Sim. to get Gradients | Tot. Sim. with OCT |
|---|---|---|---|
| 1 | 1 | 2 | 2 |
| 2 | 2 | 5 | 2 |
| 5 | 5 | 26 | 2 |
| 10 | 10 | 101 | 2 |
| 20 | 10 | 201 | 2 |
| 20 | 20 | 401 | 2 |
| ..... | ..... | ..... | 2 |
| 90 | 200 | 1.8E+04 | 2 |
| 150 | 400 | 6.0E+04 | 2 |

Table 2-1 Number of model evaluations for gradient calculation with numerical approximation and optimal control theory (from Brouwer [5])

This chapter explores the application of adjoint models for efficient production optimization through the efficient calculation of gradients. Of the few existing methods for calculating gradients, adjoint techniques are the most efficient, especially for a large number of controls, as the algorithm is independent of the number of controls. However, the complexity of the adjoint calculations is similar to that of the forward simulation, which is one of the main drawbacks of the algorithm, and is also likely one of the primary reasons why adjoint methods have not gained greater popularity in the petroleum industry. There have been some investigations directed toward the use of (adjoint-based) optimal control for production optimization. Ramirez and coworkers have used it to optimize surfactant flooding [27], carbon dioxide flooding [28] and steam flooding [29]. Zakirov et al. [30] have used adjoint models to optimize production from a thin oil rim. Optimization of waterflooding using adjoints has been studied by many researchers including Asheim [31], Virnovsky [32], Sudaryanto and Yortsos [33], and recently by Brouwer and Jansen [5, 34]. In all of

these investigations, the major emphasis was on the results of the optimization process, rather than on the algorithm itself. Further, in some of the above papers, the forward model (simulator) used was highly simplified [31, 32] or even analytical in nature [33]. Also, almost all of the above studies lack the implementation of nonlinear constraints on the controls themselves, while, in practical production optimization problems, the optimization process is almost always subject to many nonlinear constraints on the controls.

In this chapter, we investigate an adjoint construction procedure that makes it relatively easy to create the adjoint and has the additional advantage of making the adjoint code quite independent of the forward code. Note that the basic idea behind this approach has been known in the petroleum industry for quite some time, for example, Zakirov et al. [30] allude briefly to the idea, although without any detailed discussion of the steps necessary for its implementation. Li et al. [35] also discuss the idea in the context of the history matching problem. In this chapter, we discuss the procedure at a greater level of detail compared to Zakirov et al., with emphasis on its implementation for the production optimization problem. This procedure was implemented within the context of a general purpose research simulator [11]. The current implementation of the algorithm requires the forward model to be fully implicit (the more general case is discussed in the Appendix). Most of the previous investigators have used an IMPES [36] formulation. The performance and practicality of this approach is demonstrated through two examples.

## 2.1. Mathematical Formulation of the Problem

The production optimization problem discussed above requires finding a sequence of control vectors $\mathbf{u}^n$ (of length $m$) for $n = 0,1,...,N-1$, where $n$ is the control step index and $N$ is the total number of control steps, to maximize (or minimize) a performance measure $J\left(\mathbf{u}^0,...,\mathbf{u}^{N-1}\right)$. The optimization can be described very generally with the following mathematical formulation:

$$\max_{\mathbf{u}^n}\left[ J = \phi\left(\mathbf{x}^N\right) + \sum_{n=0}^{N-1} L^n\left(\mathbf{x}^{n+1}, \mathbf{u}^n\right) \right] \forall n \in (0,..,N-1)$$

subject to:

$$\begin{aligned} & g^n(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{u}^n) = 0 \qquad \forall n \in (0,..,N-1) \\ & \mathbf{x}^0 = \mathbf{x}_0 \qquad \text{(Initial Condition)} \\ & \mathbf{A}\mathbf{u}^n \le \mathbf{b} \qquad \forall n \in (0,..,N-1) \\ & \mathbf{LB} \le \mathbf{u}^n \le \mathbf{UB} \qquad \forall n \in (0,..,N-1) \end{aligned} \tag{2.2}$$

Here, $\mathbf{x}^n$ refers to the dynamic states of the system, such as pressures, saturations, compositions etc. The cost function $J$ consists of two terms. The first term $\phi$ is only a function of the dynamic states of the last control step; in an application it could represent, for example, an abandonment cost. The second term, which is a summation over all control steps, consists of the kernel $L^n$ known as the Lagrangian in control literature [37]. For our purposes, it could include the oil and water rates or some function of the saturations (for sweep efficiency). Since $L^n$ usually consists of well parameters or quantities that are functions of well parameters, it is written here in a fully implicit form.

The set of equations $g^n$ together with the initial conditions define the dynamic system, which are basically the reservoir simulation equations for each grid block at each time step:

$$g^n\left(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{u}^n\right) = Accumulation - Flux - Well \tag{2.3}$$

The last two equations of Equation (2.2) refer to the additional constraints for the controls, that is, linear constraints and bounds on controls. These are handled directly by the standard constrained optimization algorithm applied in the following examples. Nonlinear constraints (not shown in Equation (2.2)) are much more difficult to satisfy and a method to honor them will be discussed in Chapter 5. Note that in the above formulation of the problem, the control steps and the actual time steps of the simulator

are considered equivalent, and the derivations below are based on this assumption. This is however, not usually the case, and the number of time steps is generally greater than the number of control steps. The modifications necessary to handle the more general problem when the time steps and control steps are not the same are discussed at the end of the next section.

### 2.2. Gradients with the Adjoint Model

It was stated earlier that the gradients of the cost function with respect to the controls could be calculated very efficiently using the adjoint equations. The adjoint model equations are obtained from the necessary conditions of optimality of the optimization problem defined by Equation (2.2). These necessary conditions of optimality are obtained from the classical theory of calculus of variations. For a relatively simple treatment of this subject, refer to Stengel [37]. A more detailed and rigorous analysis of the problem and generalization to infinite dimensional problems in arbitrary vector spaces is given by Luenberger [38]. The essence of the theory is that the cost function of Equation (2.2) along with all the constraints can be written equivalently in the form of an augmented cost function given by Equation (2.4).

$$J_A = \phi\left[\mathbf{x}^N\right] + \sum_{n=0}^{N-1} L^n\left(\mathbf{x}^{n+1}, \mathbf{u}^n\right) + \boldsymbol{\lambda}^{T0}\left(\mathbf{x}_0 - \mathbf{x}^0\right) + \sum_{n=0}^{N-1} \boldsymbol{\lambda}^{T(n+1)} g^n\left(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{u}^n\right) \tag{2.4}$$

For the moment, only the simulation equations are considered. Treatment of the other constraints is discussed later. The vectors $\boldsymbol{\lambda}^n$ are known as Lagrange multipliers, which can be thought of as elements of the dual space of the vector space to which $\mathbf{u}^n$ belongs. One Lagrange multiplier is required for each constraint with which the cost function is augmented. That is, the total number of Lagrange multipliers is equal to the product of the number of dynamic states and control steps. For example, if we have a two-phase black oil model with 3000 grid blocks and 400 control steps, the number of Lagrange multipliers is equal to $2\times3000\times400 = 2.4\times10^6$.

For optimality of the original problem as well as the augmented cost function, the first variation (or Frechet differential [38]) of the augmented cost function must equal zero. The first variation of $J_A$ is given by:

$$\begin{aligned}\delta J_A = &\left[\left.\frac{\partial \phi}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^N} + \frac{\partial L^{N-1}}{\partial \mathbf{x}^N} + \boldsymbol{\lambda}^{TN}\frac{\partial g^{N-1}}{\partial \mathbf{x}^N}\right]\delta\mathbf{x}^N + \sum_{n=0}^{N-1}\left[g^n\right]\delta\boldsymbol{\lambda}^{T(n+1)} + \left(\mathbf{x}_0 - \mathbf{x}^0\right)\delta\boldsymbol{\lambda}^{T0} \\ &+ \sum_{n=1}^{N-1}\left[\frac{\partial L^{n-1}}{\partial \mathbf{x}^n} + \boldsymbol{\lambda}^{T(n+1)}\frac{\partial g^n}{\partial \mathbf{x}^n} + \boldsymbol{\lambda}^{Tn}\frac{\partial g^{n-1}}{\partial \mathbf{x}^n}\right]\delta\mathbf{x}^n + \sum_{n=0}^{N-1}\left[\frac{\partial L^n}{\partial \mathbf{u}^n} + \boldsymbol{\lambda}^{T(n+1)}\frac{\partial g^n}{\partial \mathbf{u}^n}\right]\delta\mathbf{u}^n\end{aligned} \tag{2.5}$$

We observe that the total variation is a sum of the variations of $\mathbf{x}^n, \mathbf{u}^n$ and $\boldsymbol{\lambda}^n$. Since these variations are independent of one another, each of these terms must vanish for optimality [37, 38]. The $g^n\delta\boldsymbol{\lambda}^{T(n+1)}$ and $\left(\mathbf{x}_0 - \mathbf{x}^0\right)\delta\boldsymbol{\lambda}^{T0}$ terms are zero by definition. The terms involving $\delta\mathbf{x}^n$ can be made to vanish by choosing $\boldsymbol{\lambda}^n$ such that:

$$\begin{aligned}\boldsymbol{\lambda}^{Tn} &= -\left[\frac{\partial L^{n-1}}{\partial \mathbf{x}^n} + \boldsymbol{\lambda}^{T(n+1)}\frac{\partial g^n}{\partial \mathbf{x}^n}\right]\left[\frac{\partial g^{n-1}}{\partial \mathbf{x}^n}\right]^{-1} \quad \forall n = 1,\ldots,N-1 \\ \boldsymbol{\lambda}^{TN} &= -\left[\left.\frac{\partial \phi}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^N} + \frac{\partial L^{N-1}}{\partial \mathbf{x}^N}\right]\left[\frac{\partial g^{N-1}}{\partial \mathbf{x}^N}\right]^{-1} \quad \text{(Final Condition)}\end{aligned} \tag{2.6}$$

Equation (2.6) is known as the adjoint model. We notice that the Lagrange multipliers $\boldsymbol{\lambda}^n$ depend on $\boldsymbol{\lambda}^{n+1}$. Thus the Lagrange multipliers for the last control step must be calculated first according to the second equation above. It is for this reason that the adjoint model is solved backwards in time. With the Lagrange multipliers calculated in this manner, Equation (2.5) reduces to the following:

$$\delta J_A = \sum_{n=0}^{N-1}\left[\frac{\partial L^n}{\partial \mathbf{u}^n} + \boldsymbol{\lambda}^{T(n+1)}\frac{\partial g^n}{\partial \mathbf{u}^n}\right]\delta\mathbf{u}^n \tag{2.7}$$

Thus the required gradients of the cost function with respect to the controls are given as:

$$\frac{dJ}{d\mathbf{u}^n} = \frac{dJ_A}{d\mathbf{u}^n} = \left[ \frac{\partial L^n}{\partial \mathbf{u}^n} + \boldsymbol{\lambda}^{T(n+1)} \frac{\partial g^n}{\partial \mathbf{u}^n} \right] \qquad \forall n \in (0, ...., N-1) \tag{2.8}$$

If these gradients are zero for some value of $\mathbf{u}^n$, then optimality has been achieved with respect to $\mathbf{u}^n$, otherwise, these gradients could be used with any iterative gradient-based algorithm to determine the new search direction. The basic steps required for gradient-based optimization with adjoints are summarized as follows:

1. Solve the forward model equations for all time steps with given initial condition and initial control strategy. Store the dynamic states at each time step.

2. Calculate the cost function with results of the forward simulation.

3. Solve the adjoint model equations using the stored dynamic states to calculate the Lagrange multipliers with Equation (2.6).

4. Use the Lagrange multipliers to calculate the gradients using Equation (2.8) for all control steps.

5. Use these gradients with any optimization algorithm to choose new search direction and control strategy.

6. Repeat process until optimum is achieved, that is, all gradients are close enough to zero.

It is clear that one forward model evaluation and one adjoint model evaluation is required to calculate the gradients of the cost function, irrespective of the number of controls. The time required to solve the adjoint model is of the same order of magnitude as the forward simulation. Thus with this process, a time equivalent to approximately two simulations is all that is required to calculate any number of gradients (Table 2-1). This is why adjoint-based algorithms can be very efficient, and can potentially lead to huge time savings if the number of controls is large.

In the more general case when the time steps and control steps are not equivalent, that is, there is more than one time step in each control step, the production optimization problem can be formulated as follows:

$$\max_{\mathbf{u}^m}\left[ J = \sum_{m=0}^{N-1}\sum_{n=0}^{N_m-1} L^{m,n}\left(\mathbf{x}^{m,n+1},\mathbf{u}^m\right)\right] \forall m \in (0,..,N-1)$$

subject to:

$$\begin{aligned}
& g^{m,n}(\mathbf{x}^{m,n+1},\mathbf{x}^{m,n},\mathbf{u}^m) = 0 && \forall m \in (0,..,N-1) \\
& && n \in (0,..,N_m-1) \\
& \mathbf{x}^{0,0} = \mathbf{x}_{0,0} && \text{(Initial Condition)} \\
& \mathbf{A}\mathbf{u}^m \le \mathbf{b} && \forall m \in (0,..,N-1) \\
& \mathbf{LB} \le \mathbf{u}^m \le \mathbf{UB} && \forall m \in (0,..,N-1)
\end{aligned} \tag{2.9}$$

Here, $m$ is the control step index, and $n$ is the time step index within each control step, and $N_m$ is the number of time steps in control step $m$. Note that the $\phi$ term has been removed from the objective function for simplicity. The adjoint equations are given as:

$$\begin{aligned}
\boldsymbol{\lambda}^{Tm,n} &= -\left[\frac{\partial L^{m,n-1}}{\partial \mathbf{x}^{m,n}} + \boldsymbol{\lambda}^{Tm,n+1}\frac{\partial g^{m,n}}{\partial \mathbf{x}^{m,n}}\right]\left[\frac{\partial g^{m,n-1}}{\partial \mathbf{x}^{m,n}}\right]^{-1} \quad \forall n = 1,...,N_m-1, m = 0,...,N-1 \\
\boldsymbol{\lambda}^{Tm,N_m} &= -\left[\frac{\partial L^{m,N_m-1}}{\partial \mathbf{x}^{m,N_m}} + \boldsymbol{\lambda}^{Tm+1,1}\frac{\partial g^{m+1,0}}{\partial \mathbf{x}^{m,N_m}}\right]\left[\frac{\partial g^{m,N_m-1}}{\partial \mathbf{x}^{m,N_m}}\right]^{-1} \quad \forall m = 0,...,N-1 \text{ (Final Cond.)}
\end{aligned} \tag{2.10}$$

We see from the above that there is a final condition at the end of each control step with this formulation corresponding to the last time step of the control step, and this is obtained from the solution of the adjoint system of the first time step of the next control step. The gradients of the cost function with respect to the controls are finally given as:

$$\frac{dJ}{d\mathbf{u}^m} = \sum_{n=0}^{N_m-1}\left[\frac{\partial L^{m,n}}{\partial \mathbf{u}^m} + \boldsymbol{\lambda}^{Tm,n+1}\frac{\partial g^{m,n}}{\partial \mathbf{u}^m}\right] \quad \forall m \in (0,....,N-1) \tag{2.11}$$

Note that the following identities hold in the context of the above nomenclature:

$$\mathbf{x}^{m,N_m} = \mathbf{x}^{m+1,0}; \boldsymbol{\lambda}^{m+1,1} = \boldsymbol{\lambda}^{m,N_m+1} \tag{2.12}$$

This formulation is the actual implementation of the algorithm that is used for the examples demonstrated below.

### 2.3. Modified Algorithm for Adjoint Construction

Despite the great efficiency of the adjoint algorithm, a major drawback of the approach is that an adjoint code is required in order to apply the algorithm. The complexity of the adjoint equations is similar to that of the forward model [39]. Since in our case the forward model is the reservoir simulator, it is understandable why adjoint models have not gained popularity in the petroleum industry. We discuss a modified approach to constructing the adjoint that makes it relatively easy to create the adjoint code. The approach is possible due to certain properties of the fully implicit simulation code (in the current implementation) and the specific forms of the cost function used for production optimization.

The main ingredients of the adjoint equations given by Equation (2.6) are the two Jacobians of the simulation equations:

$$\frac{\partial g^n\left(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{u}^n\right)}{\partial \mathbf{x}^n}; \ \frac{\partial g^{n-1}\left(\mathbf{x}^n, \mathbf{x}^{n-1}, \mathbf{u}^n\right)}{\partial \mathbf{x}^n} \tag{2.13}$$

Of all the terms comprising the adjoint equations, these are the most difficult terms to calculate, as they are functions of the simulation equations. Now, during the forward simulation, at each time step (assume time step = control step for the moment), we solve Equation (2.14) to determine $\mathbf{x}^{n+1}$.

$$g^n\left(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{u}^n\right) = 0 \tag{2.14}$$

Since these equations are nonlinear with respect to $\mathbf{x}^{n+1}$, the usual method to solve them is through the Newton-Raphson algorithm [36]:

$$\left.\frac{\partial g^n}{\partial \mathbf{x}^{n+1}}\right|_{\mathbf{x}^{n+1}=\mathbf{x}^{n+1,k}} \left(\mathbf{x}^{n+1,k+1}-\mathbf{x}^{n+1,k}\right) = -g^n\left(\mathbf{x}^{n+1,k},\mathbf{x}^{n,k},\mathbf{u}^n\right) \tag{2.15}$$

Here, *k* is the iteration index of the Newton-Raphson algorithm at a given time step. At convergence of the algorithm, we observe that the Jacobian used is the same as the second Jacobian appearing in Equation (2.13). In order to obtain the first Jacobian given in Equation (2.13), consider the general form of the fully implicit mass balance equations:

$$g^n\left(\mathbf{x}^{n+1},\mathbf{x}^n,\mathbf{u}^n\right) = F^n\left(\mathbf{x}^{n+1}\right)+W^n\left(\mathbf{x}^{n+1},\mathbf{u}^n\right)-\frac{1}{\Delta t^n}\left[A^{n+1}\left(\mathbf{x}^{n+1}\right)-A^n\left(\mathbf{x}^n\right)\right] \tag{2.16}$$

Here *F* refers to the flux terms, *W* refers to the source terms and *A* refers to the accumulation terms. Thus the first Jacobian of Equation (2.13) is given as:

$$\frac{\partial g^n}{\partial \mathbf{x}^n} = \frac{1}{\Delta t^n}\frac{\partial A^n\left(\mathbf{x}^n\right)}{\partial \mathbf{x}^n} \tag{2.17}$$

Now, consider the mass balance equations of the previous time step:

$$g^{n-1}\left(\mathbf{x}^n,\mathbf{x}^{n-1},\mathbf{u}^{n-1}\right) = F^{n-1}\left(\mathbf{x}^n\right)+W^{n-1}\left(\mathbf{x}^n,\mathbf{u}^{n-1}\right)-\frac{1}{\Delta t^{n-1}}\left[A^n\left(\mathbf{x}^n\right)-A^{n-1}\left(\mathbf{x}^{n-1}\right)\right] \tag{2.18}$$

The second Jacobian for this time step is given by:

$$\frac{\partial g^{n-1}}{\partial \mathbf{x}^n} = \frac{\partial F^{n-1}\left(\mathbf{x}^n\right)}{\partial \mathbf{x}^n}+\frac{\partial W^{n-1}\left(\mathbf{x}^n,\mathbf{u}^{n-1}\right)}{\partial \mathbf{x}^n}-\frac{1}{\Delta t^{n-1}}\frac{\partial A^n\left(\mathbf{x}^n\right)}{\partial \mathbf{x}^n} \tag{2.19}$$

The last term of Equation (2.19), scaled with $\Delta t$ of the two time steps, is the same as the RHS of Equation (2.17). Thus the first Jacobian of any given time step is

calculated during the computation of the second Jacobian of the previous time step. The rest of the terms constituting the adjoint equations are relatively easy to calculate, as they are functions of the scalar cost function. In fact, if the cost function can be written in the following manner:

$$J = \sum_{n=0}^{N-1} L^n \left( \mathbf{u}^n, W^n, \Delta t^n \right) \tag{2.20}$$

That is, it is directly a function of the well terms of the simulation equations rather than a function of the dynamic states, then:

$$\frac{\partial L^{n-1}}{\partial \mathbf{x}^n} = f\left( \frac{\partial W^{n-1}}{\partial \mathbf{x}^n} \right); \ \frac{\partial \phi}{\partial \mathbf{x}^N} = 0 \tag{2.21}$$

The first term is a function of the converged well term derivatives with respect to dynamic states at each time step. This is also calculated within the forward simulation as seen from Equation (2.19) and is relatively easy to extract. An example of a cost function of the form of Equation (2.20) is net present value given by the following equation (see the Nomenclature for definition of symbols):

$$L^n \left( \mathbf{u}^n, W^n, \Delta t^n \right) = \sum_{j=1}^{N_P} \left[ \frac{P_{op}}{\rho_{o,SC}} W_{o,j}^n - \frac{C_{wp}}{\rho_{w,SC}} W_{w,j}^n \right] \frac{\Delta t^n}{\left(1+\alpha\right)^{t^n}} - \sum_{j=1}^{N_I} C_{wi} q_{wi,j}^n \frac{\Delta t^n}{\left(1+\alpha\right)^{t^n}} \tag{2.22}$$

Thus we see that all the terms required for calculating the Lagrange multipliers through the adjoint model can be calculated during the forward model evaluation itself. Furthermore, if NPV is the cost function, and BHP or rates are the controls, then the terms of Equation (2.8) can also be extracted from the forward run:

$$\frac{\partial L^n}{\partial \mathbf{u}^n} = f\left( \mathbf{u}^n, \frac{\partial W^n}{\partial \mathbf{u}^n} \right); \ \frac{\partial g^n}{\partial \mathbf{u}^n} = \frac{\partial W^n}{\partial \mathbf{u}^n} \tag{2.23}$$

Therefore, the algorithm for gradient-based optimization using adjoints is modified as follows:

1. Solve the forward model equations for all time steps with given initial condition and initial control strategy.

2. Store the two Jacobians of the simulation equations and the well derivatives at each time step.

3. Calculate the cost function with results of the forward simulation.

4. Solve the adjoint model equations using the stored Jacobians and well derivatives with respect to dynamic states to calculate the Lagrange multipliers with Equation (2.6).

5. Use the Lagrange multipliers and stored well derivatives with respect to controls to calculate the gradients using Equation (2.8) for all control steps.

6. Use these gradients with any optimization algorithm to choose new search direction and control strategy.

7. Repeat process until optimum is achieved, that is, all gradients are close to zero.

It should be noted that mathematically there is no change in the algorithm, but the adjoint equations become much simpler to code. Specifically, the version of the General Purpose Research Simulator (GPRS) [11] developed at Stanford University and used as the forward simulator in the following examples consists of around 20000 lines of C++ code, whereas the adjoint code consists of only around 500 lines of Matlab code. However, more importantly, this approach allows the adjoint code to remain fully consistent with the forward model code if any changes to the flux terms or accumulation terms are made or new terms reflecting new physics are added to

Equation (2.16). This is because the Jacobians are taken directly from the forward model, so any changes made to the simulation equations are reflected in these.

For example, suppose a dual porosity model [40] is implemented into the forward simulator. This implies that a new term (the transfer function) is added to the simulation equations, changing $g^n\left(\mathbf{x}^{n+1},\mathbf{x}^n,\mathbf{u}^n\right)$. No change is required in the adjoint model code, however, as the partial derivatives of the simulation equations with respect to the states and controls are taken directly from the forward simulator. This is a useful algorithmic feature in general, but it is particularly important in a research or development setting where the forward model is updated frequently. Further, the modified approach is slightly more efficient than the standard approach as the Jacobian forming calculations are not repeated but are directly loaded from memory. However, because the Jacobians must be stored with the modified approach instead of the dynamic states, the storage requirement of the modified approach is much larger. The approximate memory requirements can be estimated with the following equation:

$$Total\ GB \approx 1.6\left(N_D+1\right)N_g N_c^2 N_t / 10^8 \tag{2.24}$$

Here, $N_D$ is the number of physical dimensions, $N_g$ is the number of grid blocks, $N_c$ is the number of components and $N_t$ is the number of time steps. For example, if a given problem is a 3D, 3-phase black oil model with 100000 cells and 100 time steps, the total storage requirement is around 6 GB. Note that this is hard disk memory and not RAM memory, as the Jacobians are stored to files in our implementation.

## 2.4. Case Study – Horizontal Smart Wells

The first case is a simple example adapted from Brouwer and Jansen [34] that effectively demonstrates the applicability of adjoint-based optimization to smart well control. The schematic of the reservoir and well configuration is shown in Figure 2-3. The model consists of one horizontal "smart" water injector and one horizontal "smart" producer, each having 45 controllable segments. The reservoir covers an area

of 450 × 450 $m^2$ and has a thickness of 10 m and is modeled by a 45 × 45 × 1 horizontal 2D grid. The fluid system is an essentially incompressible two-phase unit mobility oil-water system, with zero connate water saturation and zero residual oil saturation. Figure 2-4 shows the heterogeneous permeability field with two high permeability streaks running from the injector (left) to the producer (right). The contrast in permeability between the high permeability streaks and the rest of the reservoir is around a factor of 20-40, and it is this heterogeneity that makes the optimization results interesting.

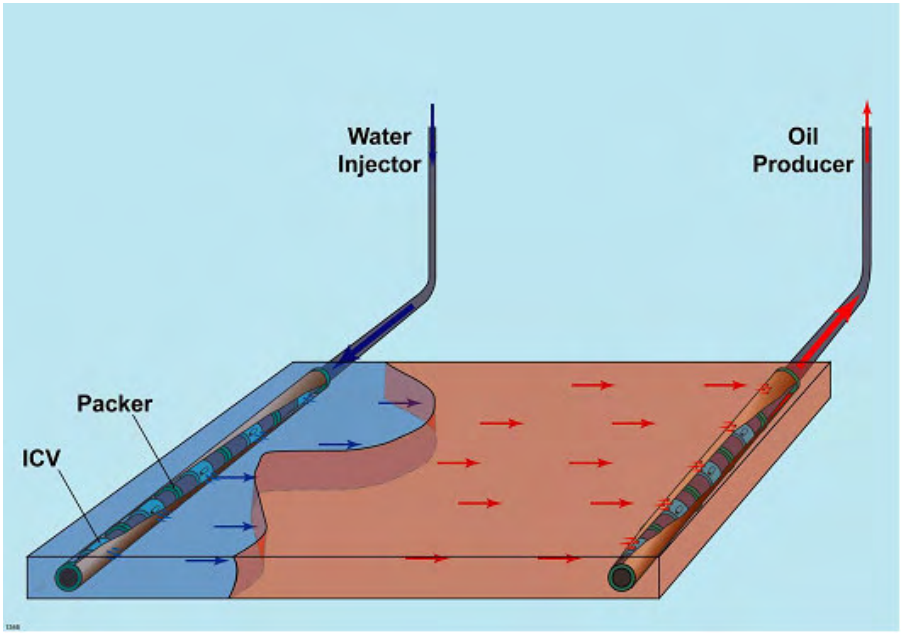


Figure 2-3 Schematic of reservoir and wells for Example 1 (From Brouwer and Jansen [34])

For purpose of optimization, the injector segments are placed under rate control, and the producer segments are under BHP control. There is a total injection constraint of 2700 STB/day (STBD); thus the optimization essentially results in a redistribution of this water among the injection segments. Further, there are also bounds on the minimum and maximum rates allowed per segment, and also bounds on the BHPs of the producers, which could for example correspond to bubble point pressures or fracture pressures. The model is produced until exactly one pore volume of water is injected, which corresponds to around 950 days of injection. This time period is divided into five control steps of 190 days each. Thus the total number of controls is

equal to (45+45) × 5 = 450. All constraints in this problem are linear with respect to the controls.

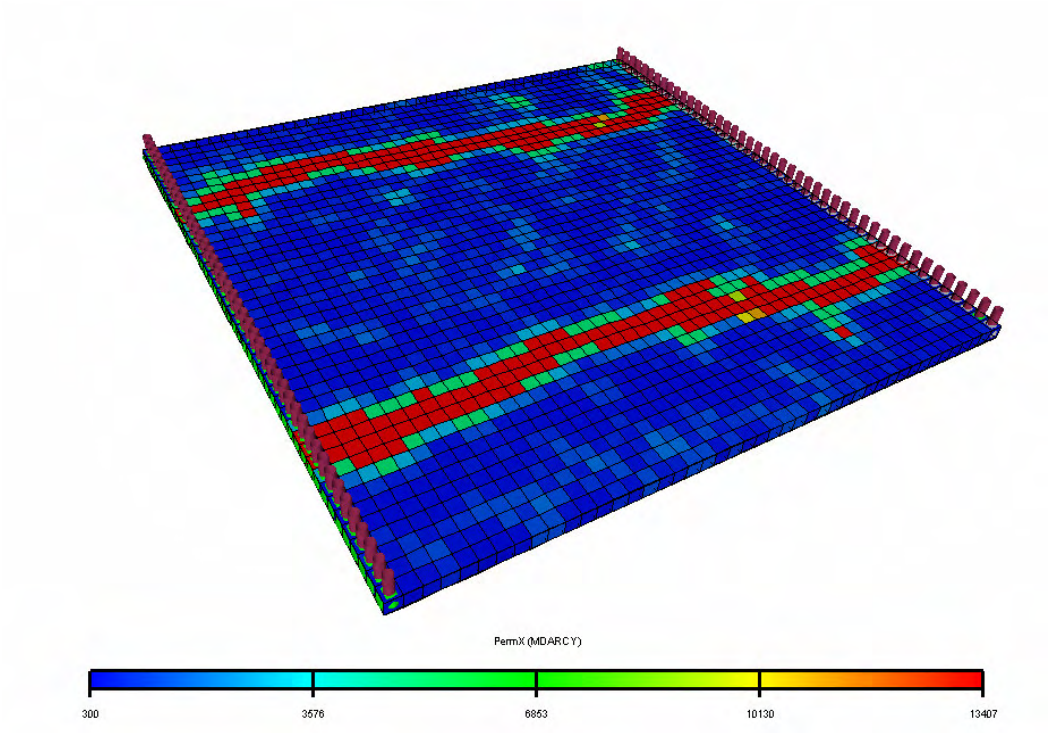


Figure 2-4 Permeability field for Example 1 (From Brouwer and Jansen [34])

In order to understand the benefit of any optimization process, it is usual to compare the optimization results against a base or reference case. In the case of production optimization, such a base case would be a reasonable production strategy that an engineer might devise given a simulation model and a set of constraints. It is, however, very difficult (and often nonintuitive) to understand the implications of varying well controls on the optimization process. It is thus usual for engineers to specify constant production/injection rates or BHPs until some detrimental reservoir response such as water breakthrough is observed. For this case study as well, the base case is kept quite simple.

For the purpose of this case study, the base case is a constant rate/constant BHP production strategy. The 2700 STBD of injection water is distributed among the 45 injection segments according to their *kh* (permeability × pay thickness), which

corresponds to an uncontrolled case. The producer BHPs are set in such a way that a balanced injection-production is obtained.

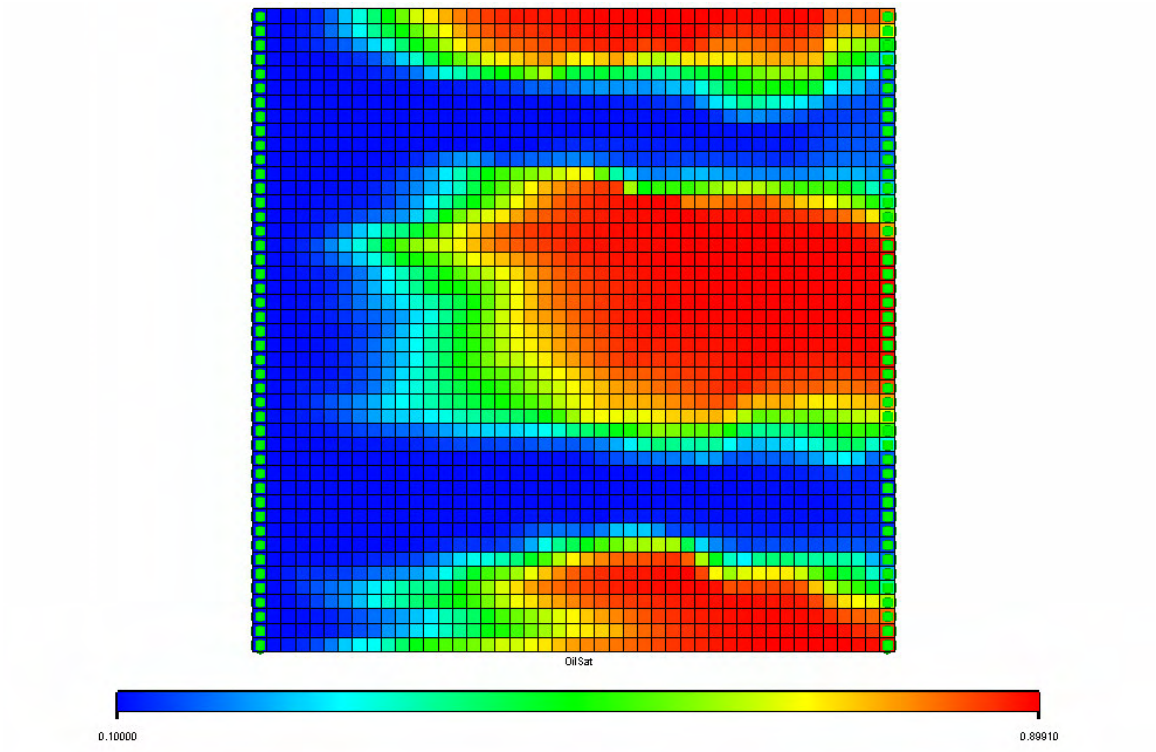


Figure 2-5 Final oil saturations after 1 PV injection for reference case

The objective of the optimization process is to maximize NPV as given in Equation (2.22). The NPV discounting factor is set to zero, meaning that the effect of discounting is neglected. Thus, maximizing NPV is essentially maximizing cumulative oil production and minimizing cumulative water production. The oil price is conservatively set at \$80/$m^3$, water injection costs at \$0/$m^3$, and water production costs at \$20/$m^3$ [34]. It should be noted that it is relatively easy to vary these cost/prices with time and even to implement uncertainty models for them, provided of course that such models can be inferred.

Starting from 100% oil saturation throughout the reservoir, Figure 2-5 and Figure 2-6 show the final oil saturations for the uncontrolled and the optimized case after exactly 1 PV of water has been injected. It is clear that the optimization leads to a huge improvement in the sweep efficiency, leading to the increase in NPV of around 100%.

Note that the final oil saturations maps obtained are very similar to those obtained by Brouwer and Jansen [34].

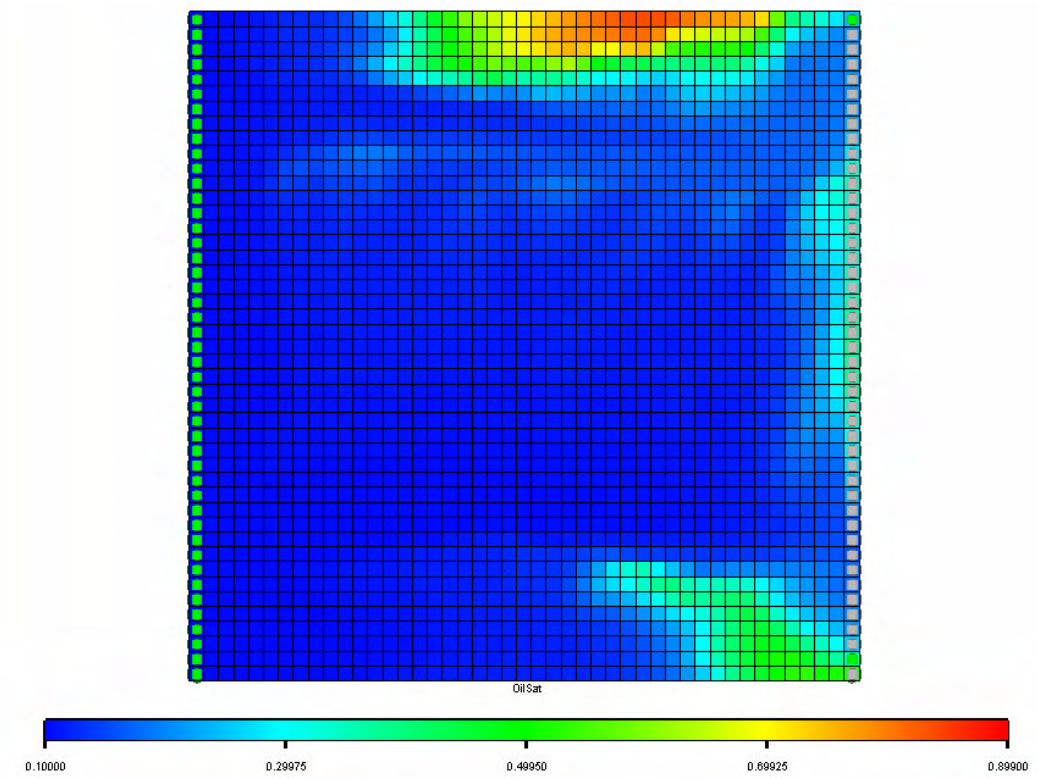


Figure 2-6 Final oil saturations after 1 PV injection for optimized case

The reasons behind the better sweep in the optimized case can be easily explained by analyzing the optimized trajectories of the controls – rates/ BHPs of the injectors and producers – as seen in Figure 2-7 and Figure 2-8. The *y*-axis of Figure 2-7 corresponds to 45 injector segments and the *x*-axis corresponds to the 5 control steps. The color scale corresponds to injection rates of the segments, with blue being lowest rates (almost closed) and brown being highest (fully open). It is obvious that the injector segments completed in or near the high permeability streaks are shut down at early time, as they force water to move very quickly toward the producers, resulting in early breakthrough and thus poor sweep. Further, some of the injector segments completed in the lower streak tend to open up towards the later control steps, so that the water front between two streaks actually moves laterally (Figure 2-9) towards the streaks, resulting in the almost 100% sweep of this region.

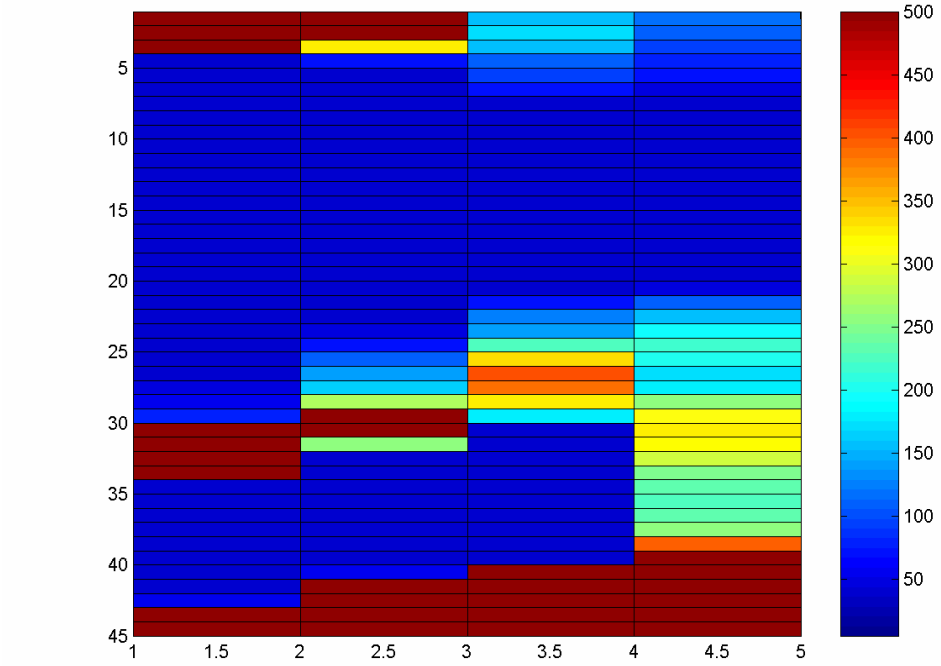


Figure 2-7 Injection rate variation with time for optimized case (*x*-axis represents the control steps, and *y*-axis represents the 45 injector segments)

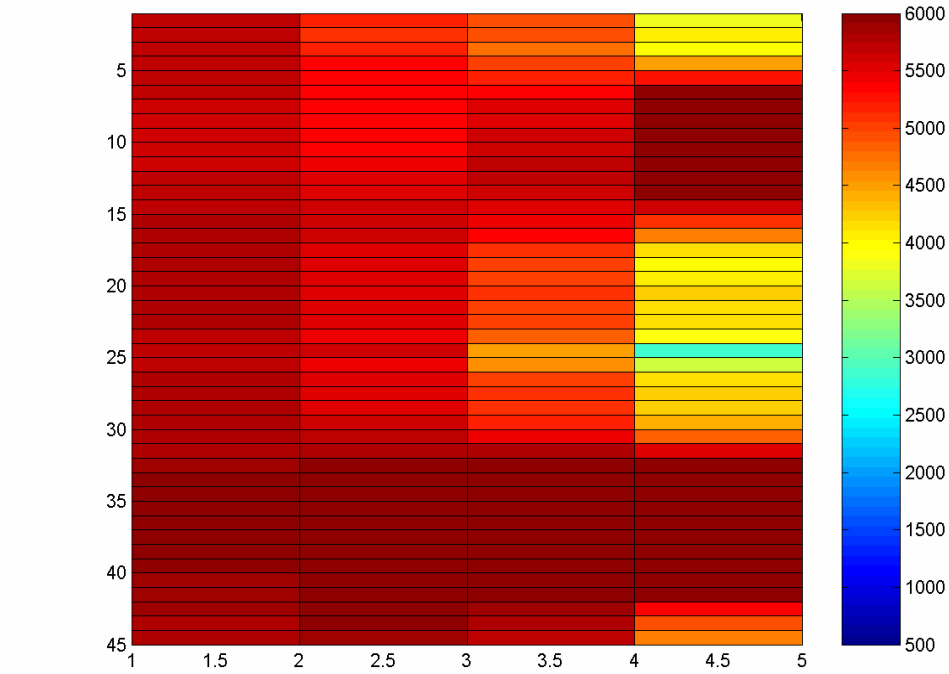


Figure 2-8 Producer BHP variation with time for optimized case (*x*-axis represents the control steps, and *y*-axis represents the 45 producer segments)

In Figure 2-8, the color scale corresponds to the BHPs of the producer segments, and we again observe that the producer segments completed in or near the high

permeability streaks are shut most of the time. Also, the producer segments in between the streaks again open towards the later control steps, so that the water present in the streaks moves toward this region leading to a better sweep.

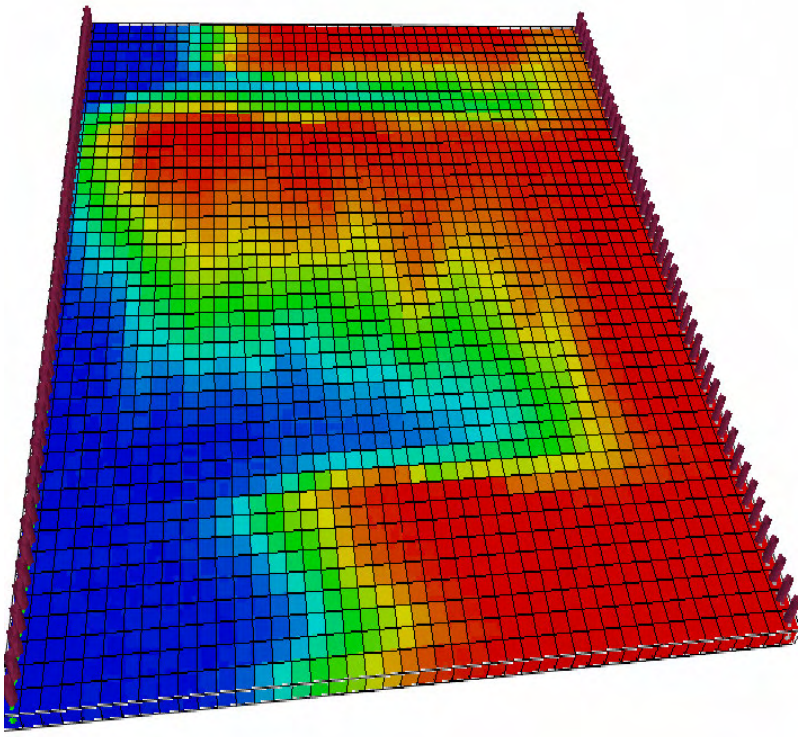

Figure 2-9 Lateral movement of water in optimized case

Figure 2-10 shows the field water injection rate, the oil production rate and the water production rate for the base case and optimized case. In the optimized case, for most of the time, the oil production rate is significantly higher and the water production lower than the base case. The four peaks in the oil rate correspond to the start of the control steps. Because the actual time steps are smaller than the control steps, the oil or water production rate over a control step does not remain the same. It is interesting to note that although the final saturation map and increase in NPV are quite close to that obtained by Brouwer and Jansen [34], the optimal trajectories of the controls are somewhat different. These differences may be caused by the different control variables used on wells, slightly different fluid compressibilities, or different lengths of control steps, or it may be that the local optima obtained are different, but with similar values of the optima.

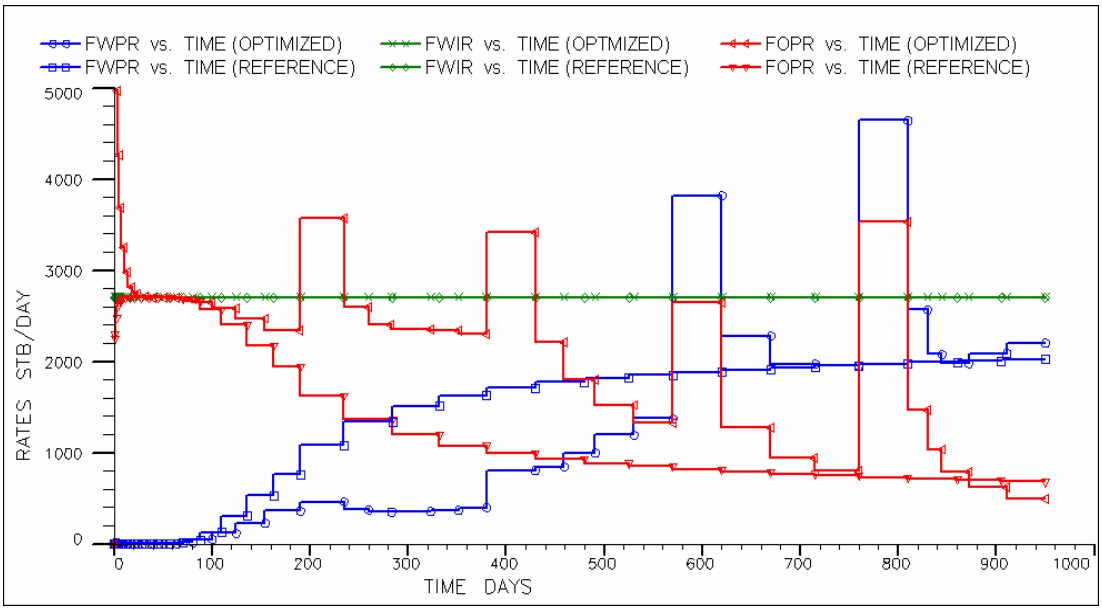


Figure 2-10 Comparison of total production rates for reference and optimized case

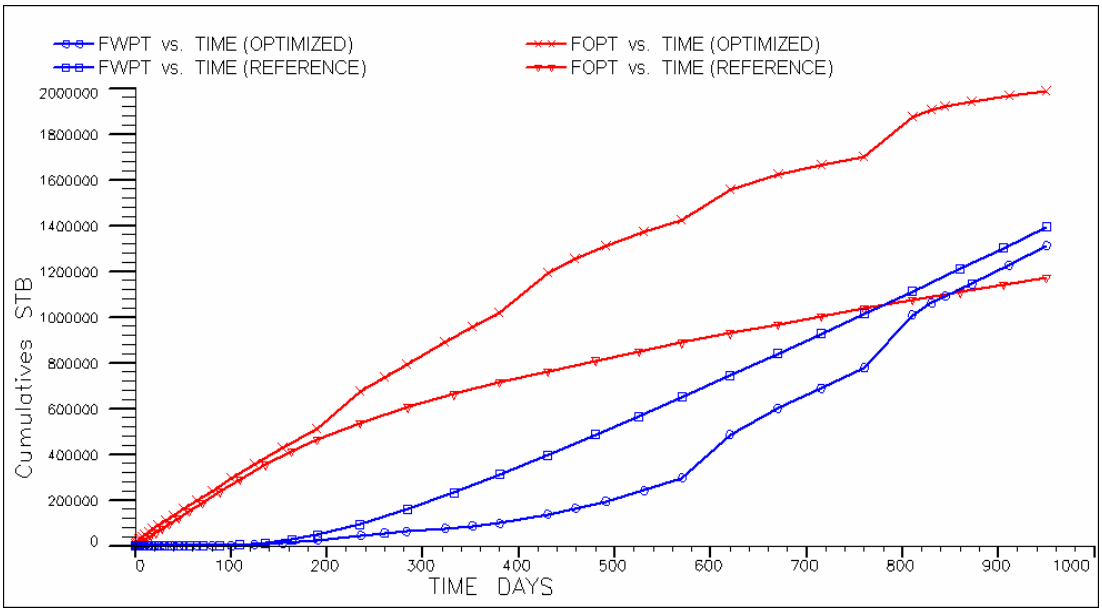


Figure 2-11 Comparison of cumulatives for reference and optimized case

Figure 2-11 shows that there is a substantial increase in cumulative oil production (70%), attributed to the better sweep, and a slight decrease in water production (6%) after the optimization process. The optimization process required five iterations of the optimization algorithm (Sequential Quadratic Programming from the Matlab Optimization Toolbox [41]); the total number of simulations required for the optimization was around 12-15.

## 2.5. Case Study – SPE 10 Layer 61

This example is a more complex case adapted from the 10th SPE Comparative Solution Project [42]. The model is described on a regular Cartesian grid. The model dimensions are $1200 \times 2200 \times 170$ (ft). The top 70 ft (35 layers) represents the Tarbert formation and the bottom 100 ft (50 layers) represents Upper Ness formation. The cell size is 20 ft × 10 ft × 2 ft. The model has $60 \times 220 \times 85$ cells and consists of part of a Brent sequence. For this case study, only layer 61 belonging to the Upper Ness fluvial formation is selected. The highly heterogeneous permeability field for this layer is shown in Figure 2-12.

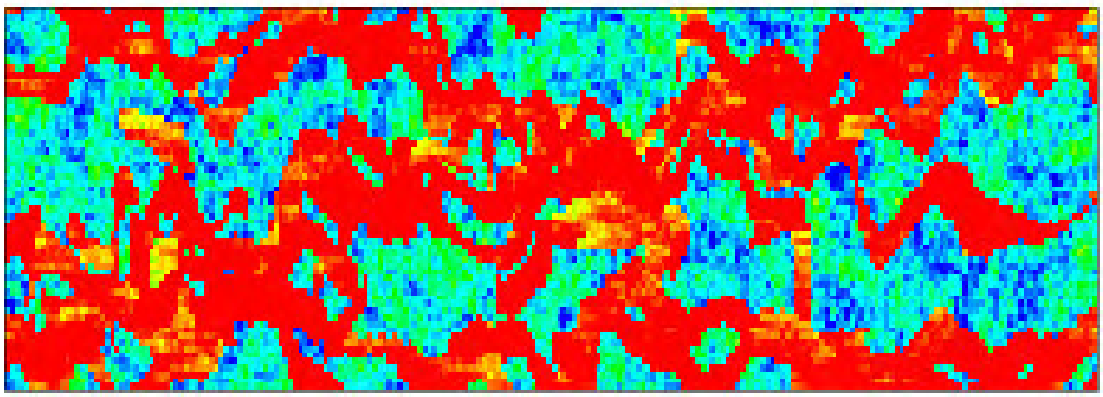

Figure 2-12 Permeability field for SPE 10 Layer 61

The fluid system is a two-phase oil water system with low compressibility and a very unfavorable mobility ratio (≅ 75). The relative permeabilities are the usual Corey type curves. The production model is a five-spot pattern with a vertical injector at the center and four vertical producers at the corners. Thus there are two drive mechanisms: depletion drive and water injection. The injector is under rate control and the producers are under BHP control. The reference case is again a constant rate/BHP case, with the injection rate set at 5000 STBD and all producer BHPs at 3000 psi. The injection rate is constrained at a maximum of 6000 STBD and the producer BHPs at a minimum of 500 psi and a maximum of 4000 psi. These are linear constraints with respect to the controls.

The objective is to maximize the NPV over a period of 4.5 years, divided into control periods of 30 days. Thus the total number of controls is equal to 5×55 = 275. The NPV discounting factor is again set at zero, meaning that the effect of discounting is neglected. The oil price is conservatively set at $30/Bbl and water injection costs at $3/Bbl. The case is made interesting by making it extremely expensive to process produced water, and the water production cost is set at $40/Bbl. Although this high water production cost is unrealistic in most settings, it makes the optimization process more complex and different from the first case, as the optimization must here reduce water production, even at the cost of reducing oil production.

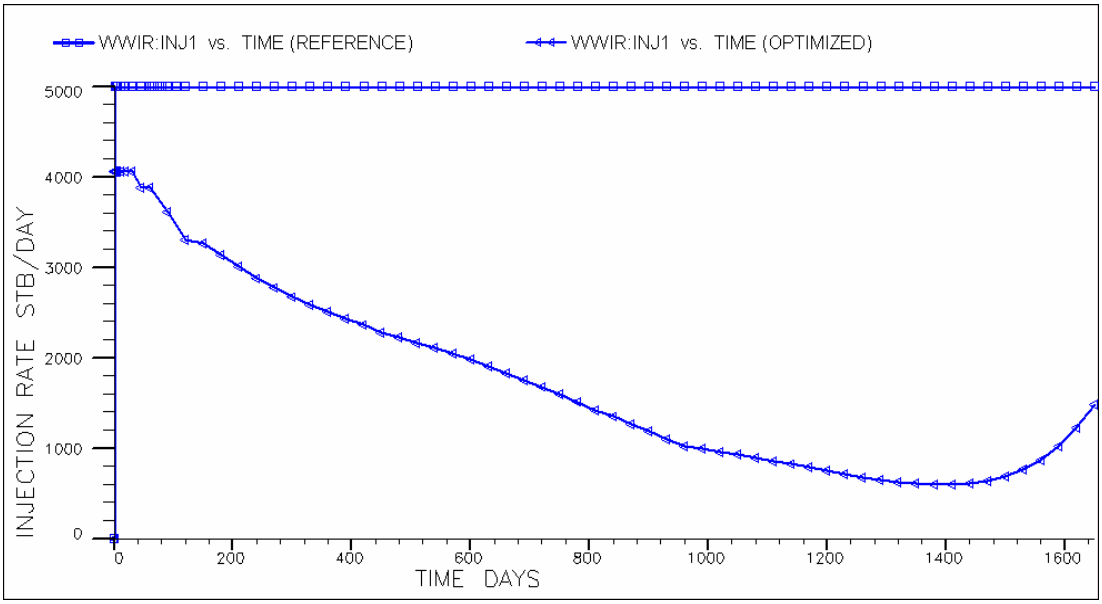


Figure 2-13 Comparison of injection rates for reference and optimized case

Figure 2-13 compares the injection rate of the optimized case to that of the reference case. There is a substantial decrease in injection rate as expected. Due the extremely high water production costs, water injection is reduced so that water production may be reduced. However, in order to maintain the oil production rate, all of the producers except PROD 1 produce at around the minimum BHP (Figure 2-14), that is, they are fully open. The main drive mechanism thus changes from water injection in the reference case to depletion drive in the optimized case. PROD 1 on the other hand is producing the maximum amount of water in the reference case as seen in Figure 2-15.

Thus, in the optimized case, instead of a reduction of BHP, in an attempt to curtail the high watercut, the BHP of PROD 1 increases, and in fact goes towards the maximum allowed BHP at the end of the run (Figure 2-14).

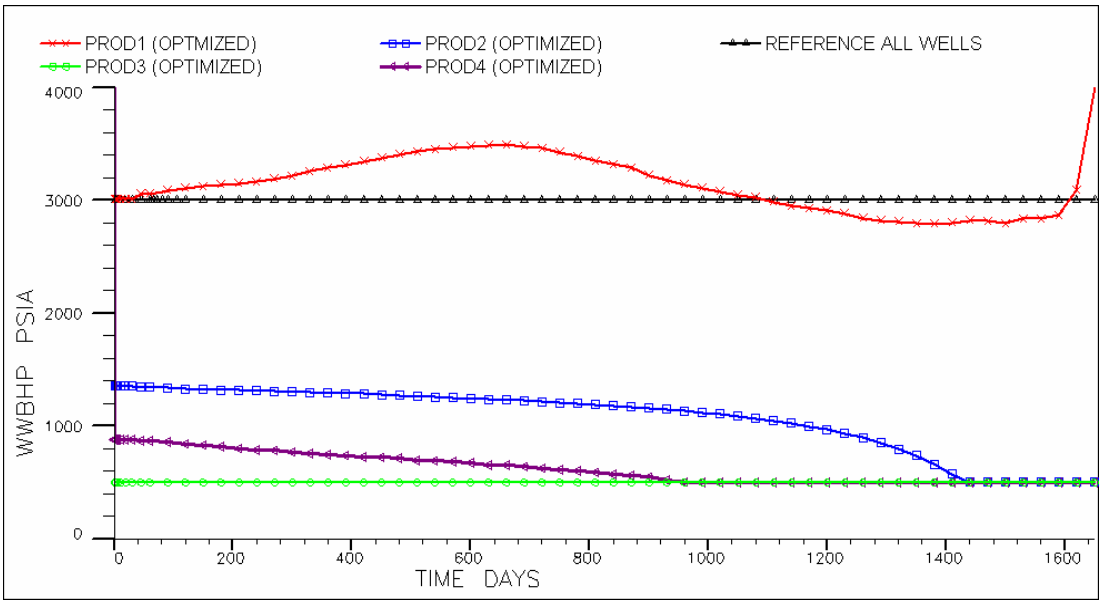


Figure 2-14 Comparison of BHPs of producers for reference and optimized case

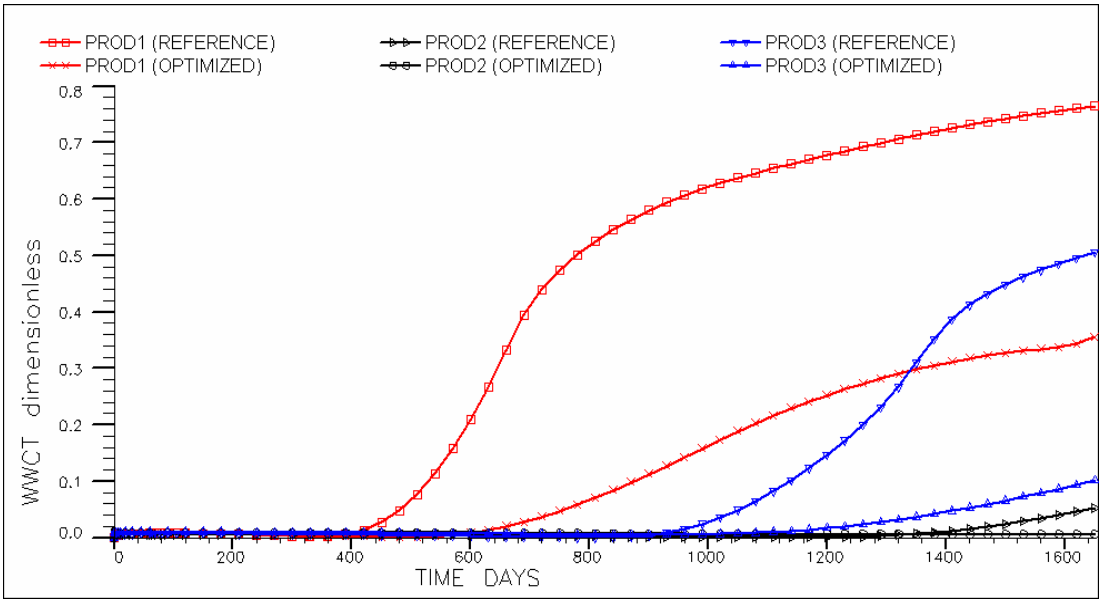


Figure 2-15 Comparison of watercuts for some producers for reference and optimized case

Figure 2-16 compares the cumulative water injection, oil production and water production of the base case and optimized case. It is clear that the increase in NPV (by 200%) is due to the huge reduction in water injection (67%) and production (19%), while the cumulative oil production is not much reduced (11%). It is interesting to compare the final oil saturation maps for the reference and optimized case, as seen in Figure 2-17 and Figure 2-18. Unlike the previous case, the optimization results in a decrease of the overall sweep, as the water injected is much less, but the channels leading to PROD 2 and PROD 4 show a slightly better sweep, implying that the injected water is distributed more evenly compared to the reference case.

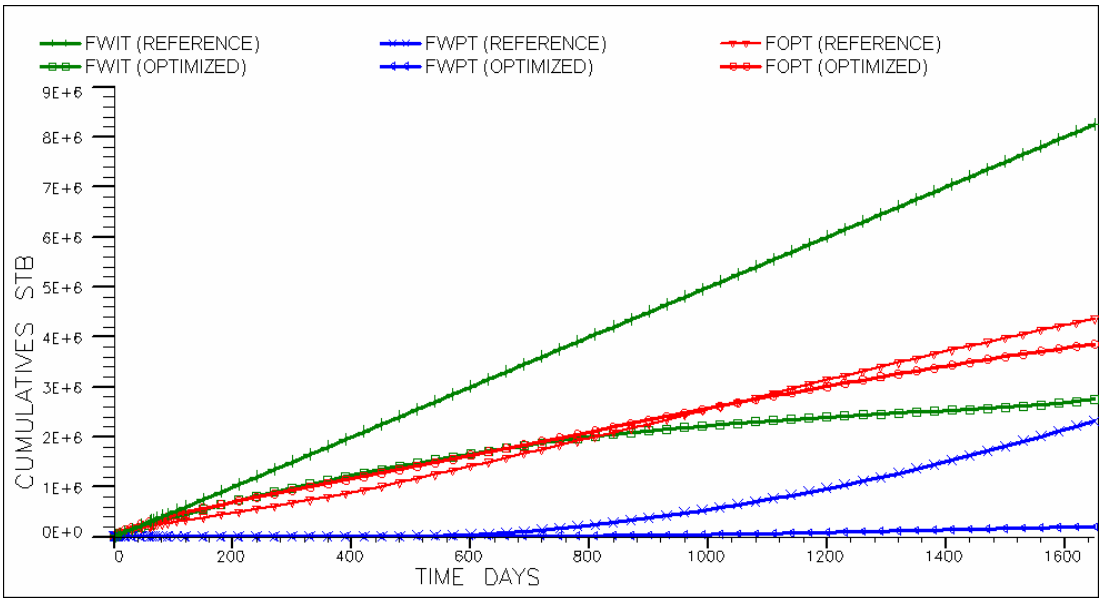


Figure 2-16 Comparison of cumulatives for reference and optimized case

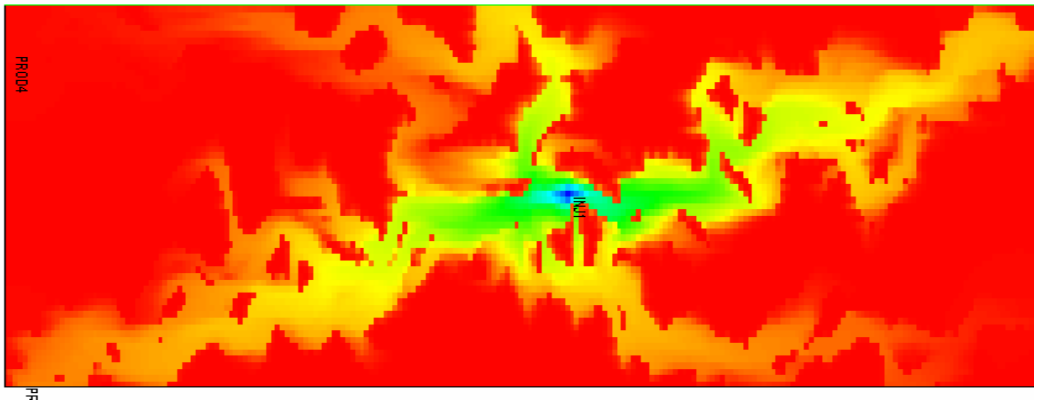


Figure 2-17 Final oil saturation map for reference case

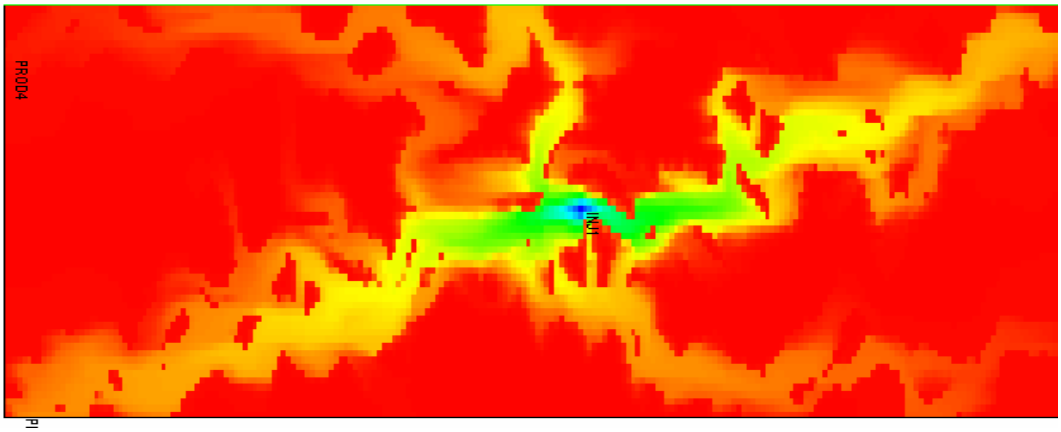


Figure 2-18 Final oil saturation map for optimized case

The number of iterations of the Sequential Quadratic Programming [41] algorithm required for the above optimization was 10, which corresponds to around 25-30 simulation runs. Average run time for one simulation on a 2.8 GHZ machine with 4 GB RAM was around 10 minutes, and the total optimization time was around 5.7 hours.

## 2.6. Summary

This chapter demonstrated the application of adjoint models for the efficient calculation of gradients. A modified approach for the construction of the adjoint model was also described, provided a fully implicit forward model is available, and the cost function can be cast into certain forms. It was shown that the modified approach makes it relatively easy to code the adjoint as compared to the standard approach. Further, another important advantage of this technique is that the consistency between the adjoint code and the forward model code is automatically maintained if any changes to the flux terms or accumulation terms are made or new terms reflecting new physics are added to the forward model. Two dynamic waterflood examples were discussed demonstrating the practicality and efficiency of the approach.

# Chapter 3

# 3. Adjoint-based Optimal Control and Model Updating

As discussed in the first chapter, the closed-loop approach for efficient realtime optimization consists of three key components: efficient optimization algorithms, efficient model updating algorithms, and techniques for uncertainty propagation. In this chapter, we discuss a gradient-based model updating technique, and combine it with the optimal control algorithm presented in the last chapter to obtain a simplified implementation of the closed-loop approach. Uncertainty propagation is not considered here. Neglecting uncertainty propagation essentially means that the closed-loop process is applied to a single realization of the uncertain parameters; e.g., the maximum likelihood estimate, which is updated at every control step. Such a procedure can be expected to provide near-optimal results in many cases, though the treatment of uncertainty will of course be important in many applications. Uncertainty propagation and the entire loop are discussed in Chapter 4.

Within the context of closed-loop reservoir management, Brouwer et al. [9] used adjoint models for optimization and Kalman filters for model updating. Adjoint models allow for very efficient optimization, although their implementation can be complicated. The ensemble Kalman filter has only recently been applied for history matching [43]. Although this approach is straightforward to implement [43,44], its efficiency compared to established methods like adjoint models may be an issue. Aitokhuehi and Durlofsky [45] used conjugate gradient algorithms with numerical gradients for optimization and the probability perturbation method [46] for model updating. The use of numerical gradients and the stochastic probability perturbation

method makes the implementation quite easy, but both algorithms are very expensive computationally, which may limit the use of this procedure in practical settings.

In this chapter, we apply the adjoint methods described in the last chapter for the efficient calculation of gradients of the objective function with respect to the controls, which are then used by gradient-based optimization algorithms for the optimization part of the closed-loop. For the model updating procedure, we use Bayesian inversion theory and apply an efficient parameterization of the permeability field using the Karhunen-Loeve (K-L) expansion. This allows us to describe the uncertain parameter field (e.g., permeability), in terms of two-point statistics, using very few parameters. A key advantage of the K-L description is that these parameters can be varied continuously while maintaining the underlying geostatistical description. As a result, adjoint techniques can be applied for the history matching while preserving some degree of geological realism. This procedure is much faster than stochastic algorithms and, unlike standard gradient-based algorithms, implicitly honors the two-point statistics of the geological model. An extension of the K-L procedure described in this chapter which preserves multi-point statistics will be presented in Chapter 6.

The use of adjoints for history matching was pioneered by Chen et al. [47] and Chavent et al. [48], who applied it to single-phase problems. Since then, many other researchers have modified and improved the application of adjoint models for multiphase history matching including Wasserman et al. [49], Watson et al. [50], Wu et al. [51], Li et al. [35], Wu and Datta-Gupta [52], and Zhang et al. [53]. Gavalas et al. [15] introduced the use of an eigenfunction expansion for efficient parameterization of reservoir properties, which was also used later by Oliver [54] and Reynolds et al. [55].

This chapter starts with a description of the model updating algorithm. It is then combined with the optimization algorithm of the last chapter in a sequential manner to perform optimization for an uncertain reservoir description. Two variants of the implementation of the model updating algorithm within the closed-loop are discussed.

Next, the efficiency and applicability of this approach is demonstrated through a realtime dynamic waterflood optimization of a synthetic reservoir under production constraints and with an uncertain permeability field. The closed-loop optimization methodology is shown to provide a substantial improvement in NPV and sweep efficiency over the base case, and the results are quite close to those obtained with known geology.

### 3.1. Model Updating as a Minimization Problem

A number of techniques are available for the history matching problem. Gradient-based procedures are in general very efficient, but standard implementations suffer from two limitations. First, they tend to find local rather than global minima. Although this is the case to some extent with all history matching algorithms, stochastic optimization techniques introduce a random component to sample the parameter space more broadly. The second limitation inherent in standard gradient-based techniques is that geological constraints are not preserved. This occurs because, during the optimization, geostatistical correlations between model parameters are not maintained.

The technique applied here circumvents this latter difficulty by introducing an efficient parameterization of the permeability field in terms of the Karhunen-Loeve expansion (the K-L expansion, described in more detail below, is essentially an eigenfunction expansion) [15, 54, 55]. Because the parameters appearing in the K-L expansion are uncorrelated, any set of these parameters provides a permeability field that honors the underlying two-point geostatistics. Thus, these parameters can be varied in any way to achieve a history match. The technique is much more efficient than stochastic search procedures and has the additional advantage that much of the adjoint code developed for the production optimization problem can also be applied with some modifications to the history matching problem.

The model updating component of the closed-loop is a problem of inversion of production data (well pressures and flow rates) in order to determine reliable estimates

of uncertain model parameters (porosity and permeability). Within the context of Bayesian inverse modeling, the solution to the general inverse problem consists of combining all prior information as given by the prior probability density of the observed data $\mathbf{d}$, given by $\rho_D(\mathbf{d})$, the prior probability density of the model parameters $\mathbf{m}$, given by $\rho_M(\mathbf{m})$ and the forward model $\mathbf{d} = f(\mathbf{m})$ to determine the posterior probability density of the model parameters $\sigma_M(\mathbf{m})$ given by the following general equation [13]:

$$\sigma_M(\mathbf{m}) = k\rho_M(\mathbf{m})\rho_D(f(\mathbf{m})) \tag{3.1}$$

where $k$ is a normalization constant. The most general approach for solving any nonlinear inverse problem involves determining the entire probability distribution $\sigma_M(\mathbf{m})$, which requires an extensive exploration of the model space, usually accomplished using random search techniques such as a Monte Carlo method [13].

For the case of history matching, however, solving the forward model $\mathbf{d} = f(\mathbf{m})$ is usually quite time consuming, and in practical cases a single evaluation can take several hours of computation. Therefore we must often be satisfied with the determination of the maximum likelihood and a reasonable estimate of the dispersion of the distribution around it. This is generally accomplished through least-squares techniques, which is a special case of Equation (3.1), when both the prior probability densities $\rho_M(\mathbf{m})$ and $\rho_D(\mathbf{d})$ are Gaussian. Under these assumptions, the model updating problem reduces to the minimization of the following misfit function, where $\mathbf{C}_D$ and $\mathbf{C}_M$ are the data and parameter prior covariance matrices [13]:

$$S(\mathbf{m}) = \left(f(\mathbf{m}) - \mathbf{d}_{obs}\right)^T \mathbf{C}_D^{-1}\left(f(\mathbf{m}) - \mathbf{d}_{obs}\right) + \left(\mathbf{m} - \mathbf{m}_{prior}\right)^T \mathbf{C}_M^{-1}\left(\mathbf{m} - \mathbf{m}_{prior}\right) \tag{3.2}$$

In the general problem of model updating, the forward model $\mathbf{d} = f(\mathbf{m})$ (simulation equations and outputs) is not only a function of the model parameters $\mathbf{m}$, but is also a

function of the dynamic states $\mathbf{x}$ (grid pressures, saturations, etc.) and a set of controls $\mathbf{u}$ (well rates, bottom hole pressures etc.), that is: $\mathbf{d} = f\left(\mathbf{x}^{n+1}, \mathbf{u}^n, \mathbf{m}\right)$, where $n$ is the time step index ($\mathbf{m}$ is assumed to be time invariant). The dynamic states $\mathbf{x}$ are functions of both $\mathbf{m}$ and $\mathbf{u}$. Thus the mathematical formulation of the general model updating problem is as follows:

$$\min_{\mathbf{m}} \left[ S = \left(\mathbf{m} - \mathbf{m}_{prior}\right)^T \mathbf{C}_M^{-1} \left(\mathbf{m} - \mathbf{m}_{prior}\right) + \sum_{n=0}^{N-1} L^n \left(\mathbf{x}^{n+1}, \mathbf{u}^n, \mathbf{m}\right) \right] \forall n \in (0,..,N-1)$$

subject to:

$$g^n(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{u}^n, \mathbf{m}) = 0 \qquad \forall n \in (0,..,N-1) \tag{3.3}$$

$$\mathbf{x}^0 = \mathbf{x}_0 \qquad \text{(Initial Condition)}$$

$$\mathbf{m} \in \text{Geologically Consistent Realizations}$$

The Lagrangian $L^n\left(\mathbf{x}^{n+1}, \mathbf{u}^n, \mathbf{m}\right)$ has the following form for the history matching problem (assuming that measurement uncertainties are independent):

$$L^n\left(\mathbf{x}^{n+1}, \mathbf{u}^n, \mathbf{m}\right) = \sum_{i=1}^{N_w} \left\{ \frac{f_i^n\left(\mathbf{x}^{n+1}, \mathbf{u}^n, \mathbf{m}\right) - \mathbf{d}_{obs_i}^n}{\sigma_i^n} \right\}^2 \tag{3.4}$$

Here $\sigma_i^n$ is the standard deviation of the measured data and $N_w$ is the number of wells. The set of equations $g^n(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{u}^n, \mathbf{m}) = 0$ together with the initial condition $\mathbf{x}^0 = \mathbf{x}_0$ refers to the reservoir simulation equations (forward model) which constrain the dynamic states $\mathbf{x}$. The geological constraints (last set of constraints in Equation (3.3)) are required because production data on its own is not fully constraining. Even with these geological constraints the system is still not fully constrained, but the use of these constraints guarantees that the resulting $\mathbf{m}$ will be consistent with the geostatistical description.

### 3.2. Bi-orthogonal Expansions and Adjoints for Updating

As mentioned before, standard gradient-based algorithms will not maintain the necessary geological constraints. As a result, although the objective function might be reduced by a large amount, the final $\mathbf{m}$ may not be geologically realistic, which will generally result in poor predictions. By introducing the K-L expansion of the random parameter field $\mathbf{m}$, the problem defined in Equation (3.3) is transformed such that the geological constraints (as defined by the covariance matrix of $\mathbf{m}$) are implicitly honored. In addition, the number of parameters defining $\mathbf{m}$ is decreased significantly, resulting in a greater reduction of the uncertainty envelope compared to the direct solution of the original problem [54, 55].

The Karhunen-Loeve expansion is a very powerful tool for representing stationary and non-stationary processes with explicitly known covariance functions. Any random field or process can be represented as a series expansion involving a complete set of deterministic functions with corresponding random coefficients [56]. This method provides a second-moment characterization in terms of random variables and deterministic functions. The use of K-L expansion with orthogonal deterministic basis functions and uncorrelated random coefficients has generated interest because of its bi-orthogonality property, that is, both the deterministic basis functions (eigenfunctions) and the corresponding random coefficients are orthogonal. This allows for the optimal encapsulation of the information contained in the random process into a set of discrete uncorrelated random variables. For a random field $m(x,\theta)$ with a finite variance and a mean $\bar{m}(x)$, the K-L expansion in continuous form is given as [14]:

$$m(x,\theta) = \bar{m}(x) + \sum_{i=1}^{\infty} \sqrt{\lambda_i} f_i(x) \xi_i(\theta) \tag{3.5}$$

Here, $x$ is the spatial (or temporal) variable, $\theta$ is a random event, $\xi_i(\theta)$ is a set of uncorrelated random variables, and $\lambda_i$ and $f_i(x)$ are the eigenvalues and

eigenfunctions of the covariance function $C(x_1, x_2)$ of $m(x,\theta)$. By definition $C(x_1, x_2)$ is bounded, symmetric and positive definite. Following Mercer's theorem [56], *C* has the following spectral or eigen decomposition:

$$C(x_1, x_2) = \sum_{i=1}^{\infty} \lambda_i f_i(x_1) f_i(x_2) \tag{3.6}$$

Because reservoir simulation models are defined on discrete grids, we are more interested in the discrete form of the K-L expansion. The truncated K-L expansion of the correlated (geologically constrained) random variables $\mathbf{m}$ is given as [56]:

$$[\mathbf{m}]_{H,1} = [\mathbf{\Phi}]_{H,K} [\mathbf{\Sigma}]_{K,K} [\xi]_{K,1} + [\bar{\mathbf{m}}]_{H,1} \tag{3.7}$$

Here, $\mathbf{\Phi}$ is the matrix of the eigenvectors corresponding to the *K* largest eigenvalues of the covariance matrix $\mathbf{C}_M$, $\mathbf{\Sigma}$ is a diagonal matrix consisting of the *K* largest standard deviations (square roots of eigenvalues), $\xi$ is a vector of uncorrelated standard random variables with zero mean and unit variance (dimension *K*), and $\bar{\mathbf{m}}$ is the expected value of $\mathbf{m}$. In practice, $K << H$, where *H* is the dimension of $\mathbf{m}$ (e.g., if the geology is characterized by porosity and isotropic permeability, $H=2N_C$, where $N_C$ is the total number of grid blocks in the problem). Thus $\mathbf{m}$ is represented by a much smaller set of parameters $\xi$. Using this expansion for $\mathbf{m}$, the proposed formulation of the model updating problem is as follows:

$$\min_{\boldsymbol{\xi}} \left[ S = \left\{\mathbf{m}(\xi) - \mathbf{m}_{prior}(\xi)\right\}^T \mathbf{C}_M^{-1} \left\{\mathbf{m}(\xi) - \mathbf{m}_{prior}(\xi)\right\} + \sum_{n=0}^{N-1} L^n\left(\mathbf{x}^{n+1}, \mathbf{u}^n, \mathbf{m}(\xi)\right) \right] \qquad \forall n \in (0,..,N-1)$$

subject to: (3.8)

$$g^n(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{u}^n, \mathbf{m}(\xi)) = 0 \qquad \forall n \in (0,..,N-1)$$

$$\mathbf{x}^0 = \mathbf{x}_0 \qquad \text{(Initial Condition)}$$

The problem is thus formulated with $\boldsymbol{\xi}$ as the unknown parameters. Since the components of $\boldsymbol{\xi}$ are uncorrelated, the optimization (minimization) algorithm is free to modify $\boldsymbol{\xi}$ in any manner; i.e., whatever the values of $\boldsymbol{\xi}$, the set of $\mathbf{m}$ obtained from them using Equation (3.7) will always be correlated according to the correlation structure of the covariance matrix. Thus any gradient-based algorithm can be used to accomplish the minimization using $\boldsymbol{\xi}$ as the unknowns while at the same time honoring the geological constraints for $\mathbf{m}$.

As in the optimization problem, an adjoint model is used to calculate the gradients of the objective function $S$ with respect to the parameters $\boldsymbol{\xi}$. Using the same approach as in the production optimization problem (i.e., adjoining the dynamic system to the objective function), the adjoint model is derived as:

$$\begin{aligned}
\boldsymbol{\lambda}^{Tn} &= -\left[\frac{\partial L^{n-1}}{\partial \mathbf{x}^n} + \boldsymbol{\lambda}^{T(n+1)}\frac{\partial g^n}{\partial \mathbf{x}^n}\right]\left[\frac{\partial g^{n-1}}{\partial \mathbf{x}^n}\right]^{-1} \quad \forall n = 1,...,N-1 \\
\boldsymbol{\lambda}^{TN} &= -\left[\frac{\partial L^{N-1}}{\partial \mathbf{x}^N}\right]\left[\frac{\partial g^{N-1}}{\partial \mathbf{x}^N}\right]^{-1} \qquad \text{(Final Condition)}
\end{aligned} \tag{3.9}$$

After $\boldsymbol{\lambda}$ is calculated using the adjoint model, the derivative of $S$ with respect to $\mathbf{m}$ is calculated as:

$$\frac{dS}{d\mathbf{m}} = \sum_{n=0}^{N-1}\left\{\frac{\partial L^n}{\partial \mathbf{m}} + \boldsymbol{\lambda}^{T(n+1)}\frac{\partial g^n}{\partial \mathbf{m}}\right\} + \left\{2\mathbf{C}_M^{-1}\left(\mathbf{m}-\mathbf{m}_{prior}\right)\right\}^T \tag{3.10}$$

Equation (3.10) can be used to calculate the gradient of $S$ with respect to $\boldsymbol{\xi}$ using the chain rule and Equation (3.7):

$$\frac{dS}{d\boldsymbol{\xi}} = \frac{dS}{d\mathbf{m}}\frac{d\mathbf{m}}{d\boldsymbol{\xi}} = \left[\sum_{n=0}^{N-1}\left\{\frac{\partial L^n}{\partial \mathbf{m}} + \boldsymbol{\lambda}^{T(n+1)}\frac{\partial g^n}{\partial \mathbf{m}}\right\} + \left\{2\mathbf{C}_M^{-1}\left(\mathbf{m}-\mathbf{m}_{prior}\right)\right\}^T\right]\left[\boldsymbol{\Phi}\right]\left[\boldsymbol{\Sigma}\right] \tag{3.11}$$

After the minimization is accomplished, if $\bar{\boldsymbol{\xi}}$ is the posterior mean of $\boldsymbol{\xi}$, then the posterior covariance of $\boldsymbol{\xi}$ can be approximated as [13]:

$$\bar{\mathbf{C}}_{\xi} \approx \left\{\mathbf{G}^T \mathbf{C}_D^{-1} \mathbf{G} + \mathbf{C}_{\xi}^{-1}\right\}^{-1} \tag{3.12}$$

Here the components of $\mathbf{G}$ are the partial derivatives of $S$ with respect to $\boldsymbol{\xi}$ taken at the convergence point ($\bar{\boldsymbol{\xi}}$). This can be used to determine the posterior covariance of $\mathbf{m}$, which can generally provide an accurate estimate of the dispersion of the distribution around $\bar{\mathbf{m}}$. $\bar{\mathbf{C}}_{\xi}$ is also equal to the inverse of the Hessian at convergence and is thus directly given (approximately) by any Newton type algorithm [13]. Using the posterior mean $\bar{\boldsymbol{\xi}}$ and covariance $\bar{\mathbf{C}}_{\xi}$, any number of posterior realizations of $\mathbf{m}$ can be obtained [13]. Note however, that in this simplified closed-loop, we only work with the mean realization $\bar{\mathbf{m}}$ (updated at every model updating step), and therefore $\bar{\mathbf{C}}_{\xi}$ is not required. However, in the complete closed-loop to be described in the next chapter, optimization is performed on more than one realization, implying that $\bar{\mathbf{C}}_{\xi}$ will be required.

This completes the description of the history matching procedure. Many of the main components of this adjoint, such as $\partial g^{n-1} / \partial \mathbf{x}^n$, are already calculated and stored for the production optimization problem (as described in Chapter 2), and can thus be easily reused, leading to added efficiency.

### 3.3. Implementation of the Closed-Loop

The closed-loop process (without uncertainty propagation) can be implemented very efficiently with the components discussed above. The main steps required to complete the loop are as follows:

1. From the prior model of the uncertain but correlated parameter field, calculate the covariance matrix, either numerically or analytically. Note that the covariance model may be non-stationary.

2. Perform Karhunen-Loeve expansion to determine the eigenvectors and eigenvalues of the covariance matrix. Retain only the largest eigenpairs; the number to be retained can be determined from the percentage of the total energy contained in the eigenpairs. Typically, retaining 60-70% of the total energy is sufficient.

3. Perform optimization from control step $k$ to $N_t$ (total control steps) starting with $k$ = 1, that is, the first control step, using the current maximum likelihood estimate of the parameter field $\mathbf{m}$.

4. Apply the optimized trajectory of the controls on the "true" reservoir from control step $k$ to $k+1$, and record the reservoir response for this time period. This provides the "data" to be used for history matching.

5. Perform model updating to assimilate new data. Updating can be performed from control step 1 to the step $k+1$, that is, assimilate new data and re-assimilate earlier data, or from step $k$ to $k+1$, that is, only assimilate new data.

6. Perform optimization step 3 for the next control step, that is, step $k+1$, with the new maximum likelihood estimate of the parameter field obtained from step 5. Repeat steps 3 to 6 until $k= N_t$.

Some of the above steps require further elaboration. If the prior model of the parameter field is multi-Gaussian, the covariance matrix can be calculated analytically. In general, especially if the prior model is obtained from multi-point geostatistics, the covariance matrix must be calculated numerically from a sufficiently large number of realizations of the prior model. The covariance matrix will be non-stationary if prior conditioning data such as hard data are present. The Karhunen-Loeve expansion can be

performed using singular value decomposition (SVD). However, since the standard implementation of SVD is a very expensive process, another formulation of the K-L expansion using kernels will be applied in Chapter 6 that provides a very efficient solution.

For each optimization step, the optimization process must be performed to the last control step $N_t$, even though only the trajectory from step $k$ to $k+1$ is actually applied on the "true" reservoir. This is because we are interested in long-term optimization, and future events (beyond step $k$+1) can have significant impact on the optimal trajectory from step $k$ to $k+1$. However, if the maximum likelihood estimate of the parameter field does not change much from one update to the next, the optimal trajectory obtained in the next optimization loop should not be very different from the last optimization, and therefore, techniques like neighboring optimal control [37] might be used for added efficiency.

The two methods to perform data assimilation discussed in step 5 constitute the two variants of the model updating process considered here. The traditional approach to history matching is to use all existing data at any given time to execute the updating process, even though some of that data may have been assimilated previously. This is required to maintain consistency (history match) with all existing data if the standard form of the least square error is used as the objective function (without the prior term as in Equation (3.8)).

The updating process can, however, be made much more efficient by performing the update only from step $k$ to $k+1$, that is, only assimilating new data. Consistency with previously assimilated data can be maintained approximately by using the prior term in the objective function (Equation (3.8)), with each new updating step starting with the maximum likelihood estimate from the previous step as $\mathbf{m}_{prior}$ and the posterior covariance matrix from the previous step (Equation (3.12)) as the new prior covariance $\mathbf{C}_M$. The covariance changes from one step to the next because the dynamic data

being assimilated introduces correlations into $\xi$ (independent initially by construction) after each model update. As a result, at each model updating step, a new K-L expansion must be performed on the small covariance matrix of $\xi$ (designated $\mathbf{C}_\xi$) but not on $\mathbf{C}_M$ ($\mathbf{C}_\xi$ is of dimension 20×20 in the example below, as opposed to 2025×2025, the size of $\mathbf{C}_M$). This K-L expansion of the now correlated $\xi$ then replaces the vector $\xi$ in Equation (3.7), thus giving a new set of independent $\xi$, used for updating the next control step.

Because $\mathbf{C}_\xi$ generally reduces from step to step as new data are assimilated, the $\mathbf{m}$ associated with different choices of $\xi$ will look more and more similar to the most recent $\mathbf{m}_{prior}$. Using this approach, previously assimilated data, though not assimilated directly from step $k$ to $k+1$, appear indirectly through $\mathbf{m}_{prior}$ and $\mathbf{C}_\xi$. Further, as we proceed from one control step to the next, since $\mathbf{C}_\xi$ reduces (i.e., the variance of $\xi$ reduces), the weight of the prior term in the objective function increases, implying that deviation from $\mathbf{m}_{prior}$ becomes more difficult as time proceeds. This can be helpful in alleviating problems such as those observed by [9], where late-time updates became problematic, presumably due to the assimilation of noise.

The total time required to perform one cycle of the closed-loop at control step $k$ can be quantified in terms of the total number of simulations required. One iteration of the optimization algorithm requires an equivalent of 2 simulations (from $k = k$ to $N_t$) to calculate the gradients (if constraints are implemented internally) and 1-4 simulations (from $k = k$ to $N_t$) to calculate the step size in the search direction (using sequential quadratic programming). Typically 4-6 iterations result in substantial improvements in the objective function. Similarly, for the model updating component, using the first approach, an equivalent of 2 simulations (from $k = 1$ to $k$) is required to calculate the gradients and 1-4 simulations (from $k = 1$ to $k$) to calculate the step size in the search direction. Using the second approach for model updating, the simulation length is

reduced to one control step. Usually about 5 to 10 iterations result in convergence. Thus, if we update the reservoir model and controls 10 times over the course of the simulated production, the equivalent of about 300 (120 for optimization + 180 for model updating) complete simulations (from $k$=0 to $N_t$) would be required under the first approach, compared to about 155 (120 for optimization + 35 for model updating) complete simulations using the second approach.

### 3.4. Case Study – Dynamic Waterflooding

The closed-loop approach discussed above is now applied to an idealized example case somewhat similar to that used by Brouwer et al. [9], which was also discussed in the last chapter. This case was chosen primarily because it effectively demonstrates the applicability of adjoint-based optimization to smart well control and because it illustrates that the model updating approach can be successfully used with realizations based on multipoint (as opposed to two-point) geostatistics. In addition, this example allows us to qualitatively compare our model updating approach to Kalman filters as used by Brouwer et al. [9].

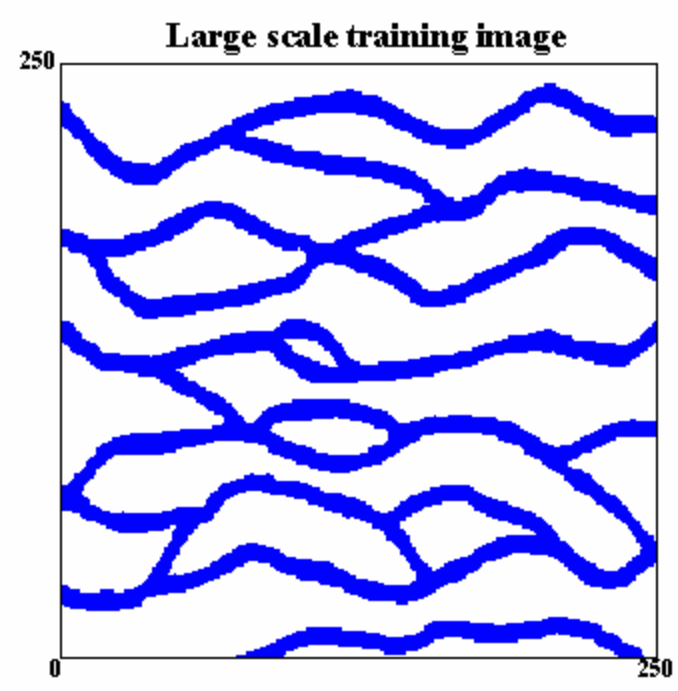


Figure 3-1 Training image used to create the original realizations (from Strebelle [57])

The schematic of the reservoir and well configuration is shown in Figure 2-3. The model, as explained in the last chapter, consists of one horizontal "smart" water injector and one horizontal "smart" producer, each having 45 controllable segments. The reservoir covers an area of $450 \times 450$ $m^2$ and has a thickness of 10 m and is modeled by a $45 \times 45 \times 1$ horizontal 2D grid. The fluid system is essentially an incompressible two-phase unit mobility oil-water system, with zero connate water saturation and zero residual oil saturation.

In order to apply the closed-loop approach, one or more of the reservoir properties must be unknown. For this example, permeability is assumed to be unknown and will be updated by assimilating production data. Further, it is assumed that we have some prior knowledge of the reservoir, which informs us that the reservoir is a fluvial channelized reservoir as depicted by the training image [57] shown in Figure 3-1, with the channel sand permeability being about 10 Darcy and the background sand permeability about 500 mD. The contrast in permeability between the high permeability sand and the background reservoir is about a factor of 20, and it is this heterogeneity that makes the optimization results interesting.

Figure 3-2 shows some unconditioned realizations of the permeability field generated using the *snesim* software [57] with the training image of Figure 3-1. In order to validate our closed-loop approach, a "true" realization is required, against which the optimization and model updating results can be compared. Realization 9 from Figure 3-2 is arbitrarily taken to be the true realization. Note that although we are using unconditioned realizations for this example, realizations conditioned to hard data can be used just as easily with this approach. In the absence of any other data, all realizations obtained from *snesim* are equiprobable, thus any realization could be chosen as an initial guess. We choose realization 8 to be the initial guess. Note that the connectivity and the location of the channels are quite different in these two realizations, and therefore the nature of the production data would also be very different, particularly in terms of important features like breakthrough times.

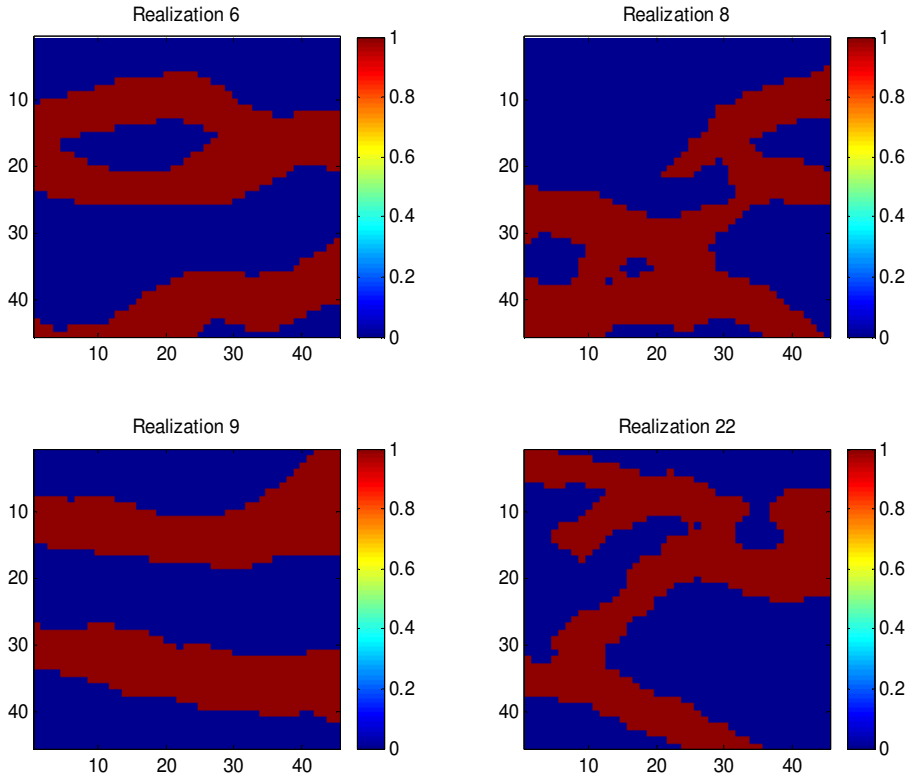


Figure 3-2 Some of the realizations created with *snesim* [57]; realization 9 is assumed to be the "true" realization for case 1 and realization 8 is the initial guess

Karhunen-Loeve expansion is performed using the covariance matrix created from 1000 initial realizations. The energy retained in the eigenpairs is plotted in Figure 3-3. We observe that most of the energy is associated with the first few eigenpairs. Figure 3-4 and Figure 3-5 show the reconstructions of the true and initial permeability fields using 20 eigenpairs. It is clear that although the long correlation structures (low frequencies) are essentially preserved, the smaller correlation structures (high frequencies) are lost, which results in a smoothing effect. Although this smoothing results in an approximation of the actual multipoint geostatistics, it is beneficial in that it provides smoother gradients, which in turn leads to better convergence behavior of the minimization algorithm. For the purpose of model updating, we chose to retain only 20 eigenpairs; this corresponds to about 65% of the total energy.

For purposes of optimization, the injector segments are placed under rate control, and the producer segments are under BHP control. There is a total injection constraint of 2700 STBD; thus the optimization essentially results in a redistribution of this water

among the injection segments. There are also bounds on the minimum and maximum rates allowed per segment, as well as bounds on the BHPs of the producers, which could for example correspond to bubble point pressures or fracture pressures. The model is produced until exactly one pore volume of water is injected, which corresponds to around 950 days of injection. This time period is divided into seven control steps of 30, 60, 100, 190, 190, 190 and 190 days. Thus the total number of controls is equal to (45 + 45)×7 = 630. All constraints in this problem are linear with respect to the controls. These seven control steps also correspond to the model updating steps. The injection BHPs and producer water and oil rates from the "true" model are used as data to update the permeability field. Since these measurements are synthetic, they are noise-free, though in reality these measurements would also contain noise, which can be accounted for using $\mathbf{C}_D$.

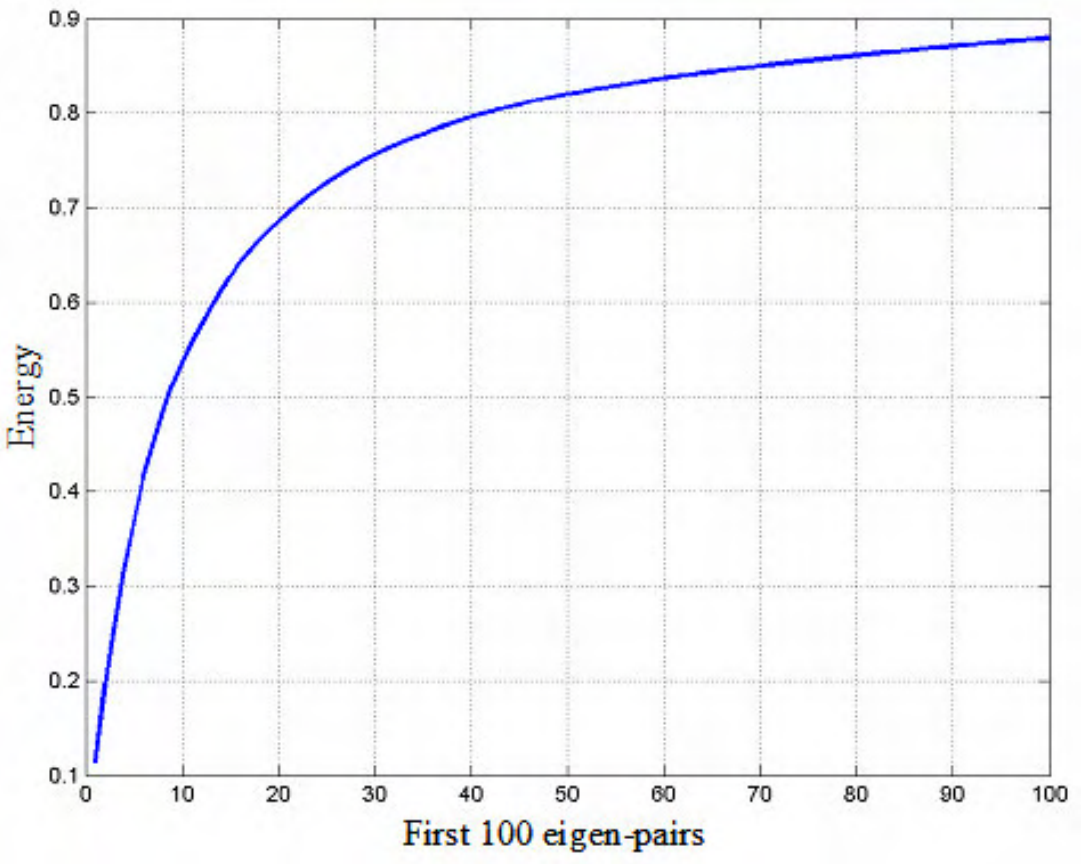


Figure 3-3 Energy retained in the first 100 eigenpairs

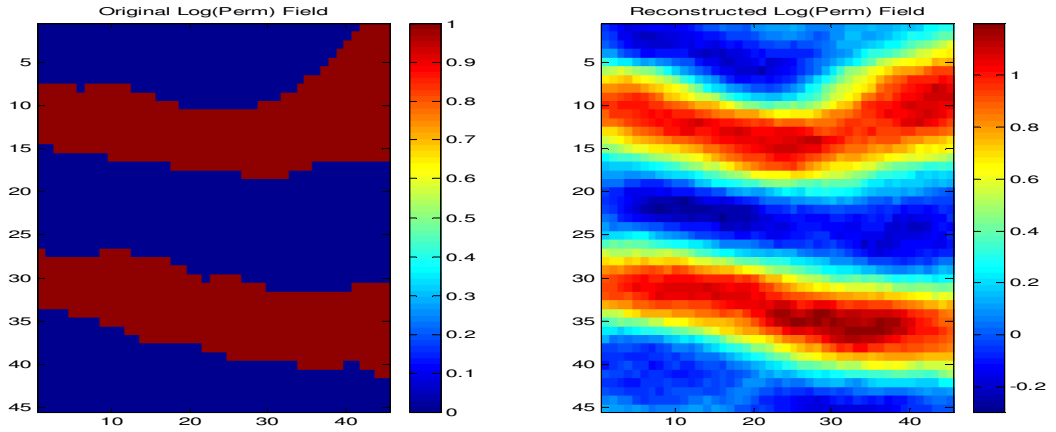


Figure 3-4 Reconstruction of "true" realization with 20 eigenpairs

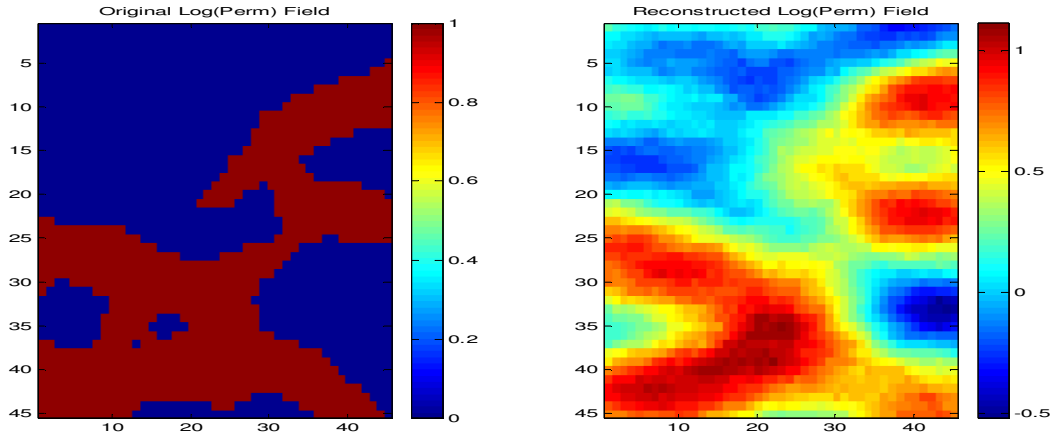


Figure 3-5 Reconstruction of initial realization with 20 eigenpairs

In order to understand the benefit of any optimization process, it is usual to compare the optimization results against a base or reference case. Here, the base case is a constant rate/constant BHP production strategy. The 2700 STBD of injection water is distributed among the 45 injection segments according to their *kh*, which corresponds to an uncontrolled case. The producer BHPs are set in such a way that a balanced injection-production is obtained, meaning that total liquid injection is equal to total liquid production. The objective of the optimization process is to maximize NPV, defined by Equation (2.22). Also, as in Chapter 2, the NPV discounting factor $\alpha$ is set to zero. The oil price is conservatively set at \$80/m$^3$, water injection cost at \$0/m$^3$, and water production cost at \$20/m$^3$.

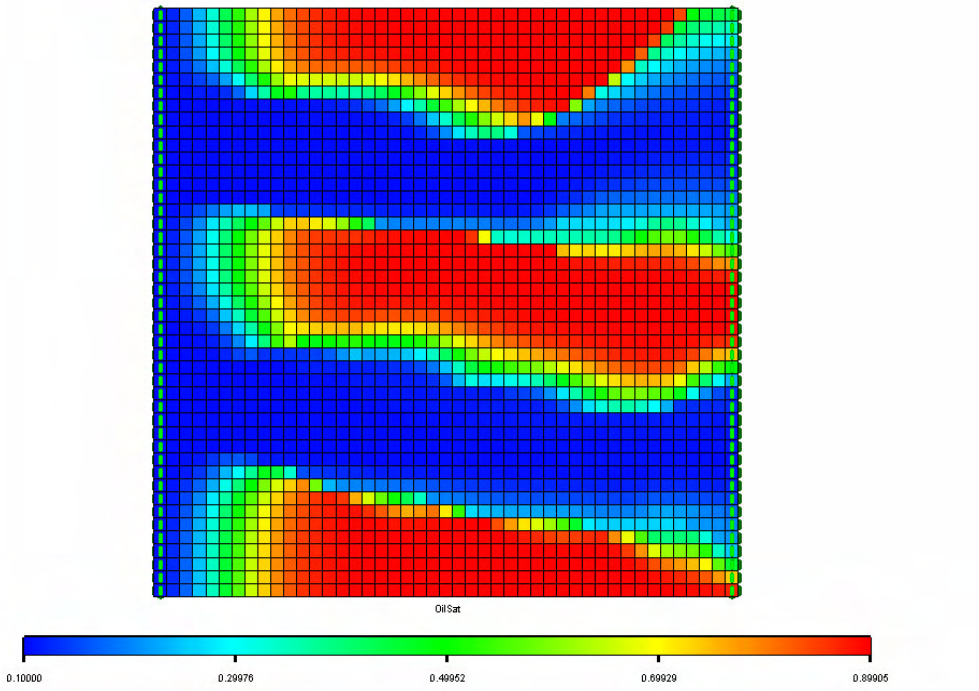


Figure 3-6 Final oil saturations after 1 PV injection for reference case with “true” realization

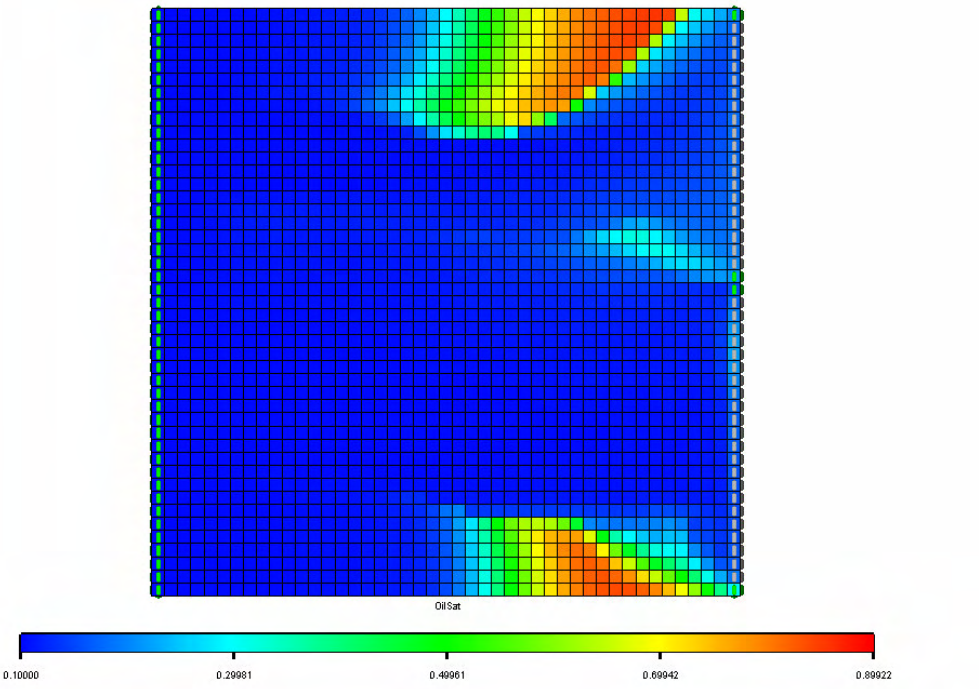


Figure 3-7 Final oil saturations after 1 PV injection for optimization with “true” realization

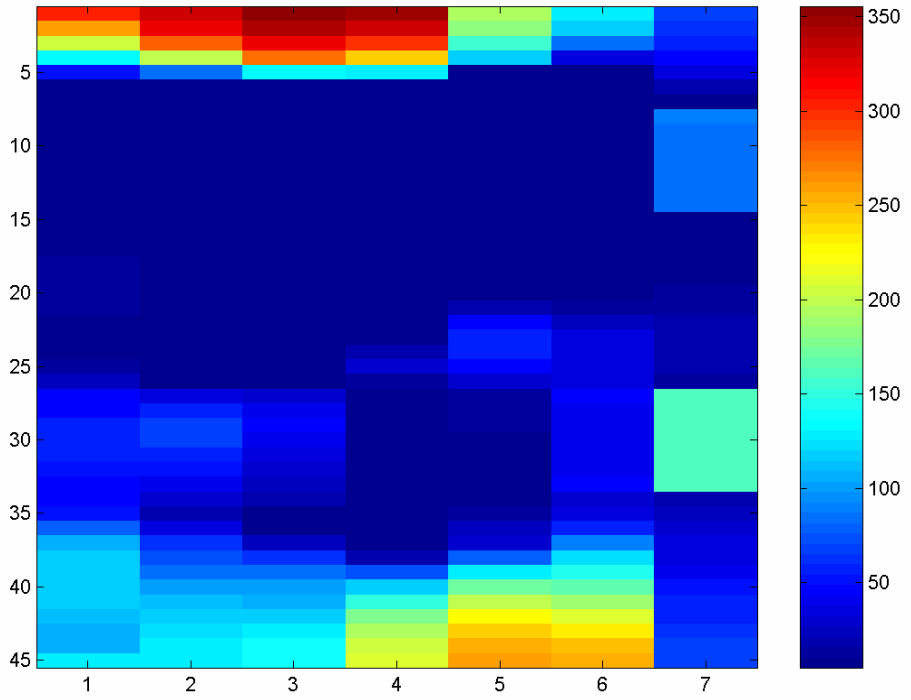


Figure 3-8 Injection rate variation with time for optimization with "true" realization (*x*-axis represents the control steps, and *y*-axis represents the 45 injector segments)

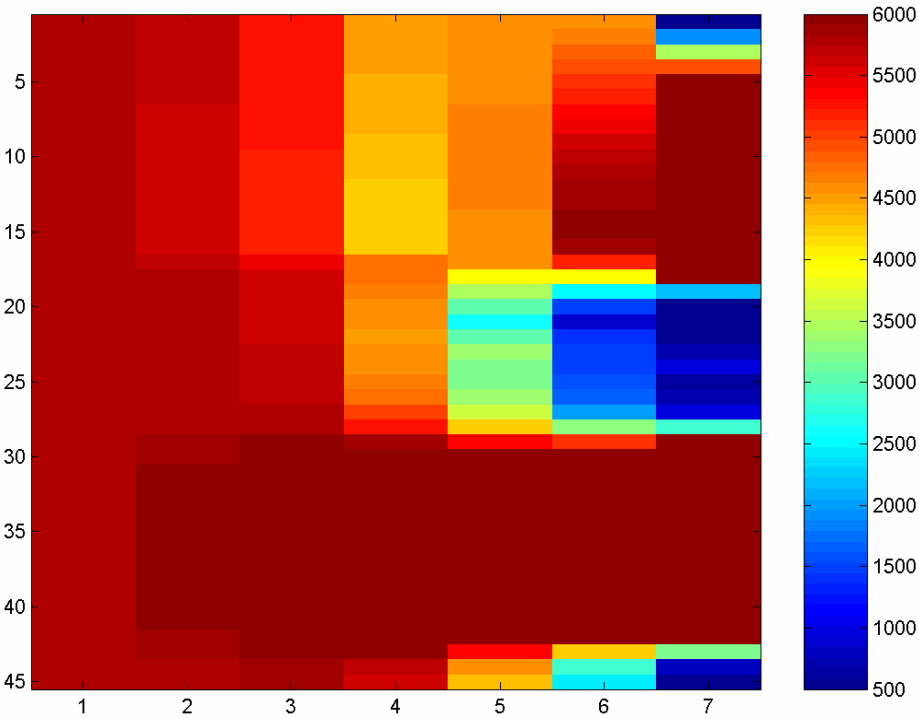


Figure 3-9 Producer BHP variation with time for optimization with "true" realization (*x*-axis represents the control steps, and *y*-axis represents the 45 producer segments)

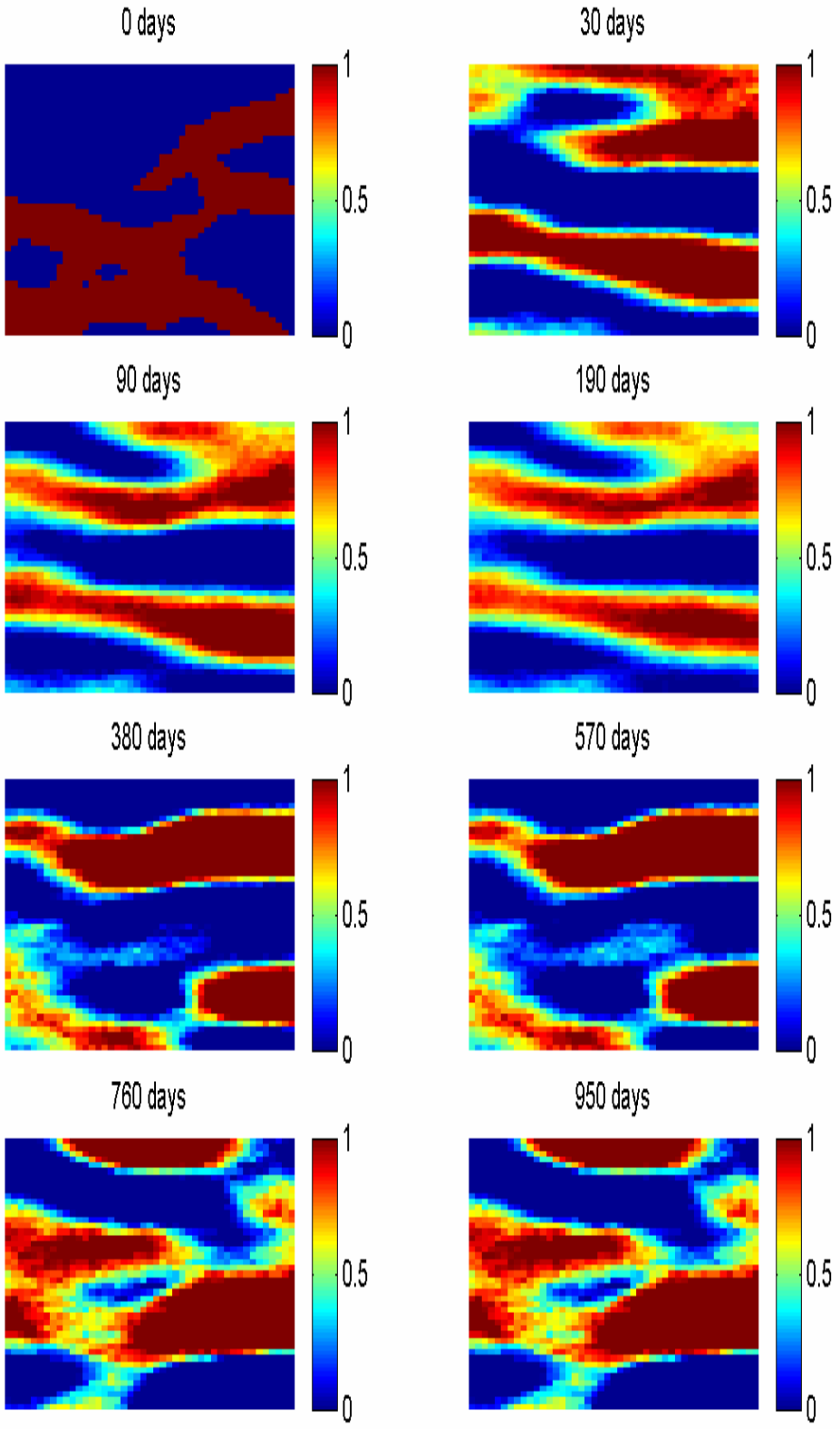


Figure 3-10 Permeability field updates with model updating performed using all available data and without prior term in objective

The optimization process is first demonstrated assuming that the "true" permeability field is known, and to realize the benefit of the process, it is compared against the reference case, also evaluated using the "true" permeability. This constitutes what is known as an "open loop" optimization. Starting from 100% oil saturation throughout the reservoir, Figure 3-6 and Figure 3-7 show the final oil saturations for the uncontrolled and the optimized case after exactly 1 PV of water has been injected. It is clear that the optimization leads to a substantial improvement in the sweep efficiency, leading to the increase in NPV of almost 100%.

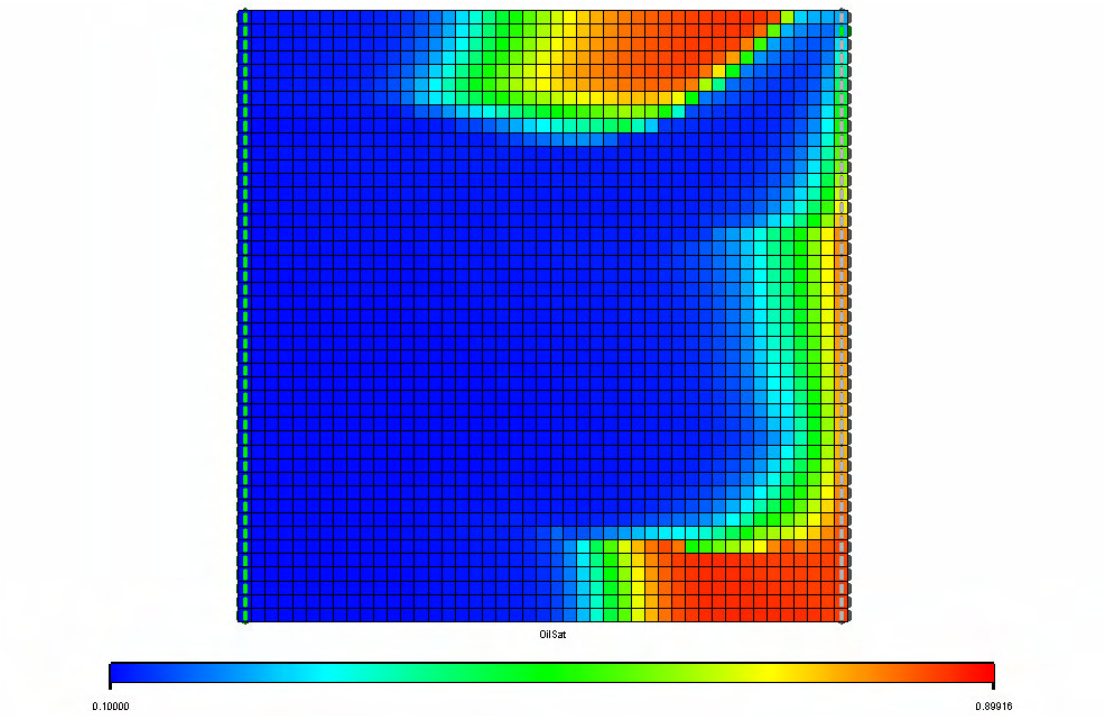


Figure 3-11 Final oil saturations after 1 PV injection for optimization with model updating using first approach

Figure 3-8 and Figure 3-9 show the optimized trajectories of the controls – rates of the injectors and BHPs of the producers – that provide an insight as to why a better sweep is obtained after optimization. The *y*-axis of Figure 3-8 corresponds to 45 injector segments and the *x*-axis corresponds to the 7 control steps. The color scale corresponds to injection rates of the segments, with blue being lowest rates (almost closed) and brown being highest (fully open). As in the deterministic optimization of Chapter 2, the injector segments completed in or near the high permeability streaks are

nearly shut for most of the time, resulting in early breakthrough and thus poor sweep. Again, the injector segments at the edges of the reservoir are almost fully open, which forces the water front away from the high permeability streaks to move laterally, resulting in the almost 100% sweep of these regions. In Figure 3-9, the color scale corresponds to the BHPs of the producer segments, and we again observe that the producer segments completed in or near the high permeability streaks are shut most of the time. Also, the producer segments between the streaks open more towards the later control steps, thus moving the water present in the streaks toward this region, which in turn leads to a better sweep.

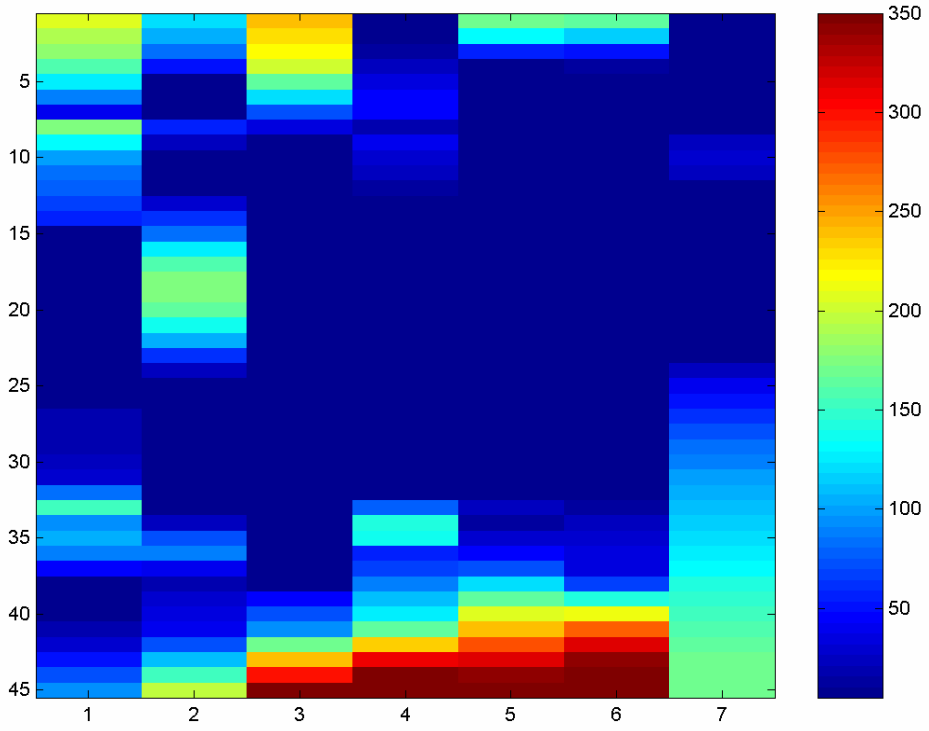


Figure 3-12 Injection rate variation with time for optimization with model updating using first approach (*x*-axis represents the control steps, and *y*-axis represents the 45 injector segments)

The open loop approach discussed above required that the permeability field (and other reservoir properties) be known completely. However, in reality, we never have complete knowledge of the reservoir, and thus the closed-loop approach must be used. The results of the open loop approach can usually be thought of as the best possible results that can be achieved by any closed-loop approach, of course under the

assumption that the same search algorithms and the same initial starting point for the controls is used. It can thus be used as a benchmark against which closed-loop algorithms can be compared. In the following set of results, we apply the closed-loop procedure.

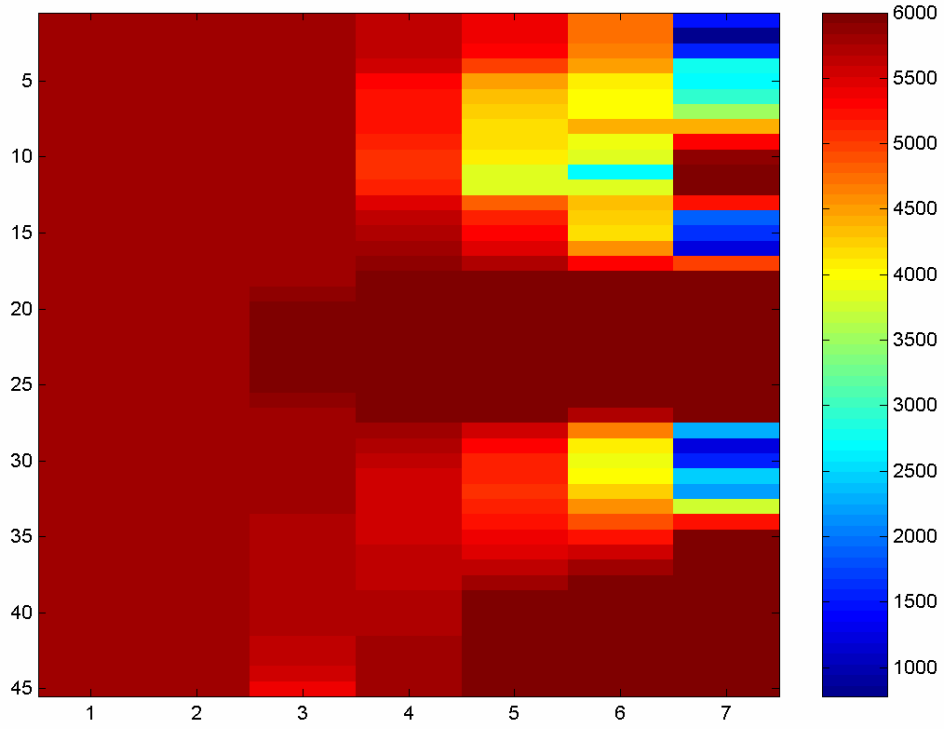


Figure 3-13 Producer BHP variation with time for optimization with model updating using first approach (*x*-axis represents the control steps, and *y*-axis represents the 45 producer segments)

Figure 3-10 shows the permeability field updates obtained using the first model updating approach, but without the prior term in the objective function. Comparing to the "true" realization (realization 9 in Figure 3-2), it is clear that the correct channel locations and connectivity are well approximated even after the first update at 30 days. However, the model appears to deteriorate somewhat at later times, and this may be due to over-fitting, which is possible due to the absence of the prior term. To elaborate, the prior term indirectly takes into account previously assimilated data by not allowing the solution to move too far from the previous prior estimate according to the weight of the prior term. Since the previous prior estimate was a result of assimilation of previous data, the amount of data "available" for the history match is

more (indirectly) when the prior term is present, although the number of parameters $\xi$ is the same for both cases.

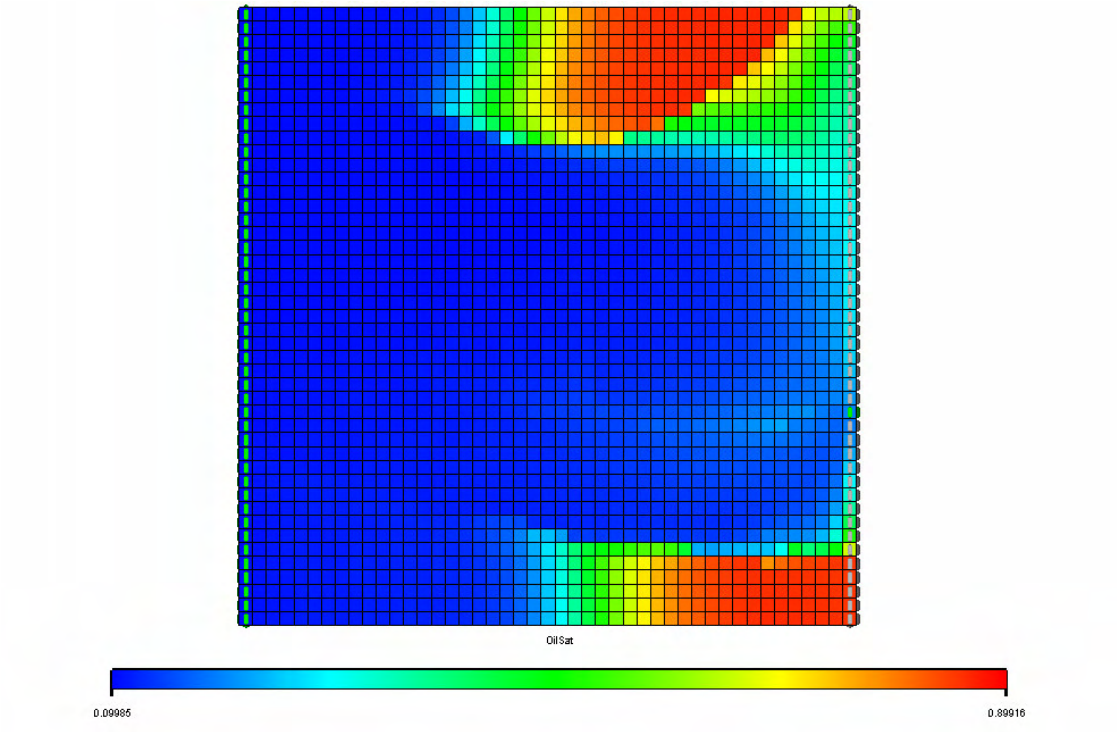


Figure 3-14 Final oil saturations after 1 PV injection for optimization with model updating using second approach

Figure 3-11 shows the final oil saturations obtained using this closed-loop approach. Comparing this to Figure 3-6 and Figure 3-7, we see that the sweep obtained is greatly improved over the uncontrolled case, and is almost as good as that using the open loop approach. Comparing Figure 3-12 and Figure 3-13 to Figure 3-8 and Figure 3-9, it is clear that the optimal control trajectories obtained are reasonably similar with the open and closed-loop approaches, which is why the sweeps are comparable.

Figure 3-14 shows the final oil saturations obtained using the second variant of the closed-loop approach. The sweep obtained is even better than the first variant and is very close to that obtained with the open loop approach. This improvement is due to improved permeability updates. This is evident through comparison of the permeability updates using the second variant (Figure 3-15) with those using the first variant (Figure 3-10). Thus the second approach to model updating is not only more

efficient, but seems to provide better model updates by preventing over-fitting. The control trajectories (Figure 3-16 and Figure 3-17) again resemble those using the open loop approach.

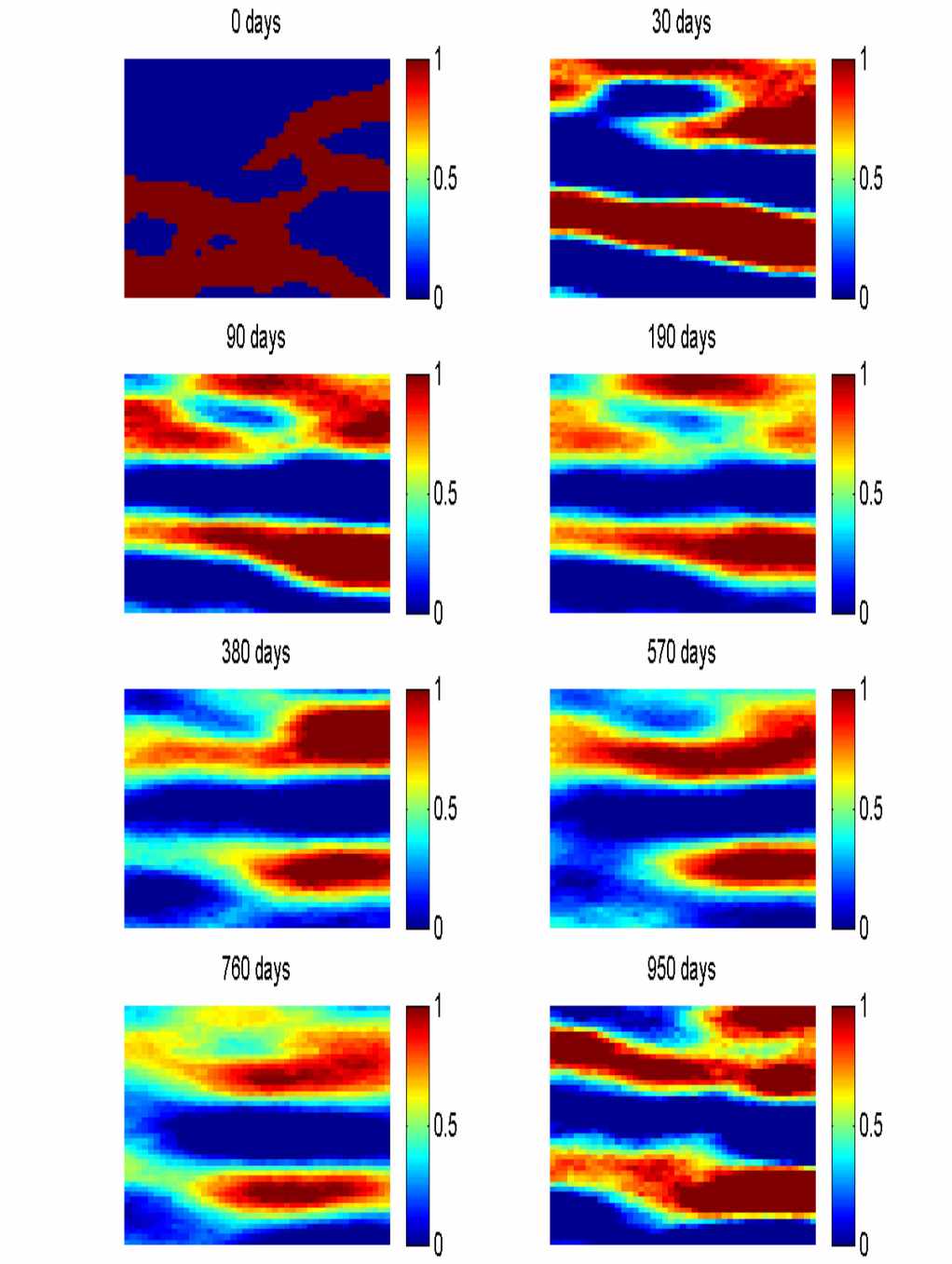


Figure 3-15 Permeability field updates with model updating performed by assimilating data only over last control step and with prior term in objective

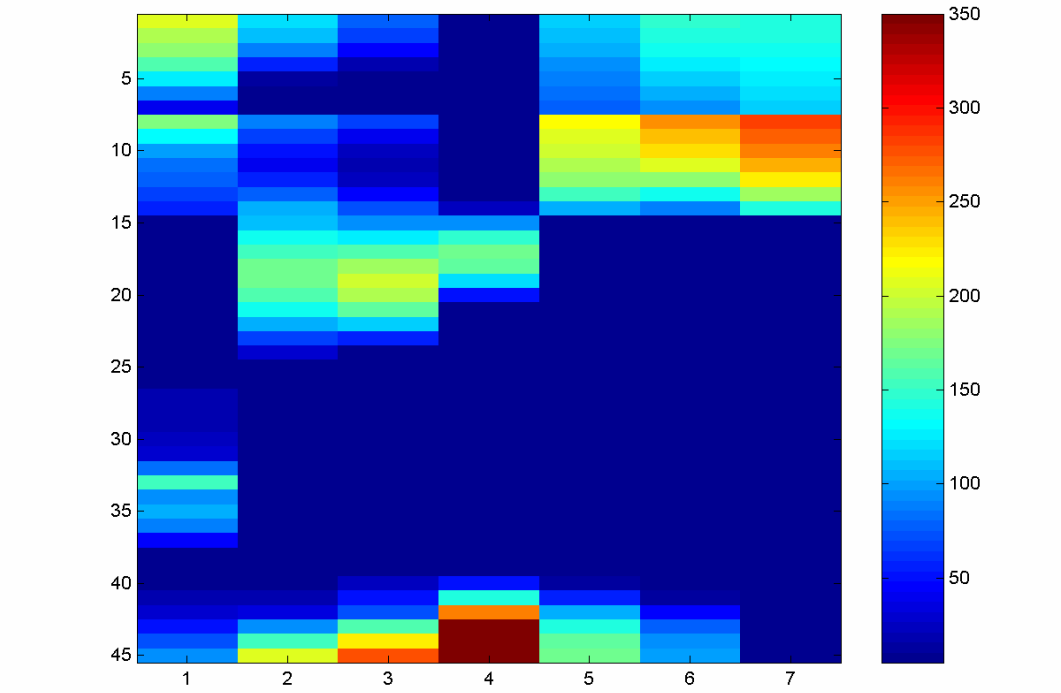


Figure 3-16 Injection rate variation with time for optimization with model updating using second approach (*x*-axis represents the control steps, and *y*-axis represents the 45 injector segments)

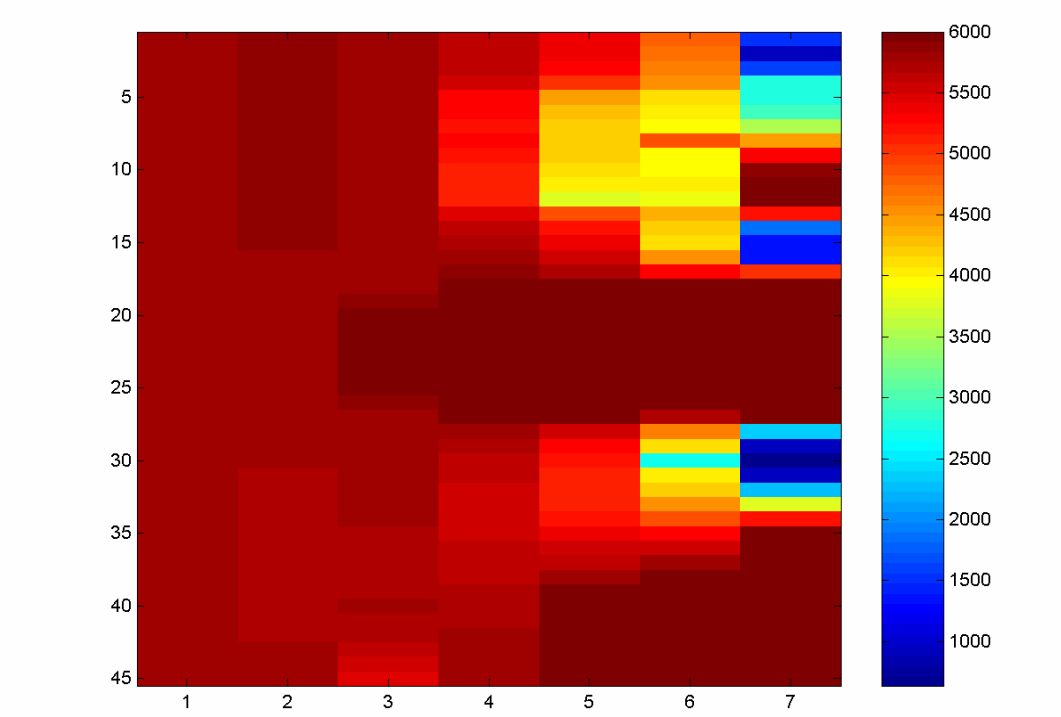


Figure 3-17 Producer BHP variation with time for optimization with model updating using second approach (*x*-axis represents the control steps, and *y*-axis represents the 45 producer segments)

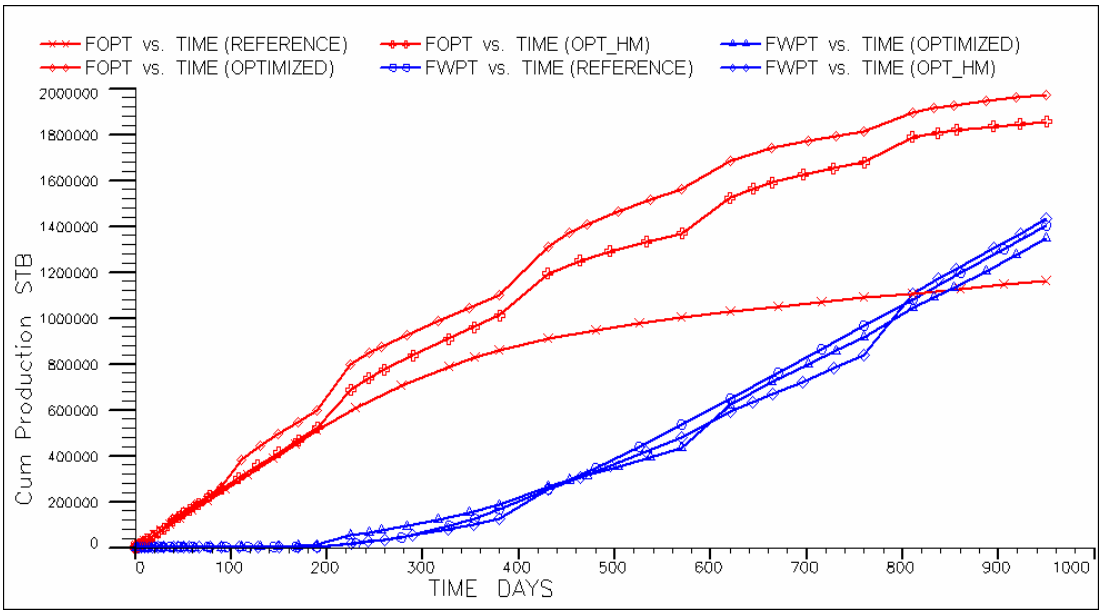


Figure 3-18 Comparison of cumulative production for reference case, optimized case run with "true" realization, and closed-loop approach

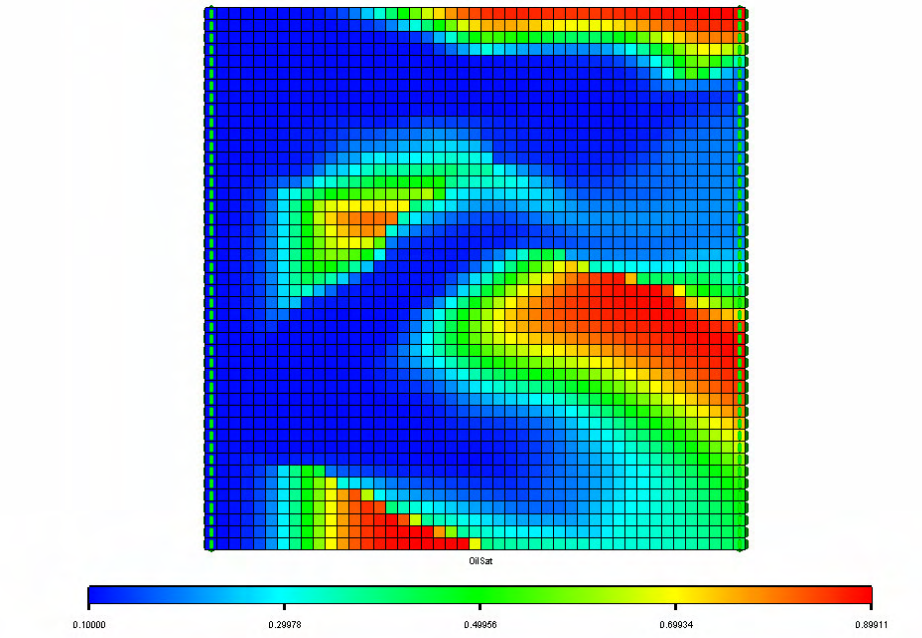


Figure 3-19 Final oil saturation for uncontrolled reference case (case 2)

Since the reservoir model and the "true" permeability field are similar to that used by Brouwer et al. [9], a qualitative comparison can be made between the model updating approach followed here and their use of Kalman filters for the same purpose. Comparing Figure 3-15 with Figure 4 of [9], it is obvious that the channel structure is much more visible with the present approach. However, this is more likely due to the

use of a better prior model (channel training image compared to a Gaussian prior) than to any underlying advantage of the method itself. In any case, given that the objective at hand is to determine the posterior uncertainty of the parameters only, minimization with adjoints seems to be more efficient, as the number of simulations required for convergence is usually around 20-30, whereas with the ensemble Kalman filter, about 100-200 forward simulations are required [9, 44]. However, the two methods are not exactly equivalent, as the Kalman filter can also provide the uncertainty in the states.

Figure 3-18 shows that there is a substantial increase in cumulative oil production with both the open loop (70%) and closed-loop (60%) approaches, whereas water production is not much affected. This is directly attributable to improved sweep. The increase in NPV is about 100% for the open loop and 85% for the closed-loop.

The second variant of the closed-loop approach is now applied to another, more geologically complex, example (realization 22, shown in Figure 3-2). Compared to realization 9, this model is more sinuous and the channels are also connected laterally. In this case the initial model is again taken to be realization 8. The final oil saturations for the uncontrolled case, the open loop and the closed-loop are shown in Figure 3-19, Figure 3-20 and Figure 3-21. The evolution of the model is shown in Figure 3-22. Due to the better connectivity of the channels, the final sweep even in the uncontrolled case is considerable. Figure 3-23 compares the cumulative oil and cumulative water produced. The increase in oil production is around 25% for both the open and closed-loop cases, but there is also an increase in water production by around 30%. The increase in NPV is approximately 25% for both cases.

It is interesting to note that although the model updates as seen in Figure 3-22 are not as accurate as in the previous case, the closed-loop results are quite close to those obtained by the open loop approach. This seems to suggest that even very approximate permeability updates may result in nearly correct optimal trajectories, resulting in improvements of the objective similar to those obtained using an open loop. The poorer permeability updates can be understood by observing that only very small

regions in the training image have similar sinuosity and connectivity as realization 22, implying that realizations similar to realization 22 are relatively rare in the parameter space. This is in contrast to realization 9, which resembles the training image and other realizations more closely, and was as a result better matched during the course of the permeability updates.

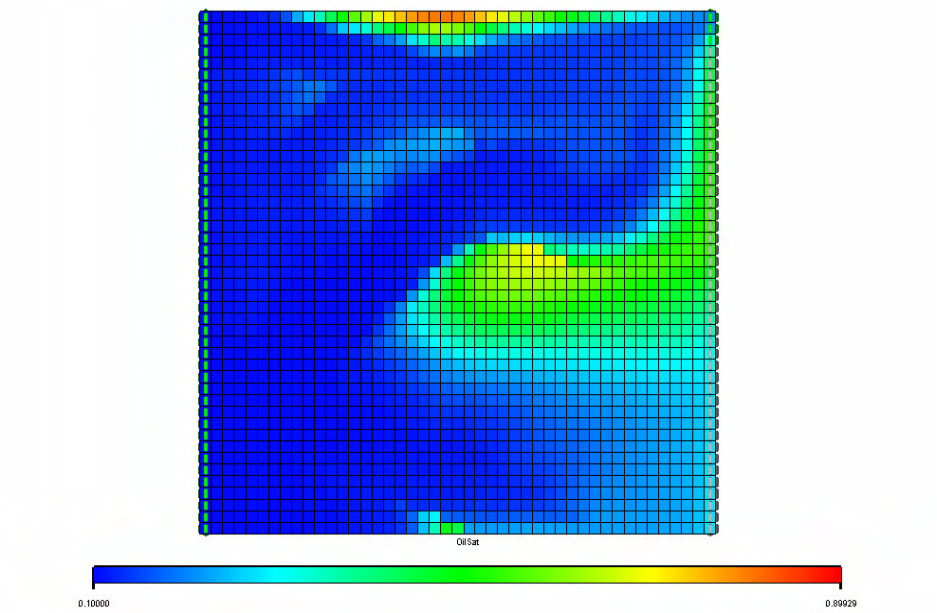


Figure 3-20 Final oil saturation for optimization on “true” realization (case 2)

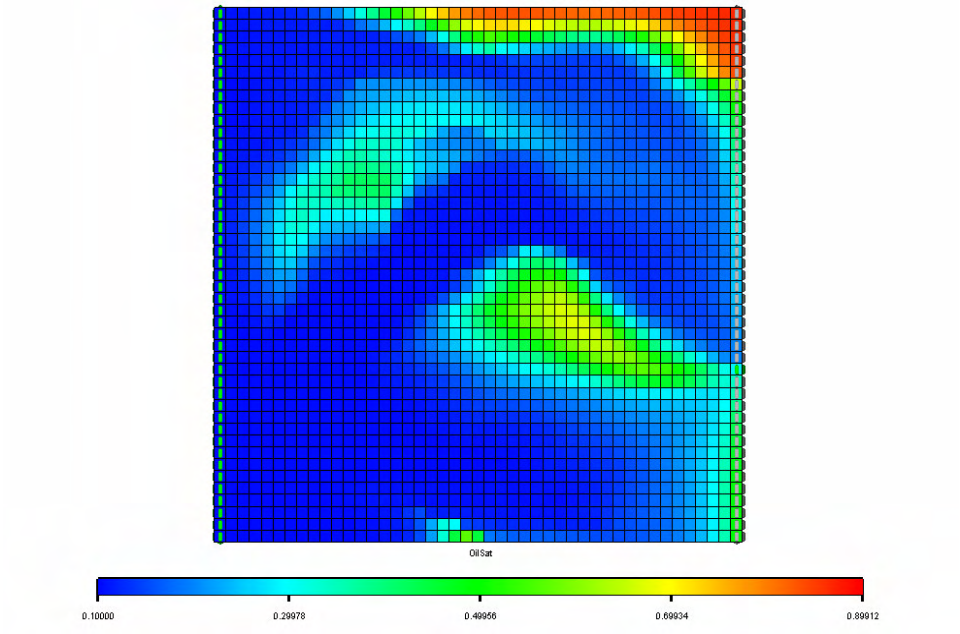


Figure 3-21 Final oil saturation for optimization with model updating (case 2)

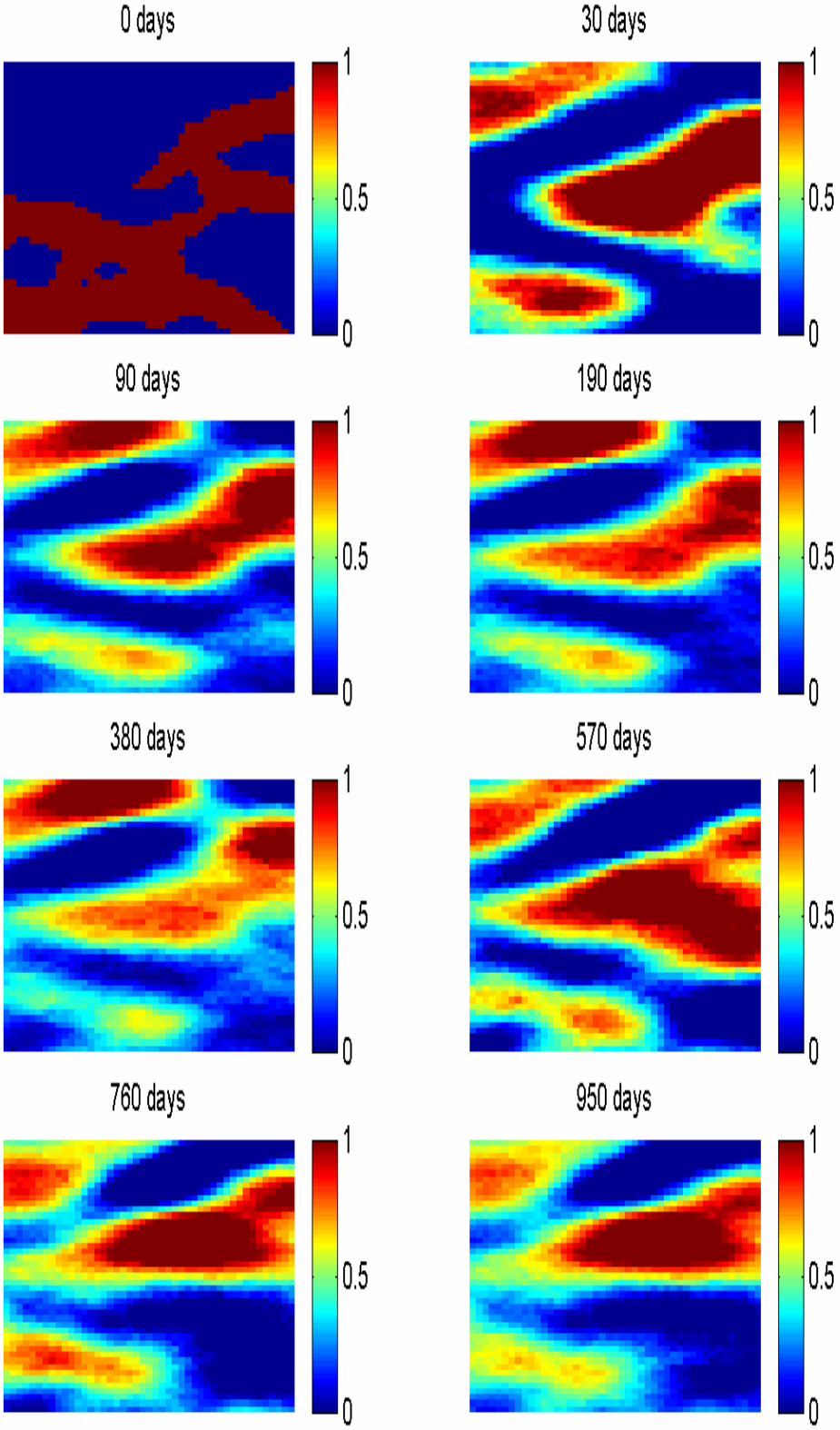


Figure 3-22 Permeability field updates with model updating performed by assimilating data only over last control step and with prior term in objective (case 2)

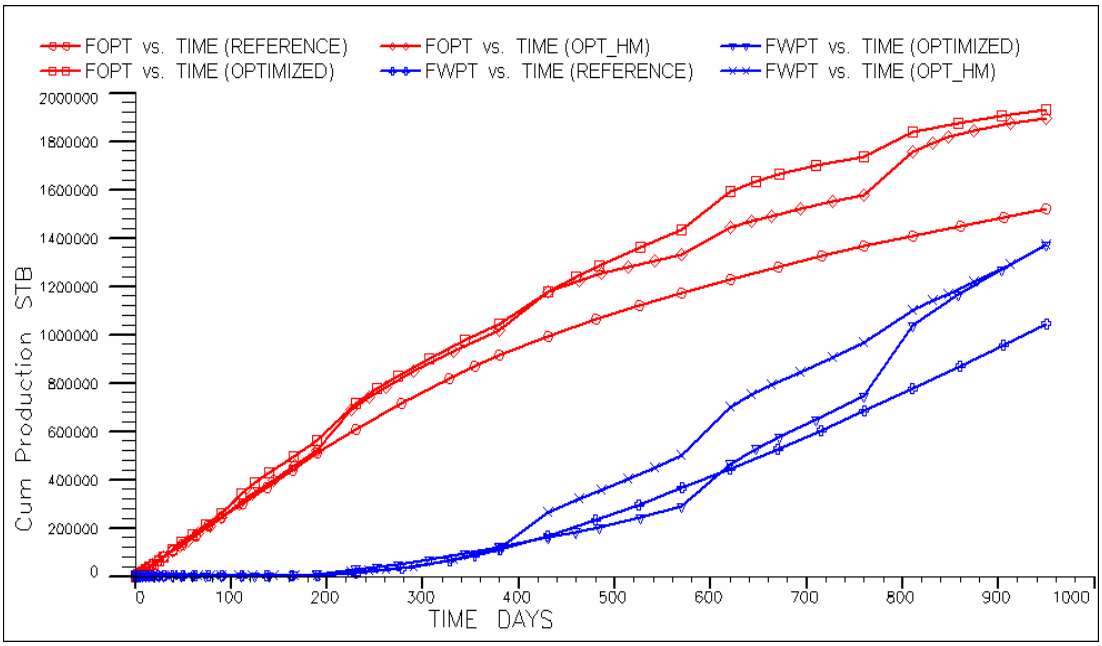


Figure 3-23 Comparison of cumulative production for reference case, optimized case run with "true" realization, and closed-loop results (case 2)

## 3.5. Summary

In this chapter, the use of adjoint-based models was shown to be very efficient for closed-loop optimal control, and significant improvements in recovery appear to be possible through the application of such techniques. In particular, the following may be concluded:

1. Efficient parameterization of the uncertain reservoir properties in terms of the K-L expansion, combined with Bayesian inversion and adjoint models, provides an efficient algorithm for model updating under geological constraints.

2. The increase in NPV and sweep efficiency by the closed-loop approach is very close to that obtained using an open loop approach for both of the examples studied. In addition, the results from both examples indicate that approximate parameter fields may be adequate for obtaining near optimal trajectories of the controls.

3. Assimilating only the new production data at any control step (coupled with the use of the prior term in the history matching objective function) is much more efficient, and may be as effective as assimilating all existing data.

4. Since adjoint models are used both for optimization and model updating, many components of the code can be reused, resulting in added efficiency.

An important issue that has not been investigated here is the propagation of uncertainty and its effect on the optimal trajectories. Although for the relatively simple cases studied here uncertainty propagation may not be necessary, it can be important for more complex geological scenarios (as demonstrated in [45]). This issue is addressed in the next chapter. In addition, although the K-L representation performed well in the cases considered here, this representation may not always be adequate. This issue is considered in Chapter 6.

# Chapter 4

# 4. Efficient Closed-loop Production Optimization

In this chapter, we extend the simplified closed-loop described in the last chapter to also include uncertainty propagation, thereby obtaining the complete closed-loop algorithm as proposed in Chapter 1. The optimization and model updating components of the closed-loop have already been explained in the last two chapters, and this chapter therefore will focus on uncertainty propagation and its integration within the closed-loop. Uncertainty propagation refers to determining the probability distribution of the output parameters of a mathematical model (in our case, the simulation model), given the probability distributions of its input parameters. In this regard, we apply polynomial chaos expansions (within the probabilistic collocation method [19]) together with the Karhunen-Loeve expansion for efficient uncertainty propagation. Polynomial chaos expansions are bi-orthogonal spectral expansions of random fields, and they allow optimal encapsulation of information contained in the input random fields and output random variables of any mathematical model, thereby generating accurate "polynomial chaos proxies" with relatively few evaluations of the mathematical model.

As summarized by Xiu et al. [17], traditional approaches such as Monte Carlo simulation and its variants (e.g., Latin Hypercube sampling), although simple to implement, are in general computationally very expensive and may be infeasible for practical reservoir simulation models. The sensitivity method [17] is a more economical approach, based on the moments of samples, but it is less robust and depends strongly on the modeling assumptions. Another popular technique is the perturbation method [17] where all the stochastic quantities are expanded around their mean via Taylor series. This approach, however, is generally limited to small

perturbations and does not readily provide information on high-order statistics of the response, and also requires access to model equations. The resulting system of equations becomes extremely complicated beyond second-order expansion. Another approach is based on expanding the inverse of the stochastic operator in a Neumann series [17], but this too is limited to small fluctuations, and even combinations with the Monte Carlo method seem to result in computationally prohibitive algorithms for complex systems. Two more recent analytical and computationally efficient methods that also use polynomial chaos expansions are the spectral stochastic finite element method [17, 58, 59] and its combination with perturbation methods [60, 61]. These methods, however, also require access to the equations of the mathematical model (i.e., a "black box" approach is not possible). Some of these methods are discussed in detail by Isukapalli [18]. The approach we follow in this work is known as the probabilistic collocation method (PCM) [19, 62] and also the stochastic response surface method (SRSM) [18] and is based on the spectral expansion of the random variables and fields by polynomial chaos expansions and the Karhunen-Loeve expansion. This approach is usually one or more orders of magnitude faster than Monte Carlo techniques, and has similar computational complexity as the Spectral Stochastic Finite Element method [17], but has the advantage that the forward model is treated as a black box, implying that implementation is relatively straightforward. The probabilistic collocation method has been used successfully in other fields for uncertainty propagation [19] but has not been used in reservoir modeling, although polynomial chaos expansions have been applied recently [60].

Since the basis behind the PCM/SRSM is the polynomial chaos expansion (PCE), the general theory behind PCE is discussed in the next section. This is followed by a detailed explanation of the standard PCM/SRSM algorithm. Next, the application of PCM/SRSM on a quarter five-spot pattern with a lognormal random permeability field with exponential covariance is demonstrated. Finally, the steps necessary to combine the three elements (optimization, model updating and uncertainty propagation) to complete the closed-loop are discussed. The overall methodology is then successfully

applied to a realtime closed-loop dynamic waterflood optimization of a synthetic reservoir with an uncertain permeability field.

### 4.1. Polynomial Chaos Expansions

Polynomial chaos expansions, as introduced by Weiner (see discussion in Lucor et al. [58]), provide a means for expanding second-order random processes in terms of Hermite polynomials of Gaussian random variables (note that the term "chaos" as used in this context is completely distinct from the chaos that appears in solutions of nonlinear differential equations). Second-order random processes are processes with finite variance, and this applies to most physical processes. Thus, a general second-order random variable $X(\theta)$, viewed as a function of $\theta$ (the independent random event), can be represented in the following form:

$$\begin{aligned} X(\theta) = a_0 H_0 + \sum_{i_1=1}^{\infty} a_{i_1} H_1\left(\xi_{i_1}(\theta)\right) + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1} a_{i_1 i_2} H_2\left(\xi_{i_1}(\theta), \xi_{i_2}(\theta)\right) \\ + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} a_{i_1 i_2 i_3} H_3\left(\xi_{i_1}(\theta), \xi_{i_2}(\theta), \xi_{i_3}(\theta)\right) + \ldots \end{aligned} \tag{4.1}$$

Here, $H_n\left(\xi_{i_1},.....,\xi_{i_n}\right)$ denotes the Hermite polynomials of order *n* in terms of the multi-dimensional independent standard Gaussian random variables $\left(\xi_{i_1},.....,\xi_{i_n}\right)$ with zero mean and unit variance. The above equation is the discrete version of the original Wiener polynomial chaos expansion [58], where the continuous integrals are replaced by summations. The general expression of the Hermite polynomials $H_n$ is given by [58]:

$$H_n\left(\xi_{i_1},.....,\xi_{i_n}\right) = \exp\left(\frac{1}{2}\xi^T\xi\right)(-1)^n \frac{\partial^n}{\partial\xi_{i_1}....\partial\xi_{in}} \exp\left(-\frac{1}{2}\xi^T\xi\right) \tag{4.2}$$

Here, $\xi$ denotes the vector of *n* Gaussian random variables. The above expression can be reduced to simpler forms. For example, the 1D Hermite polynomials can be written as:

$$\psi_0 = 1, \quad \psi_1 = \xi, \quad \psi_2 = \xi^2 - 1, \quad \psi_3 = \xi^3 - 3\xi, .... \tag{4.3}$$

For notational convenience, Equation (4.1) can be rewritten as:

$$X(\theta) = \sum_{j=0}^{\infty} \hat{a}_j \psi_j(\xi) \tag{4.4}$$

There is a one-to-one correspondence between the functions $H_n\left(\xi_{i_1}, ...., \xi_{i_n}\right)$ and $\psi_j(\xi)$, and also between the coefficients. In Equation (4.1) the summation is carried out according to the order of the Hermite polynomials, while in Equation (4.4) it is simply a re-numbering with the polynomials of lower order counted first. The polynomial chaos forms a complete orthogonal basis in the $L_2$ space of the Gaussian random variables $\xi$, i.e.,

$$\left\langle \psi_i \psi_j \right\rangle = \left\langle \psi_i^2 \right\rangle \delta_{ij} \tag{4.5}$$

Here $\delta_{ij}$ is the Kronecker delta and $\left\langle \psi_i \psi_j \right\rangle$ denotes the ensemble average. This is the inner product in the Hilbert space of the Gaussian random variables:

$$\left\langle f(\xi) g(\xi) \right\rangle = \int f(\xi) g(\xi) W(\xi) d\xi \tag{4.6}$$

The weighting function $W(\xi)$ is given as:

$$W(\xi) = \frac{1}{\sqrt{(2\pi)^n}} \exp\left( -\frac{1}{2} \xi^T \xi \right) \tag{4.7}$$

Here $n$ is the dimension of $\xi$. What distinguishes this Wiener–Hermite expansion from many other possible complete sets of expansions is that the polynomials here are orthogonal with respect to the weighting function $W(\xi)$, which has the form of the multi-dimensional independent Gaussian probability distribution with unit variance. This is the main reason behind the use of polynomial chaos as opposed to any other arbitrary polynomial. As demonstrated by authors such as Xiu et al. [17] and Lucor et al. [58], the Hermite chaos expansion generally has an exponential convergence rate in a least square sense when $\xi$ is multi-Gaussian. However, for non-Gaussian random variables, the convergence rate may not be fast [17]. In order to handle general random inputs, one approach is the use of generalized polynomial chaos or Askey chaos as described by Xiu et al. [17]. Another approach is to use a normal score transformation of the non-normal random variables, after which the Hermite chaos expansion can be used [18].

### 4.2. The Probabilistic Collocation Method

The polynomial chaos expansions discussed above can be used within the framework of the probabilistic collocation method (PCM) [19] (and the very similar stochastic response surface method (SRSM) [18]) to perform uncertainty analysis of computationally expensive models at a very low computational cost. The general collocation method is one of several mathematical techniques for reducing a model to a simpler form (e.g., differential equations to algebraic equations). Due to their similarity, we will discuss PCM and point out the differences in SRSM whenever necessary. The probabilistic collocation method described adapts the general technique by using random variables to represent the uncertain parameters. The two main advantages of this approach are that, firstly, it treats the forward model as a black box that has some set of inputs or parameters and some set of outputs or responses, and secondly, uncertainty in the input and output parameters are described by probability density functions, which is the most general representation of uncertainty [13]. Given the model and the uncertainty descriptions of the parameters, the method allows us to

determine: (a) the probability density function of the responses of interest, (b) sensitivity analysis information, (c) variance analysis information, and (d) correlations between parameters and responses.

The basic concept of the probabilistic collocation method is to try to approximate the response of the model $y = f(\mathbf{x})$, where $\mathbf{x}$ is the vector of random variables, as some polynomial function of the uncertain parameters. From Equation (4.4):

$$\tilde{y} = \sum_{j=0}^{P} \hat{a}_j \psi_j(\xi) \tag{4.8}$$

Here, $\psi_j(\xi)$ is the set of polynomial chaos expansions in terms of the independent random variables $\xi$, and $\tilde{y}$ is the approximation of $y$. Note that the expansion is truncated to *P* terms for practical applications. In the case when $\xi$ is a set of Gaussian random variables, $\psi_j(\xi)$ is the set of Hermite polynomials. In the case of SRSM, $\xi$ is always a set of Gaussian random variables obtained by normal score transformation of $\mathbf{x}$, and therefore $\psi_j(\xi)$ is always the set of Hermite polynomials [18]. In the case of PCM, $\xi = \mathbf{x}$, and $\psi_j(\xi)$ would be the specific polynomials orthogonal to the *pdf* of the random variables $\xi$ [19, 17]. $\hat{a}_j$ is a set of deterministic coefficients. $\psi_j(\xi)$ is therefore known if the *pdfs* of the uncertain input parameters $\xi$ are known. Thus, in order to approximate the forward model $y = f(\mathbf{x})$ with Equation (4.8), only the coefficients $\hat{a}_j$ have to be determined. By evaluating the true forward model *P* times, Equation (4.8) can be used to derive a set of linear equations that can be solved to determine $\hat{a}_j$. The attraction of the PCM approach derives from the fact that, given the distributions of the input random variables, we can in principle determine the polynomial chaos with an exponential convergence rate. For this reason, for a given level of accuracy, the PCM approach requires a small number of forward model evaluations compared to other possible expansions. Note that orthogonal polynomials

can also be derived for discrete probability distributions, implying that the method can be applied to discrete input random variables.

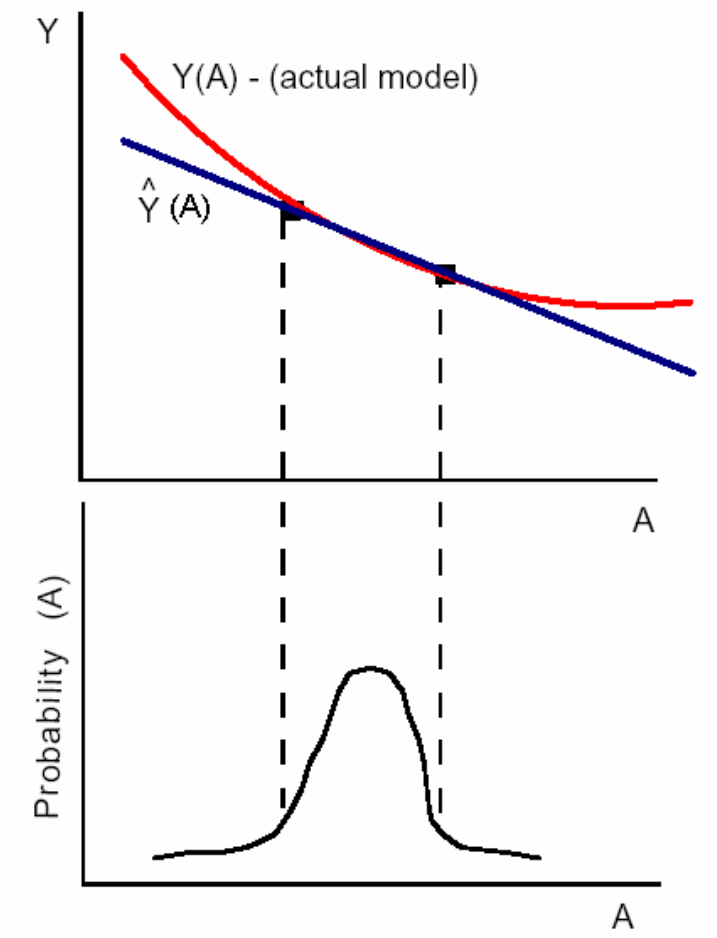


Figure 4-1 Estimation of a function in a high probability region

The crucial step in efficiently estimating a good approximation is the choice of the values of the input parameters at which the true model is evaluated. These values (collocation points) must be chosen in such a way that Equation (4.1) captures as much of the behavior of $y$ as possible. Since the uncertain input parameters have associated *pdfs*, the collocation points should be selected such that they adequately span the high probability regions of these *pdfs*. In order to achieve this, ideas from the Gaussian quadrature technique [62] for estimating integrals can be applied. Using this approach, an estimate of the integral of a polynomial can be obtained as a summation using the roots of the next higher order polynomial. Similarly, in the probabilistic collocation

method, we use the roots of the next higher order polynomial chaos as the points at which to solve the approximation.

Assume, for example, that polynomials of different orders have to be derived based on the probability density function $P(A)$ shown in Figure 4-1. To estimate a linear approximation of $y = f(A)$, two points that span the high probability region of $P(A)$ are required. The roots of the second order polynomial $\psi_2(\xi)$ provide these two points, as illustrated in Figure 4-1. By using the two points bounding the high probability region of $P(A)$, an accurate estimate $\tilde{y}$ for $y = f(A)$ can be obtained. The larger deviation of $\tilde{y}$ from $y$ only occurs in the low probability region and it thus contributes only a small error.

What then is required is a way to derive a set of polynomials from the probability density function of each input parameter such that the roots of each polynomial are spread out over the high probability region for the parameter. Further, the number of terms *P* in the polynomial expansion of Equation (4.8) should be as small as possible (while providing a good approximation of the forward model at the same time) as the number of true model evaluations is equal to *P*. Both these properties are usually satisfied by the orthogonal polynomial chaos expansions and are the main reasons for their use as opposed to any arbitrary polynomial. The following elaborates on the steps involved in the application of PCM/SRSM [18,62]:

***Step 1: Representation of Stochastic Inputs***

The first step in the application of the SRSM is the representation of all the model input distributions in terms of a set of standardized random variables of zero mean and unit variance. This step is not required in PCM as the input random variables are directly used in the polynomial chaos expansions. Isukapalli [18] describes various methods for performing this transformation, including direct transformations,

transformation with series approximations and transformation of empirical distributions.

***Step 2: Derivation of Orthogonal Polynomials***

This step is not required for SRSM as the polynomial chaos always consists of Hermite polynomials. In the case of PCM, the set of orthogonal polynomials for each of these input distributions must be derived. In case of non-Gaussian but analytical distributions like uniform distribution, Poisson distribution, etc., the correct set of orthogonal polynomials can be determined from the Askey scheme [17]. In case of empirical distributions, software such as ORTHOPOL can be used to derive the appropriate set of orthogonal polynomials [62].

***Step 3: Functional Approximations of Outputs***

Using Equation (4.8), the polynomial chaos expansion of the output can be written in terms of the orthogonal polynomials derived in step 2. Since the forward model is a black box, one might start with the simplest polynomial chaos, i.e., polynomial chaos of order 1 (linear chaos). For example, considering Gaussian random variables, the 1st and 2nd order polynomial chaos expansions are given as follows:

$$\begin{aligned}\tilde{y} &= a_{0,1} + \sum_{i=1}^{n} a_{i,1}\xi_i \\ \tilde{y} &= a_{0,2} + \sum_{i=1}^{n} a_{i,2}\xi_i + \sum_{i=1}^{n} a_{ii,2}\left(\xi_i^2 - 1\right) + \sum_{i=1}^{n-1}\sum_{j>i}^{n} a_{ij,2}\xi_i\xi_j\end{aligned} \tag{4.9}$$

Here $n$ is the number of random variables used to represent the uncertainty in the model inputs, and the coefficients $a_{i,m}, a_{ij,m}$ are the coefficients to be estimated. Note that the higher the order of the expansion, the better is the approximation. However, the number of unknown coefficients to be determined and therefore the number of collocation points increases quickly as the order is increased. For example, the number of collocation points required for a 2nd order and 3rd order expansion are as follows:

$$N_2 = 1 + 2n + \frac{n(n-1)}{2}$$
$$N_3 = 1 + 3n + \frac{3n(n-1)}{2} + \frac{n(n-1)(n-2)}{6} \tag{4.10}$$

***Step 4: Estimation of Coefficients in Functional Approximation***

In order to find a good approximation for the model with the fewest number of model runs, it is important to carefully select the parameter values, or collocation points. As noted earlier, the method for selecting collocation points is derived from the same idea as the Gaussian quadrature method to numerically solve integrals [62]. In the collocation method, the points for the model runs are selected from the roots of the next higher order orthogonal polynomial for each uncertain parameter. In general, the number of roots is usually much larger than the number of collocation points required, especially when the number of input random variables is large. The highest probability roots are usually chosen first as the collocation points. For one-dimensional problems, this method gives the same results as Galerkin's method [18] and hence is regarded as an "optimal method". Once the $P$ collocation points are chosen, the true model $y = f(\mathbf{x})$ is evaluated $P$ times. Using Equation (4.8) and the $P$ true responses, $P$ linear equations can be derived, which can be solved to obtain the $P$ coefficients $\hat{a}_j$.

***Step 5: Evaluation of Convergence of Approximation***

Before using the approximation, the convergence of the approximation has to be determined. To check the error of convergence, the model has to be run a few more times, and the model results compared to the approximation results. For the error check model runs, more collocation points are required. These points are derived from the next higher order orthogonal polynomials. The next higher order is used because if the errors are too large and a higher order of approximation is required, we will already have the model solutions needed to solve the approximation.

For each of the higher order collocation points, both the true model $y = f(\mathbf{x})$ and the approximation $\tilde{y} = \sum_{j=0}^{P} \hat{a}_j \psi_j(\boldsymbol{\xi})$ have to be solved. The convergence error is computed as:

$$\varepsilon = \sum_i \left( y - \tilde{y} \right)_i^2 P_{\boldsymbol{\xi}}(\boldsymbol{\xi}_i) \tag{4.11}$$

Here, $P_{\boldsymbol{\xi}}(\boldsymbol{\xi})$ is the joint probability density at the corresponding values of the uncertain parameters. If the error is small enough, the approximation can be used for uncertainty analysis, otherwise, the next higher order approximation is evaluated and the process repeated until convergence is achieved.

***Step 6: Use Approximation for Uncertainty Analysis***

Once a sufficiently accurate approximation has been determined, it can be used for a variety of statistical analyses. For example, a Monte Carlo simulation can be used to obtain the probability density function of $y$, as the approximation is algebraic and therefore its evaluation very fast. Other information such as the mean or standard deviation is even simpler to obtain. For example, the mean of $y$ is the expected value $E[y]$, which is just the coefficient $\hat{a}_0$ of the approximation. This is a result of the use of orthogonal polynomials; the expected value of every other term in $\tilde{y}$ is 0 because they contain products of different order orthogonal polynomials. In fact, for the closed-loop application described below, only the mean of $y$ ($y$ is NPV in the example) is required. The standard deviation is similarly simple to calculate.

One problem with the PCM as described above is that the input random variables have to be independent. In the case when the random input properties consist of random fields (a set of spatially distributed correlated random variables) or random processes (a set of temporally distributed correlated random variables), the standard probabilistic

collocation method cannot be used directly due to the independence assumption of the input random variables $\xi$. This however is usually the case with reservoir simulation models, whose input parameters such as porosity and permeability of one grid cell are related to those of another grid cell. The Karhunen-Loeve expansion, as explained in the last chapter, allows the expression of a correlated random field or process in terms of a set of independent random variables while maintaining the covariance structure. Thus, once the K-L expansion of the input random field is performed, the PCM can then be performed using the independent random variables obtained with the K-L expansion. Further, the number of transformed independent random variables is much smaller than the initial correlated random variables, which is very beneficial for PCM as the number of collocation points and therefore the efficiency is directly dependent on the number of input variables. Refer to Chapter 3 for a description of the Karhunen-Loeve expansion.

### 4.3. Application of PCM+KLE to a Gaussian Random Field

To demonstrate the validity of the PCM+KLE approach, we consider a multi-Gaussian permeability field (log permeability) with exponential covariance. The correlation length is 3/5 of the spatial dimension of the model, and there is a nugget effect of 0.1. The lognormal permeability distribution has a mean of 100 and standard deviation of 48. The model size is 50×50, that is, the number of correlated random variables (log permeability) is 2500. All other input properties of the reservoir are assumed to be deterministic. The output random variable considered is net present value (NPV). A set of 1000 realizations is created by unconditional simulation using *sgsim* [63]. Some of the realizations are shown in Figure 4-2. The simulation model is a quarter 5-spot 2D 2-phase black oil model with a water injector under rate control (left bottom corner) and a producer under BHP control (right top corner). The model is run for 900 days.

Karhunen-Loeve expansion is performed using the 1000 realizations and the eigenvalues and their energy are plotted in Figure 4-3 and Figure 4-4. We observe that

most of the energy is associated with the first few eigenpairs. For the purpose of model updating, we chose to retain only 10 eigenpairs; this corresponds to about 65% of the total energy. The number of independent random variables is thus reduced from 2500 to 10. Note that the fraction of energy associated with a number of eigenpairs is primarily dependent of the correlation lengths and not on the number of grid cells, and therefore, as long as the correlation lengths are sufficiently long, such large reduction in retained eigenpairs should also be possible for large-scale models. Figure 4-5 and Figure 4-6 show the reconstruction of the reference permeability field with the first 10 eigenpairs. As before, it is clear that although the long correlation features are preserved, the smaller correlation structures are lost. However, since flow behavior and therefore NPV is mainly dependent on longer correlation lengths, such an approximation should be reasonable, at least for multi-Gaussian type structures.

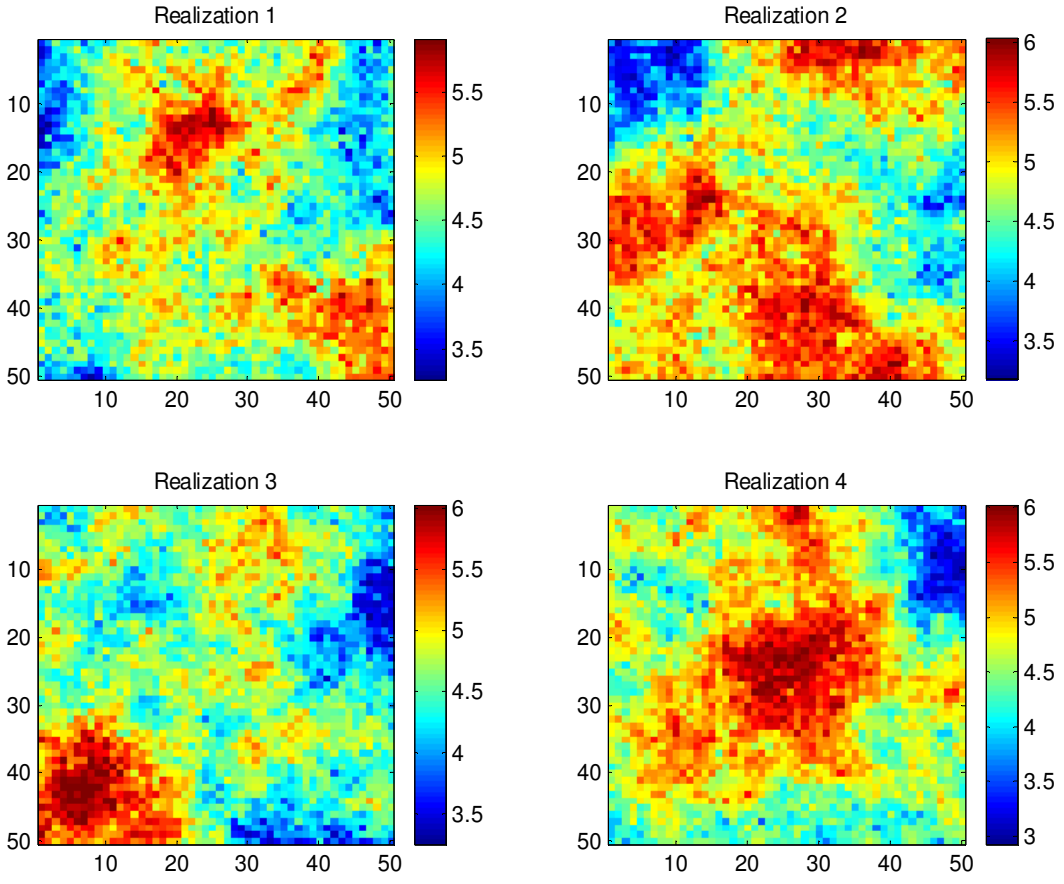


Figure 4-2 A few realizations with Gaussian permeability and exponential covariance

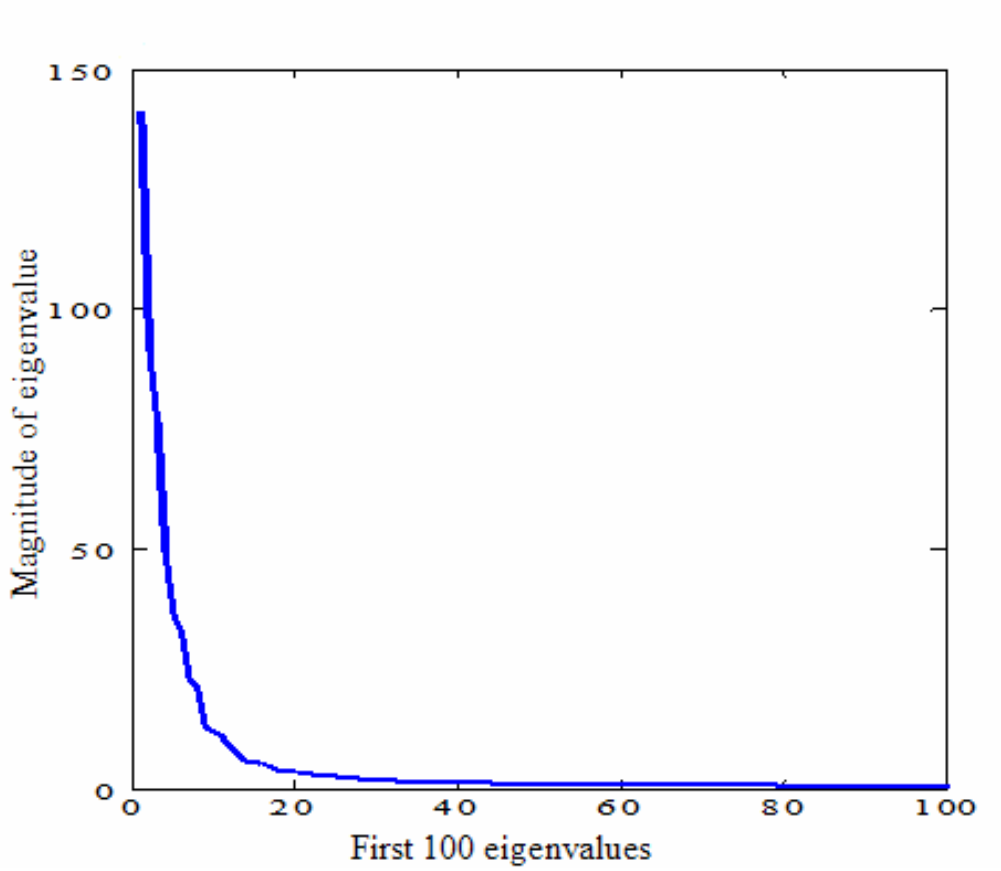

Figure 4-3 Eigenvalues of the Covariance Matrix

Since the initial random field (log permeability) is Gaussian, Hermite polynomials can be used directly. A 2nd order expansion is used, and as seen from Equation (4.10), as the number of random variables is 10, 66 collocation runs are required for a 2nd order expansion. Combinations of the roots of the 3rd order Hermite polynomial are used as collocation points according to the rule described earlier. Since an analytical solution is not possible, the validity of the approach is compared against Monte Carlo simulation . In order to determine the number of realizations of permeability required for Monte Carlo, a convergence study of the mean and variance of permeability is performed. Convergence of some of the permeabilities is shown in Figure 4-7. It is observed that the mean and variance of the realizations converge to the true mean and variance (i.e., the mean and variance used to generate the realizations) at around 600 realizations, implying that about this many realizations would be required to obtain a reliable *pdf* of NPV by a Monte Carlo procedure.

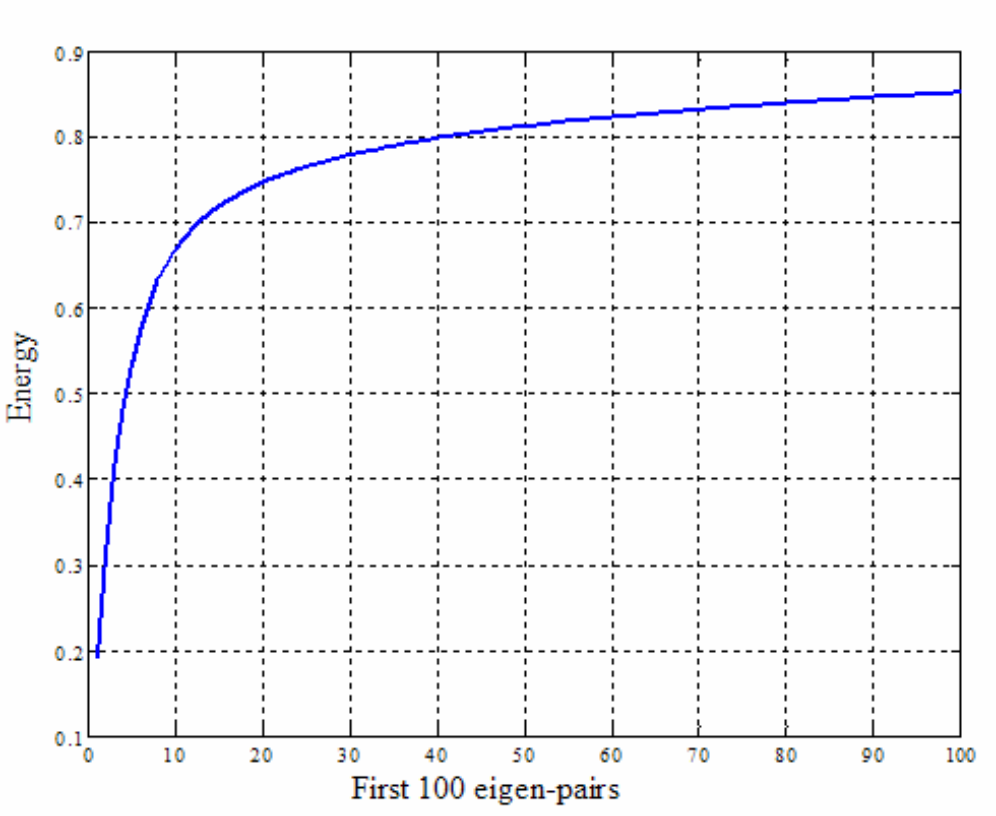


Figure 4-4 Fraction of total energy associated with the eigenpairs

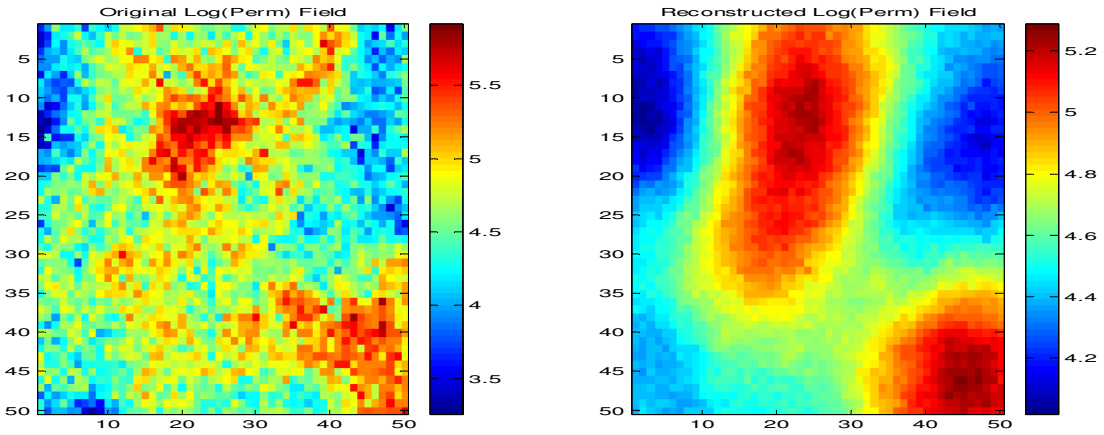


Figure 4-5 Original realization 1 and its reconstruction from first 10 eigenvectors

Figure 4-8 compares the *pdfs* of NPV obtained by Polynomial Chaos and by Monte Carlo. It is clear the PCE *pdf* is a very close approximation to the *pdf* obtained by Monte Carlo. However, 66 simulations were required in the case of PCE as compared to 600 for Monte Carlo, demonstrating a significant increase in efficiency. This can be further improved if an adjoint model is also available. Note that the Monte Carlo

method used in this work is the most basic one, and more efficient Monte Carlo techniques such as Latin hypercube sampling are also available.

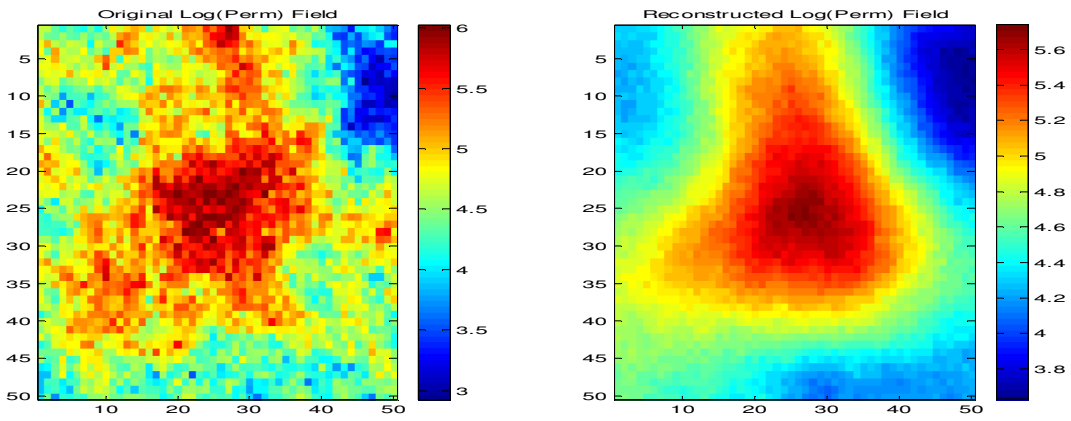


Figure 4-6 Original realization 2 and its reconstruction from first 10 Eigenvectors

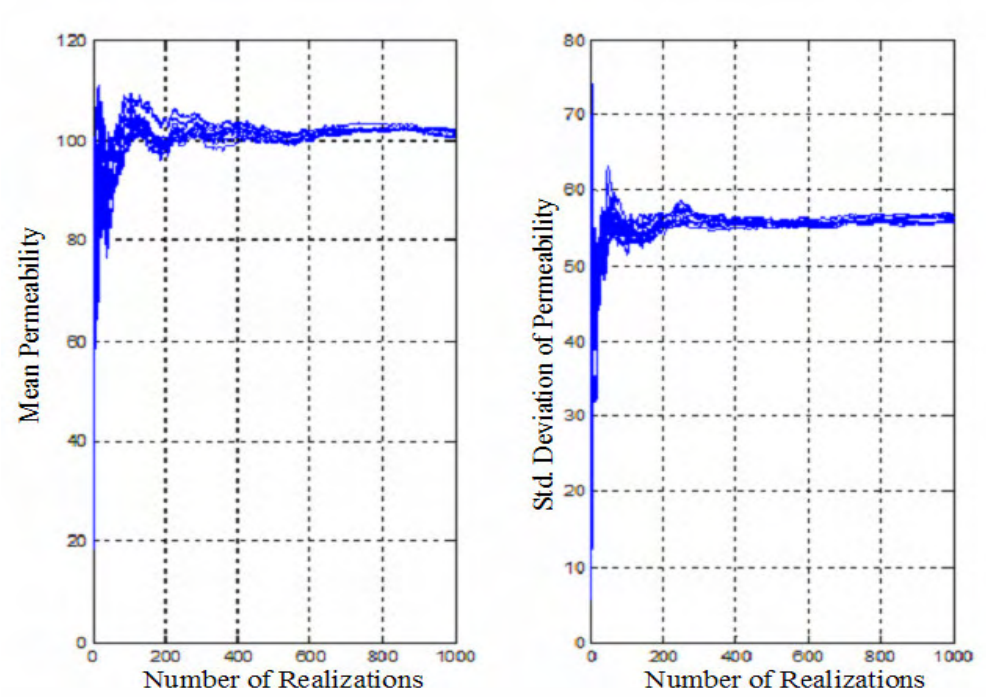


Figure 4-7 Convergence of mean and standard deviation of permeability

Table 4-1 compares the mean and variance of NPV obtained from PCE, Monte Carlo and using the 66 collocation points directly (without any polynomial chaos expansion being used). PCE and Monte Carlo display a very good agreement (7% error). Although the 66 collocation points give a reasonable value for the mean, the variance is 35% less than that obtained by Monte Carlo.

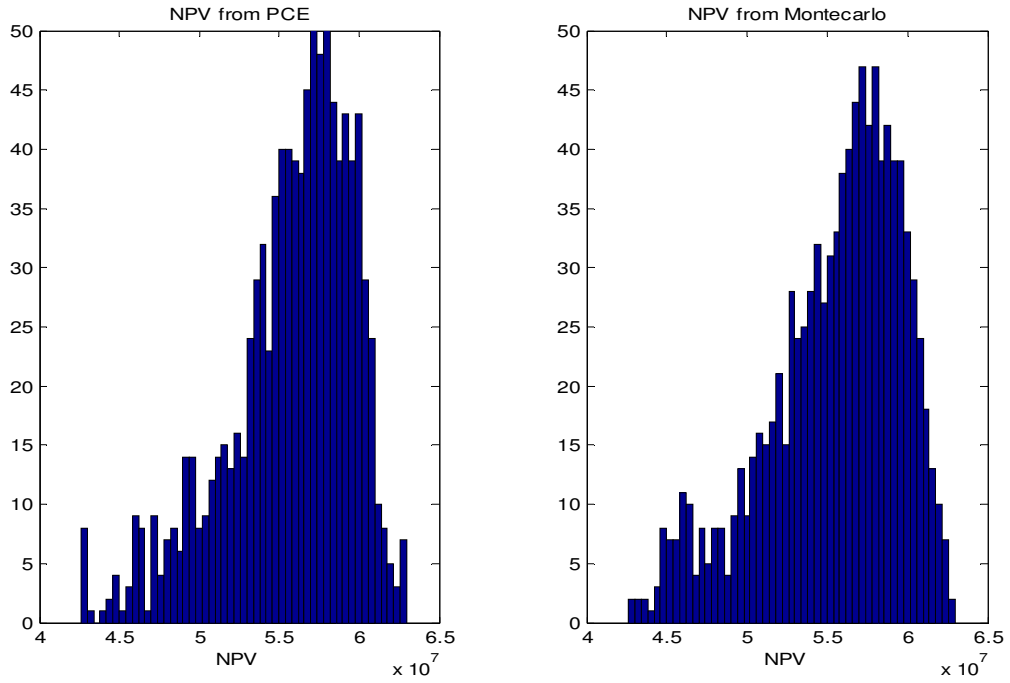


Figure 4-8 NPV distribution from PCE and Monte Carlo

| | PCE | Monte Carlo | Collocation Points |
|---|---|---|---|
| **Mean** | **5.572e+007** | **5.548e+007** | **5.593e+007** |
| **Variance** | **1.632e+013** | **1.768e+013** | **1.140e+013** |

Table 4-1 Mean and variance from PCE and Monte Carlo

## 4.4. Implementation of the Closed-Loop

Before the closed-loop formulation can be completed, the optimization algorithm described in Chapter 2 and the uncertainty propagation algorithm described above must be combined. This is required because, due to the uncertainty of the model parameters, the outputs of the forward model that comprise the objective function of the optimization problem become random variables. Since the objective function is generally required to be a scalar quantity, it usually consists of some moment of the output random variables (such as the expected value) or a combination of different moments [45]. These moments and their gradients with respect to the controls can be calculated directly from the polynomial chaos expansion, provided that the gradients of output variables at the collocation points are available (for example using the deterministic adjoints discussed earlier). Consider for example the approximation of

some output $y = f(\xi)$ of a forward model with a second order Hermite chaos representation:

$$y = a_{0,2} + \sum_{i=1}^{n} a_{i,2}\xi_i + \sum_{i=1}^{n} a_{ii,2}\left(\xi_i^2 - 1\right) + \sum_{i=1}^{n-1}\sum_{j>i}^{n} a_{ij,2}\xi_i\xi_j \tag{4.12}$$

Here $n$ is the number of input random variables. If the objective function consists only of the expected value of $y$, given by $E(y)$, then, since $\xi$ is a set of Gaussian random variables with zero mean and unit variance, $E(y) = a_{0,2}$, i.e., the first coefficient of the expansion. As explained in the last section, using the solutions at the $P$ collocation points, a linear system can be obtained to determine $a_{0,2}$ (and the other coefficients if required). The expected value and its gradient can then be written as:

$$E(y) = a_{0,2} = \sum_{i=1}^{P} b_i y_i; \quad \frac{dE(y)}{du} = \sum_{i=1}^{P} b_i \frac{dy_i}{du} \tag{4.13}$$

Here $y_i$ and $dy_i / du$ are obtained from the collocation runs of the forward model and the adjoint model (at the collocation points). Using Equation (4.13), the expected value and its gradient, as required by the optimization algorithm, can be obtained. The coefficients $b_i$ are obtained from the inverse of the matrix containing the Hermite polynomials evaluated at the collocation points [18]. Note that for a second order chaos expansion, the number of collocation runs $P$, which is equal to the number of coefficients $a_{i_1..i_n}$ to be determined, is given by the first equation of Equation (4.10).

Interestingly, if we only require the expected value $y$, then due to the nature of the roots of the third order Hermite chaos, $a_{0,2}$ is not affected by the $\xi_i\xi_j$ terms of Equation (4.12). Thus the number of collocation runs actually reduces to $1+2n$. This can be further reduced by only considering the larger $a_{i_1..i_n}$ coefficients and discarding the rest, as will be demonstrated in the example of the next section. Also, since the number of collocation runs depends on the number of input variables, reducing the

number of input variables using the K-L expansion to parameterize a random input field results in further speed up of the uncertainty propagation process.

Combining all of the components discussed in this and previous sections, the formulation of the closed-loop can be implemented efficiently. The main steps required to complete the loop are as follows:

1. From the prior model of the uncertain but correlated parameter field, calculate the covariance matrix, either numerically or analytically. Note that the covariance model may be non-stationary.

2. Perform Karhunen-Loeve expansion to determine the eigenvectors and eigenvalues of the covariance matrix. Retain only the largest eigenpairs; the number to be retained can be determined from the percentage of the total energy contained in the eigenpairs.

3. Determine the order and type of the polynomial chaos expansion to be used. Since the K-L expansion is being used for the correlated random field, an assumption of Gaussianity of this input random field has already been made. Thus the Hermite chaos representation should be used for these input random variables. Determine the coefficients $b_i$ from the values of the polynomial chaos at the collocation points.

4. Using the current estimate of the controls, evaluate the forward model and the adjoint model at the collocation points from control step $k$ to $N_t$, (total control steps) starting with $k = 1$, that is, the first control step.

5. Determine the objective function and its gradient with respect to the controls using the output from step 4 and Equation (4.13) (if the objective function only consists of the expected value of some output of the forward model) or similar equations depending on the nature of the objective function.

6. Perform optimization from control step $k$ to $N_t$, starting with $k$ = 1, that is, the first control step, to determine the new controls.

7. Apply the optimized trajectory of the controls on the "true" reservoir from control step $k$ to $k+1$, and record the reservoir response for this time period. This provides the "data" to be used for history matching.

8. Perform model updating to assimilate new data. Updating can be performed from control step 1 to the step $k+1$, that is, assimilate new data and re-assimilate earlier data, or from step $k$ to $k+1$, that is, only assimilate new data.

9. Perform steps 4 to 8 for the next control step, ($k$=$k+1$) using the new maximum likelihood estimate of the parameter field and its posterior covariance to obtain the new collocation realizations. Repeat until $k$= $N_t$.

Further discussion of some of the above steps (specifically those associated with model updating) is provided in Chapter 3. In addition, it should be noted that along with the input correlated random field, there might also be other independent input random variables of the forward model. For example, in the application demonstrated below, the permeability field is assumed unknown and is therefore the input random field. But, we may also have other unknowns such as the depth of the water-oil contact that are independent of the permeability field. These random variables can also be directly incorporated into the polynomial chaos expansion in addition to the random variables related to the K-L expansion of the correlated random field.

### 4.5. Case Study – Dynamic Waterflooding

Application of the closed-loop approach is demonstrated on a simple dynamic waterflooding example with smart wells. This example was also considered in Chapter 3; the new element here is the incorporation of uncertainty propagation via polynomial chaos expansions. Thus, much of the description below follows that in Chapter 3. Further, in Chapter 3 we discussed two approaches for model updating. Here we apply

the second such approach in which only new data (rather than all of the historical data as in the first approach) is assimilated at every control step.

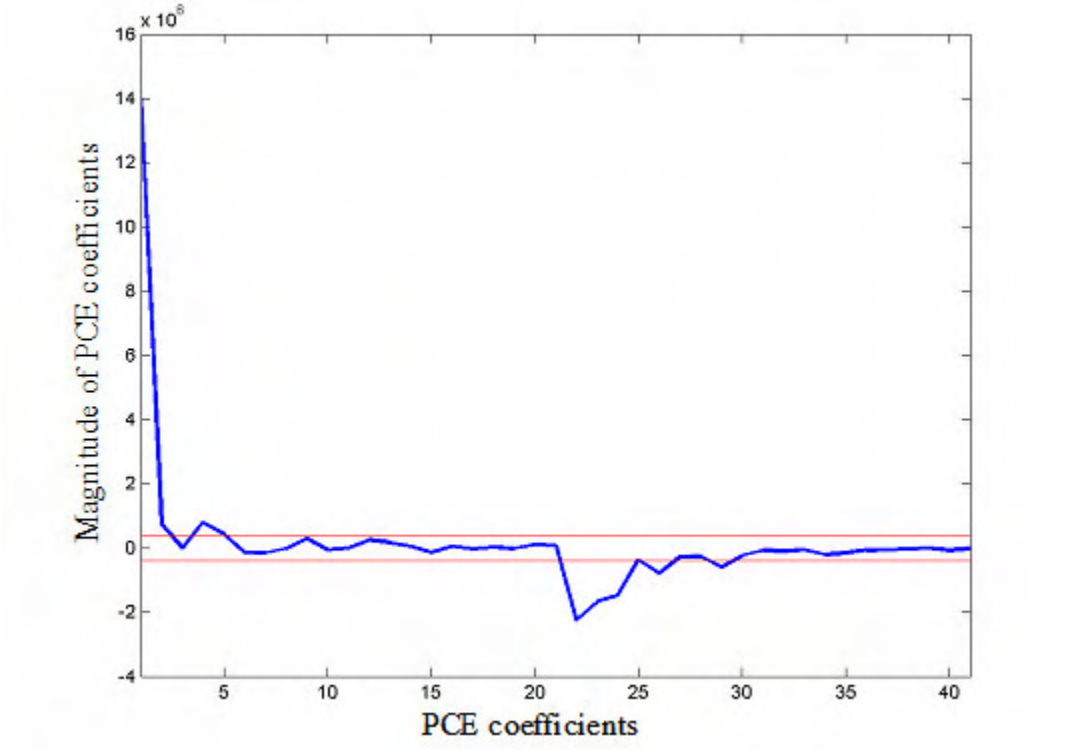


Figure 4-9 Magnitude of the coefficients of the polynomial chaos expansion

Since the simulation model is the same as in Chapter 3, it will not be discussed here. Again, permeability is assumed to be unknown and will be updated by assimilating production data. Prior knowledge of the reservoir is depicted by the training image [57] shown in Figure 3-1, with the channel sand permeability being about 10 Darcy and the background sand permeability about 500 mD. Realization 22 from Figure 3-2 is taken to be the true realization. Karhunen-Loeve expansion is performed using the covariance matrix created from 1000 initial realizations. For the purpose of model updating and uncertainty propagation, we chose to retain only 20 eigenpairs; this corresponds to about 68% of the total energy of the eigenpairs.

The production scenario is also the same as in the example from Chapter 3. There is a total injection constraint of 2700 STB/day (STBD). The model is produced until exactly one pore volume of water is injected, which corresponds to around 950 days of

injection. This time period is divided into seven control steps of 30, 60, 100, 190, 190, 190 and 190 days. Thus the total number of controls is equal to (45 + 45)×7 = 630. These seven control steps also correspond to the model updating steps. The injection BHPs and producer water and oil rates from the "true" model are used as data to update the permeability field.

The base case is again a constant rate/constant BHP production strategy. The 2700 STBD of injection water is distributed among the 45 injection segments according to their *kh*, which corresponds to an uncontrolled case. The producer BHPs are set in such a way that a balanced injection-production is obtained.

The key difference from Chapter 3 is that NPV is now treated as a random variable. Thus, the objective of the optimization process is to maximize the expected value of NPV with discounting factor set to zero. The oil price is conservatively set at \$80/$m^3$, water injection cost at \$0/$m^3$, and water production cost at \$20/$m^3$.

To perform uncertainty propagation, a second order Hermite chaos is chosen to represent NPV. As there are 20 input random variables (from the K-L expansion), and we are only interested in the expected value of NPV, the number of coefficients to be determined is equal to 41 (from Equation (4.10)). The magnitude of these coefficients calculated using the uncontrolled case is shown in Figure 4-9. We observe that most of the coefficients are quite small in magnitude, and can thus be eliminated without affecting the calculated NPV significantly. Using a cutoff of ± 0.4e6 (red lines in Figure 4-9), the number of coefficients is reduced to 9. This is the number of collocation runs that is required to calculate the expected value of NPV and its gradient (with the adjoint). Note that as the controls change during the optimization, the forward model that the Hermite chaos represents also changes, and thus the ranking (in magnitude) of the 41 coefficients may not remain the same, implying that the coefficients to be discarded may change. Retaining these particular 9 coefficients is thus based on the assumption that the change in the forward model does not affect the ranking of the coefficients. This issue requires further investigation.

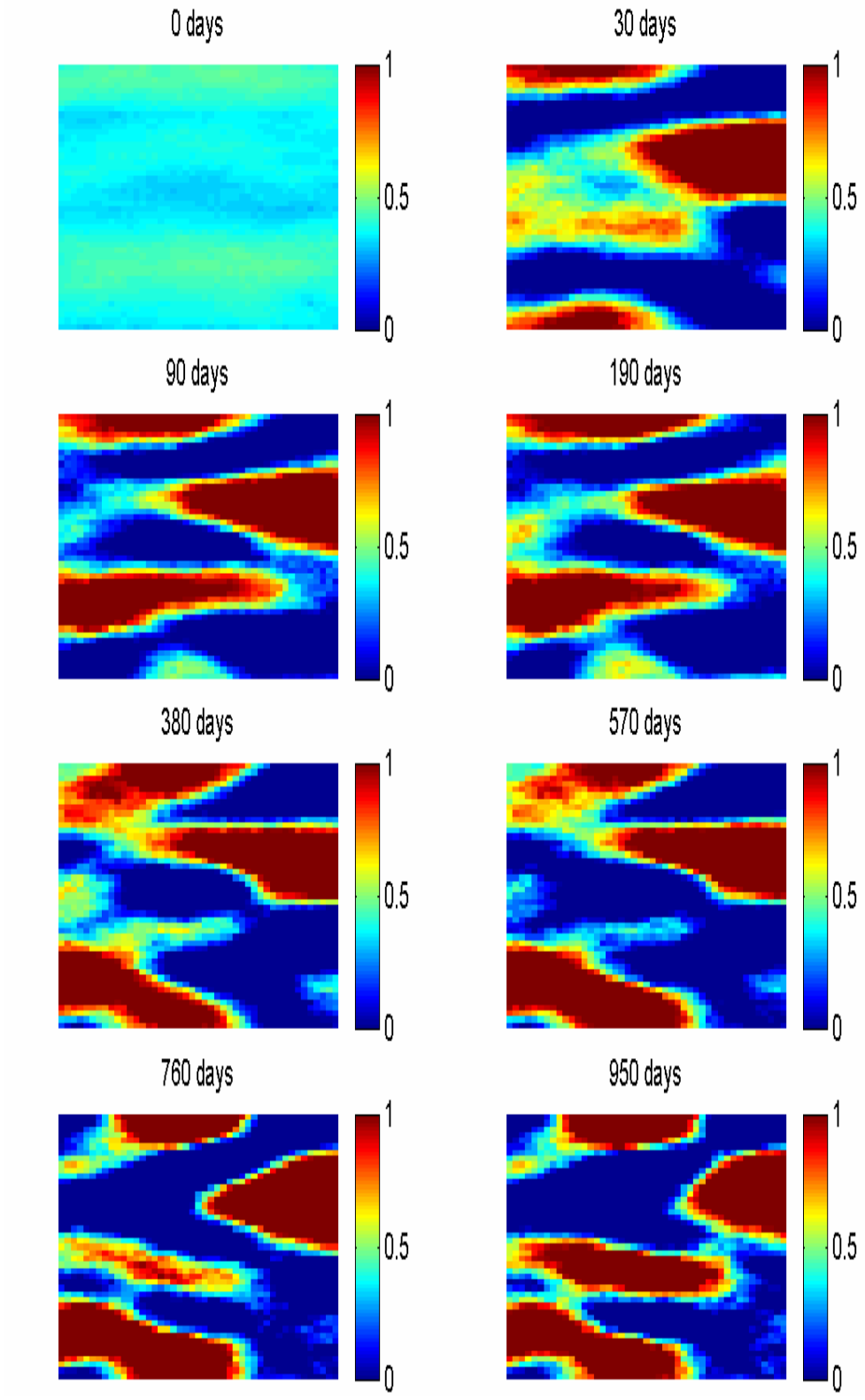


Figure 4-10 Permeability field updates obtained with the closed-loop approach with uncertainty propagation

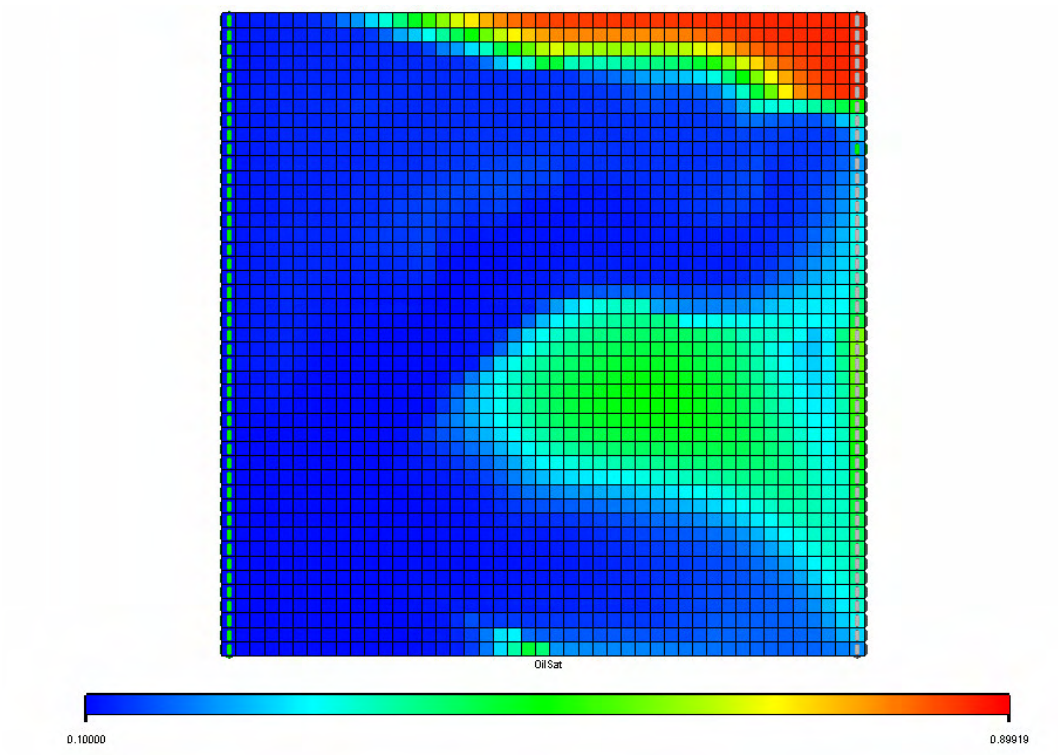


Figure 4-11 Final oil saturations obtained with the closed-loop approach with uncertainty propagation

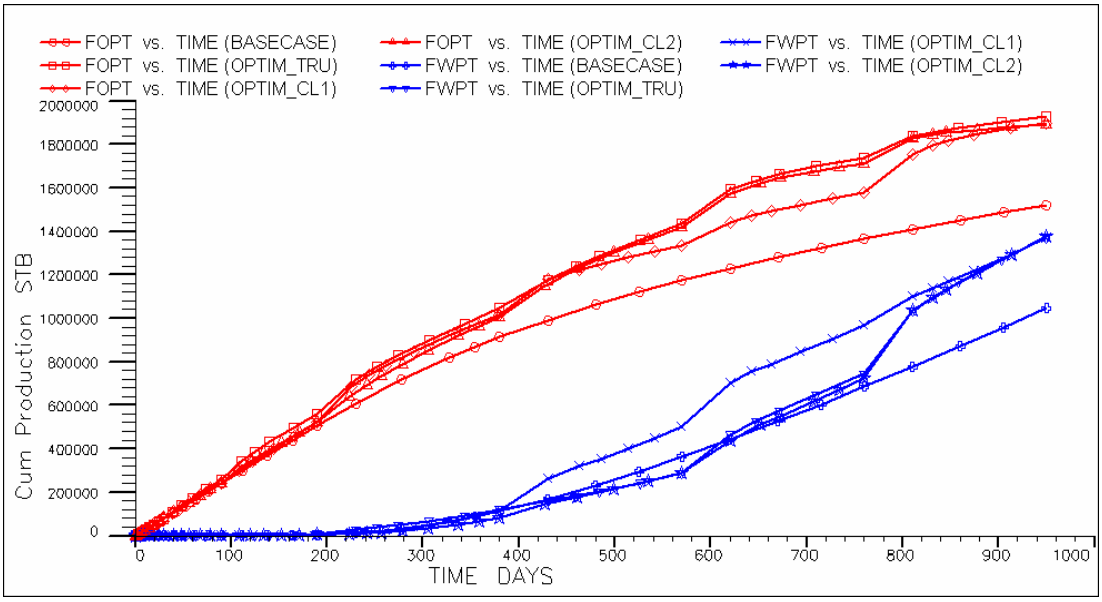


Figure 4-12 Comparison of cumulative production for base (uncontrolled) case and optimized case (OPTIM_TRU) run with "true" realization, and closed-loop with (OPTIM_CL2) and without (OPTIM_CL1) uncertainty propagation

Figure 4-10 and Figure 4-11 show the maximum likelihood models and final saturation map obtained using the complete closed-loop with uncertainty propagation. Again, as in the simplified closed-loop of Chapter 3, the sweep obtained is almost as good as with the open loop, and in this case the sweep patterns are also very similar. This is due to the fact that the maximum likelihood estimates of the permeability field (Figure 4-10) provide reasonable optimization trajectories. This is also evident in the cumulative production profiles of Figure 4-12, where the production profiles obtained with the complete closed-loop follow the profiles obtained from the optimization on the true field very closely. Although the final NPV obtained at 950 days is very close for both closed-loop approaches (around 25% over reference), the loop with uncertainty propagation clearly displays better performance for the time period before about 750 days.

## 4.6. Summary

We have demonstrated that adjoint models combined with bi-orthogonal expansions, in particular K-L and Hermite expansions, can be used for efficient uncertainty propagation and closed-loop optimal control. Significant improvements in recovery appear to be possible through the application of such techniques. In particular, the following may be concluded:

1. Polynomial chaos expansions may be used for efficient uncertainty propagation, with the main benefit that standard deterministic forward models and associated adjoints can be used as in a black box approach.

2. For the example considered, the increase in NPV and sweep efficiency by the closed-loop approach is very close to that obtained using an open-loop approach. In addition, the results indicate that approximate parameter fields may be adequate for obtaining near optimal trajectories of the controls.

3. Including uncertainty propagation within the closed-loop results in more reliable estimates of the optimal trajectories, as opposed to a closed-loop applied only on the maximum likelihood estimate of the uncertain parameter field.

A few additional issues concerning the polynomial chaos and Karhunen-Loeve expansions need to be addressed. We need to explore whether preserving two-point statistics is sufficient for the purpose of optimal control under complex geological scenarios. Further, the optimal order of polynomial chaos to be used, and the number of terms to be retained, is yet to be fully understood. Some of these issues will be addressed in later chapters.

# Chapter 5

# 5. Handling Nonlinear Path Inequality Constraints

This chapter is focused on the optimization component of the closed-loop process with emphasis on handling nonlinear path inequality constraints. This problem can essentially be cast as a large-scale path constrained optimal control problem. A large variety of methods for solving discrete-time optimal control problems now exist in control theory literature, including dynamic programming, neighboring extremal methods, gradient-based nonlinear programming methods (NLP), etc. These are discussed in detail in Stengel [37] and Bryson and Ho [20]. Of these approaches, the NLP method combined with the *Maximum Principle* [20] (adjoint models) generates a class of NLP methods in which only the control variables are the decision variables and the state variables are obtained from the dynamic equations. These algorithms are generally considered more efficient compared to the other methods. Further, within this class of NLP methods, there are many existing techniques available for handling nonlinear control-state path inequality constraints [20, 64, 65, 66]. However, as will be discussed later, these are either not practical for the production optimization problem or are difficult to implement with existing reservoir simulator codes.

In petroleum engineering literature, papers by various authors such as Asheim [31], Vironovsky [32], Brouwer and Jansen [34], etc. have discussed the application of adjoint models and gradient techniques for the production optimization problem in significant detail. However, an important element that is missing from most of these papers is the effective treatment of nonlinear control-state path inequality constraints (for example, a maximum water injection rate constraint). Such constraints are always present in practical production optimization problems, and therefore their efficient treatment is essential for such algorithms to be useful. In one of our earlier papers [10],

two methods to handle such constraints were discussed; however, they either do not satisfy the constraints exactly or are applicable only for small problems. In petroleum engineering literature, a paper by Zakirov et al. [30] discusses an approach to implement path constraints; however, there are certain theoretical issues with the approach, as discussed in a later section of this chapter.

In this chapter, we propose an approximate feasible direction optimization algorithm suitable for large-scale optimal control problems that is able to handle nonlinear inequality path constraints effectively while always maintaining feasibility. Other advantages are that only two adjoint simulations are required at each iteration and large step sizes are possible during the line search at each iteration, leading possibly to large reductions in the objective function. This method belongs to the class of NLP methods combined with the maximum principle (adjoint models) discussed above.

This chapter proceeds with a brief description of the mathematical formulation of the problem. This is followed by a discussion of the existing methods for handling nonlinear path constraints for optimal control problems, with a critical comparison of their advantages and disadvantages. The next section discusses the traditional feasible direction optimization algorithm in some detail, as it is the basis of the proposed algorithm. This is followed by detailed discussions of the proposed approximate feasible direction and feasible line search algorithms. The validity and effectiveness of the proposed algorithm for handling nonlinear path inequality constraints is demonstrated through two examples, one with a maximum water injection constraint, and the other with a maximum liquid production constraint, both of which are nonlinear with respect to the controls (BHPs in this case).

## 5.1. Production Optimization with Adjoint Models

Similar to Chapter 2, the problem definition is given by Equation (5.1). The difference of this problem definition compared to that of Chapter 2 is that the linear path constraints have been replaced by more general nonlinear path constraints. As in

Chapter 2, the set of equations $g^n$ together with the initial condition define the dynamic system. In the current application, $g^n$ is the fully implicit reservoir simulation equations written for each grid block at each time step.

$$\max_{\mathbf{u}^n}\left[ J\left(\mathbf{u}^0,...,\mathbf{u}^{N-1}\right)=\phi\left(\mathbf{x}^N\right)+\sum_{n=0}^{N-1} L^n\left(\mathbf{x}^{n+1},\mathbf{u}^n\right)\right] \forall n\in\left(0,..,N-1\right)$$

subject to:

$$\begin{aligned}
&g^n(\mathbf{x}^{n+1},\mathbf{x}^n,\mathbf{u}^n)=0 && \forall n\in\left(0,..,N-1\right)\\
&\mathbf{x}^0=\mathbf{x}_0 && \text{(Initial Condition)}\\
&c^n(\mathbf{x}^{n+1},\mathbf{u}^n)\le 0 && \forall n\in\left(0,..,N-1\right)\\
&c^n(\mathbf{x}^{n+1})\le 0 && \forall n\in\left(0,..,N-1\right)\\
&\mathbf{LB}\le\mathbf{u}^n\le\mathbf{UB} && \forall n\in\left(0,..,N-1\right)
\end{aligned} \qquad (5.1)$$

The last three equations of Equation (5.1) define additional constraints for the controls – nonlinear inequality path constraints that are functions of both states and controls, nonlinear inequality path constraints that are functions of states only, and bounds on the controls. Note that these are called path constraints because they have to be satisfied at every time step. Further, only inequality constraints are considered because almost all constraints in practical problems are inequality constraints. Examples of such constraints are maximum injection rate constraint, maximum watercut constraint, maximum liquid production rate constraint, etc. Note that whether a constraint is linear or nonlinear also depends on the choice of control variables. For example, well rates are nonlinear functions of the BHPs of wells, and hence any rate constraint will be a nonlinear path constraint if BHPs are controlled; but may become a linear constraint if well rates are controlled directly. The control-state constraints and state only constraints are written separately because in general, state only constraints are more difficult to handle, and some existing algorithms treat them in different ways. In our proposed method, however, both of them will be treated with one unified approach.

As discussed in Chapter 2, adjoint methods can be applied to calculate the gradient of the cost function $J\left(\mathbf{u}^0,...,\mathbf{u}^{N-1}\right)$ with respect to the controls $\mathbf{u}^n$ very efficiently. However, the derivation in Chapter 2 did not include the nonlinear path constraints discussed above. In the absence of such additional constraints, the gradients derived in Chapter 2 can be used directly with any optimization algorithm. However, if nonlinear path constraints are present, their effect on the optimization has to be taken into account. The treatment of these constraints is discussed in the following sections.

## 5.2. Existing Methods for Nonlinear Path Constraints

It is acknowledged in control theory literature [20] that nonlinear control-state path inequality constraints involving state variables are the most difficult to incorporate effectively into optimal control algorithms. Even the maximum principle as given by Pontryagin does not apply directly to such problems [20]. As discussed earlier, among the various classes of algorithms available for optimal control problems, gradient-based NLP algorithms combined with adjoint models are generally considered the most efficient, and there are a number of existing NLP algorithms designed for path constrained optimal control problems. This class of NLP algorithms can be further divided into two categories: (1) algorithms that solve the path constraints implicitly together with the dynamic system or convert them to simple bounds constraints, implying that the NLP becomes an unconstrained NLP, thus constraint gradients are not required, and (2) algorithms that calculate the path constraints explicitly after the dynamic system has been solved, implying that the NLP becomes a constrained NLP, thus constrained NLP algorithms are required. The first four algorithms discussed next belong to the first category, and the last two belong to the second category.

One of the early methods given by Bryson et al. [20, 67, 68] incorporates the path constraints into the forward and adjoint equations in a manner similar to the dynamic system equations. In other words, the cost function is not only augmented with the dynamic system equations (see Equation (2.4)), but also with the path constraints, using an additional set of Lagrange multipliers. For example, if only control-state

constraints are present, the augmented cost function is given by the following equation, where $\boldsymbol{\mu}^n$ are the additional Lagrange multipliers:

$$J_A = \phi\left[\mathbf{x}^N\right] + \sum_{n=0}^{N-1} L^n\left(\mathbf{x}^{n+1}, \mathbf{u}^n\right) + \sum_{n=0}^{N-1}\left\{\boldsymbol{\lambda}^{T(n+1)} g^n\left(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{u}^n\right) + \boldsymbol{\mu}^{T(n+1)} c^n\left(\mathbf{x}^{n+1}, \mathbf{u}^n\right)\right\} \quad (5.2)$$

There is, however, an additional requirement on the Lagrange multipliers $\boldsymbol{\mu}^n$ (see Bryson and Ho [20] for details). The numerical algorithm essentially involves joining together the constrained and unconstrained parts of the trajectories using the necessary conditions of optimality. This method is capable of finding the exact optimum quite efficiently and solves the path constraints implicitly together with the dynamic system, implying that solutions obtained at any iteration are always feasible. However, a major drawback of this technique is that the sequence of constrained and unconstrained parts of the trajectories at the optimum must be known apriori, and this information is not available for production optimization problems. Further, control-state and state only path constraints are not treated in the same manner, and state only constraints are generally quite difficult to implement with this approach [20].

Another method known as the *Generalized Gradient* method given by Mehra and Davis [64] shows that the difficulties associated with the method given by Bryson et al. [20, 67, 68] can be avoided by choosing different combinations of the control and state variables as the independent variables, instead of always choosing the control variables as independent variables. The different combinations of the control and state variables as independent variables are dictated by the constraints and could result in different combinations along different parts of the trajectory. The gradient of the cost function with respect to the independent variables, called the generalized gradient, is calculated by solving a set of equations similar to the Euler-Lagrange equations [64]. Unfortunately, there are a few disadvantages of this method as well. The path constraints are only calculated explicitly after the dynamic system has been solved, resulting in possible infeasible solutions, in which case one has to start from a new

guess. More importantly, since this method requires the ability to independently control the state variables, implementing this method is difficult in conjunction with existing simulator codes. For the same reason, standard NLP software cannot be easily used with this approach.

Yet another method known as the slack variable method (Jacobson and Lele [69]; Feehery [65]) essentially changes the original inequality constraint to an equality constraint by means of a slack variable *a(t)*:

$$c^n(\mathbf{x}^{n+1},\mathbf{u}^n)+\frac{1}{2}\left(a^n\right)^2=0 \qquad \forall n\in\left(0,..,N-1\right) \tag{5.3}$$

The slack variable is squared so that any value of *a* is admissible. The principle of the method is to make one of the control variables appearing in the constraint a new state variable for the corresponding constraint. Thus, the constraint is appended as an equality constraint to the dynamic system, the control variable is solved as a state variable during the forward solve and the slack variable *a(t)* becomes the new control variable. Although the method is appealing and efficient, and like the first method, only generates feasible solutions, it again has some drawbacks, the main one being that, as the method changes the category of a control variable to a state variable, this means that any possible bounds on control variables cannot be satisfied. Unfortunately, almost all control variables for production optimization problems have either physical or economic bounds. Also, if more than one control variable is a candidate for conversion to a state variable, no suitable selection strategy exists. In practice, since a combination of control variables influences the state constraint, the method is not able to choose the correct one. Further, this method also requires significant modifications to existing simulation code.

In petroleum engineering literature, Zakirov et al. [30] have proposed an algorithm for implementing inequality path constraints. Their method is similar to that of Bryson et al. [20, 67, 68] in the sense that the active constraints are adjoined to the cost function

by an additional set of Lagrange multipliers. The gradient of the objective function with respect to the independent controls is then calculated using the usual Euler-Lagrange equations (adjoint system). In order to calculate the step size, it is required that none of the constraints is violated by taking that step in the search direction. To do this, a problem of variations is solved. However, in order to do so, it is implied that the constraints that are active at a given iteration will also remain active in the next and succeeding iterations. There is no reason why this should be true, and such a scheme would result in the problem becoming overly constrained with succeeding iterations. It is also not clear how it is determined which constraints will be active at the optimum, and which controls will be kept independent.

The algorithms considered above solve the constraints implicitly together with the dynamic system. At the other extreme are algorithms that calculate the constraints explicitly after the dynamic system (forward model) has been solved, in order to determine whether the constraints were violated or not. The most popular among this class of algorithms for optimal control problems has been the penalty function approach and its variants [20, 70]. Penalty function methods transform the constrained optimization problem into alternative formulations such that the numerical solutions are sought by solving a sequence of unconstrained optimization problems [70]. For example, the production optimization as given by Equation (5.1) is converted to the following problem:

$$\max_{\mathbf{u}^n} \Phi_k\left(\mathbf{u}^0,..,\mathbf{u}^{N-1},r_k\right) = J\left(\mathbf{u}^0,..,\mathbf{u}^{N-1}\right) + r_k \sum_{n=1}^{N} \Upsilon\left[c^n\left(\mathbf{x}^{n+1},\mathbf{u}^n\right)\right] \forall n \in (0,..,N-1)$$

subject to:

$$\begin{aligned} & g^n(\mathbf{x}^{n+1},\mathbf{x}^n,\mathbf{u}^n) = 0 \qquad \forall n \in (0,..,N-1) \\ & \mathbf{x}^0 = \mathbf{x}_0 \qquad \text{(Initial Condition)} \\ & LB \le \mathbf{u}^n \le UB \qquad \forall n \in (0,..,N-1) \end{aligned} \tag{5.4}$$

Here, $\Upsilon\left[c^n\left(\mathbf{x}^{n+1},\mathbf{u}^n\right)\right]$ is some function of the constraint $c^n\left(\mathbf{x}^{n+1},\mathbf{u}^n\right)$ and $r_k$ is a positive constant known as the penalty parameter. With respect to $\mathbf{u}^0,..,\mathbf{u}^{N-1}$, the

above optimization problem is an unconstrained optimization problem (except for the simple bounds which can be satisfied easily). If the unconstrained optimization of the function $\Phi_k$ is repeated for a sequence of values of $r_k$, the solution may be brought to converge to the solution of the original problem. The main reason for the popularity of the penalty function method within this class of methods (where constraints are calculated explicitly) is because constrained NLP becomes an unconstrained NLP, implying that only one gradient (that of the $\Phi_k$) is required at each iteration; thus only one adjoint system has to be solved at each iteration. All other constrained NLP algorithms require the gradient of all active constraints at each iteration, and since for a path constrained optimal control problem, the number of active constraints and therefore the number of adjoint solves could be as large as the number of time steps of the forward problem (or more, if more than one path constraint is present), these algorithms are not practical for the production optimization problem. However, a key disadvantage of the penalty function method, which renders it impractical for the production optimization problem, is its inefficiency. Specifically, a large number of iterations for various values of $r_k$ are typically required for significant improvement of the objective function. The problem is more severe when the initial guess is close to the constraint boundaries and one or more constraints are active at the optimum, and this is often the case with production optimization problems.

Due to the fact that the constrained NLP methods require the gradients of all active constraints, and thus become very expensive for path constrained optimal control problems, one approach is to construct a single constraint from all the constraints, such that satisfying this single equivalent constraint ensures that all constraints will be satisfied. Such an approach is called constraint lumping [71]. With this approach, only two gradient calculations and therefore only two adjoint evaluations will be required at each iteration (one for the objective function and one for the equivalent constraint), which is a huge improvement compared to retaining all constraints. Various lumping

schemes are available in the literature [71, 66], with the following being commonly used for optimal control problems:

$$\sum_{n=0}^{N-1} \max\left[c^n(\mathbf{x}^{n+1},\mathbf{u}^n),0\right]=0$$
$$\sum_{n=0}^{N-1} \left\{\max\left[c^n(\mathbf{x}^{n+1},\mathbf{u}^n),0\right]\right\}^2=0 \tag{5.5}$$

The second approach is an improvement over the first because the gradient of the first is discontinuous, although the second representation is also more nonlinear. Another approach is to create a smooth approximation to the *max* function, as given by Jennings et al. [66]:

$$\sum_{n=0}^{N-1} h\left[c^n(\mathbf{x}^{n+1},\mathbf{u}^n)\right]<\varepsilon; \quad \text{where}$$
$$h\left(c^n\right)=\begin{cases}0 & \text{if} \quad c^n\le-\varepsilon\\ \dfrac{\left(c^n+\varepsilon\right)^2}{4\varepsilon} & \text{if} \quad -\varepsilon\le c^n\le\varepsilon\\ c^n & \text{if} \quad c^n\ge\varepsilon\end{cases} \tag{5.6}$$

The advantage of Equation (5.6) over the second equation of Equation (5.5) is the elimination of squaring, implying small deviations are penalized more heavily. In the algorithm proposed here, constraint lumping is applied in conjunction with the feasible direction algorithm. To our knowledge, this combination of algorithms has not been used previously for optimal control problems. Also the lumping scheme used, a relatively new procedure [72], is more rigorous compared to Equation (5.6) and has not been used for optimal control problems before. The feasible direction algorithm and the particular constraint lumping scheme used here are discussed in the next sections.

## 5.3. Feasible Direction Optimization Algorithm

The basic idea behind the method of feasible directions is to start from a feasible point (a point that satisfies all constraints) and move to a better point (with a lower objective function value) according to the iterative scheme [70]:

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \beta \mathbf{S}_i \tag{5.7}$$

Note that $\mathbf{u}$ here represents the entire set of controls $\mathbf{u}^0,..,\mathbf{u}^{N-1}$, and $\mathbf{u}_i$ is the starting point for the $i^{\text{th}}$ iteration, $\mathbf{S}_i$ is the search direction, $\beta$ is the step length, and $\mathbf{u}_{i+1}$ is the final point obtained at the end of the iteration. The search direction $\mathbf{S}_i$ is found such that the following two properties are satisfied (1) a small move in the direction violates no constraint, and (2) the value of the objective function is reduced in that direction. The iterative process is repeated until no search direction can be found satisfying both properties. The final iterate represents a constrained local minimum of the problem. A direction satisfying both above-mentioned properties is called a usable feasible direction [70].

To exemplify, consider the optimization problem depicted by Figure 5-1. The figure shows the objective function contours (orange lines, dot-dash) and three constraints (green lines, solid), which are functions of two control variables only. No state variables are present for simplicity. The blue dot labeled “optimum” depicts the constrained minimum of the problem, constrained by $c_3$. Thus only constraint $c_3$ is active at the optimum. The initial guess or starting point is at the intersection of constraints $c_1$ and $c_2$, meaning that $c_1$ and $c_2$ are active at the initial guess. The blue arrow (dashed) is the negative of the objective function gradient, and the purple arrows (dashed) are the negatives of the active constraint gradients. The thick orange lines represent the cone of feasibility at the initial guess, that is, any search direction within this cone will satisfy property 1. The pink arrow (solid) is a usable feasible direction, that is, a search direction that satisfies both properties 1 and 2. Such a direction at a

point $\mathbf{u}_i$ can be determined mathematically by a direction $\mathbf{S}_i$ that satisfies the following equations [70]:

$$\begin{aligned}
&\frac{d}{d\beta} J\left(\mathbf{u}_i + \beta \mathbf{S}_i\right)\Big|_{\beta=0} = \mathbf{S}_i^T \nabla J\left(\mathbf{u}_i\right) \le 0 \\
&\frac{d}{d\beta} c^n\left(\mathbf{u}_i + \beta \mathbf{S}_i\right)\Big|_{\beta=0} = \mathbf{S}_i^T \nabla c^n\left(\mathbf{u}_i\right) \le 0 \\
&\qquad\qquad \forall n = 0, \ldots, N-1
\end{aligned} \tag{5.8}$$

Here, $\nabla J\left(\mathbf{u}_i\right)$ and $\nabla c^n\left(\mathbf{u}_i\right)$ are the gradients of the objective function and the active constraints at point $\mathbf{u}_i$. It is possible to reduce the objective function $J$ at least by a small amount by taking a step length $\beta > 0$ along such a direction.

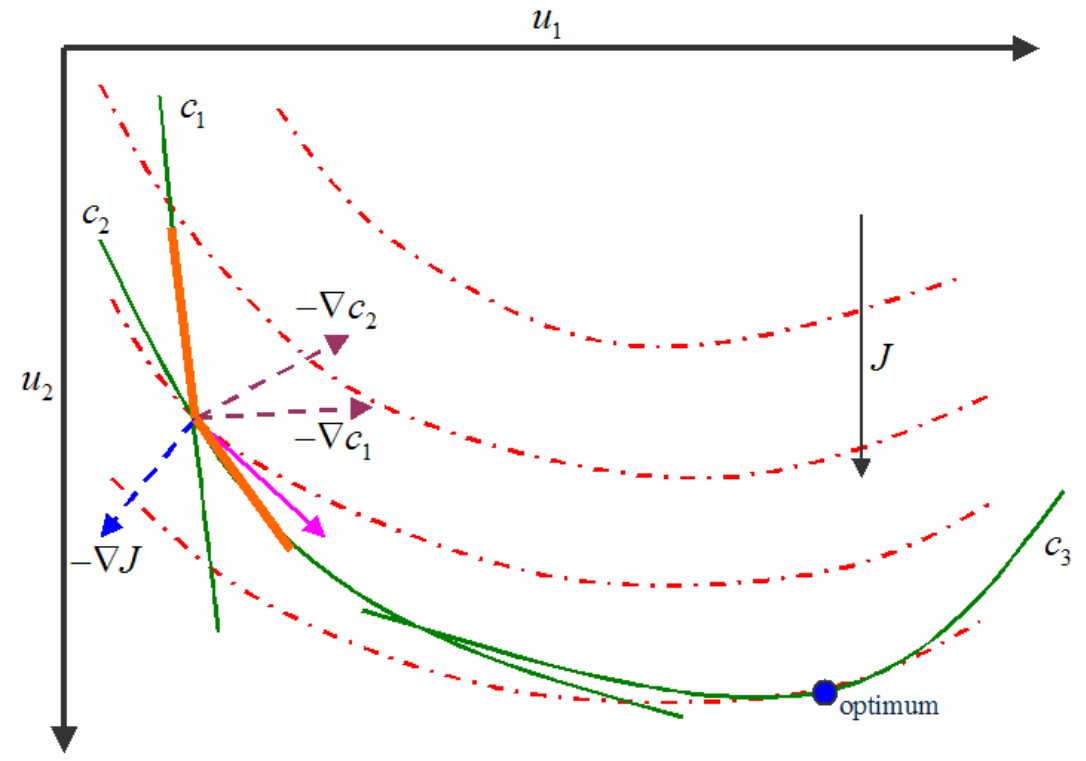


Figure 5-1 Schematic of a simple optimization problem with constraints

There are many different ways to determine a usable feasible direction, giving rise to different feasible direction algorithms. In the current work, the well-known Zoutendijk's method of feasible directions is used [70], and will therefore be discussed

here. In this method, the usable feasible direction at the current iterate is taken as the negative of the objective function gradient if the iterate lies in the interior of the feasible region. However, if it lies on the constraint boundary (or close to the boundary within some pre-set tolerance), a usable feasible direction that satisfies Equation (5.8), is found by solving a linear programming problem [70]:

$$\begin{aligned} &\min_{\mathbf{S}_i} \quad -\gamma \\ &\text{subject to:} \\ &\mathbf{S}_i^T \nabla c^n(\mathbf{u}_i) + \theta_n \gamma \le 0 \qquad n = 0, \ldots, p-1 \\ &\mathbf{S}_i^T \nabla J(\mathbf{u}_i) + \gamma \le 0 \\ &-1 \le s_{i,k} \le 1 \end{aligned} \tag{5.9}$$

Here, $s_{i,k}$ is the $k^{\text{th}}$ component of $\mathbf{S}_i$ (recall that $i$ designates iteration), and the first $p$ constraints are assumed to be active (or almost active) at point $\mathbf{u}_i$ (the constraints can always be renumbered to satisfy this requirement). Here $\gamma$ is taken as an additional design variable. Any value of $\gamma > 0$ would provide a usable feasible direction $\mathbf{S}_i$. The maximum value of $\gamma$ gives the best direction $\mathbf{S}_i$ that makes the value of $\mathbf{S}_i^T \nabla J(\mathbf{u}_i)$ maximally negative and the values of $\mathbf{S}_i^T \nabla c^n(\mathbf{u}_i)$ as negative as possible simultaneously. In other words, the maximum value of $\gamma$ makes the direction $\mathbf{S}_i$ steer away from the active nonlinear constraints. Different values of $\theta_n$ for different constraints allow us to give more importance to certain constraint boundaries as compared to others. For more details, refer to Rao [70].

The feasible direction algorithm is useful when the initial guess point is at (or close to) constraint boundaries and the steepest descent direction (negative objective function gradient direction) is pointing away from the feasible region. This is often the case with production optimization problems. In such a case, first order algorithms (like the steepest descent algorithm) or even quasi-Newton algorithms would provide search

directions moving along which would violate one or more constraints, thereby providing infeasible iterates.

### 5.4. Approximate Feasible Direction Algorithm

Zoutendijk's method of feasible directions [70], like all other constrained NLP algorithms, requires the gradients of all active constraints. As seen in Equation (5.9), these gradients are required to calculate the feasible direction. However, as mentioned earlier, this is not practical for an optimal control problem with path constraints due to the excessive number of adjoint evaluations required to calculate these gradients. One approach to alleviate this problem is to apply constraint lumping to create one equivalent constraint from all active constraints, with the benefit that only two adjoint solutions will be required at each iteration, one for the objective function and one for the equivalent constraint. The particular lumping scheme used in this work is given by Liu et al. [72]. It is essentially a smooth differentiable approximation of the *max* function, but is more rigorous compared to Equation (5.6). In their work, the lumping scheme is used within a penalty function method, whereas in this work we apply it in conjunction with the feasible direction method, which leads to a different interpretation.

Consider the discontinuous unit-step function and its approximation, the sigmoid function, given by the following equations:

$$\begin{aligned} \sigma(y) &= \begin{cases} 1 & \text{if } y > 0 \\ 0 & \text{if } y \le 0 \end{cases} \\ s(y,\alpha) &= \left\{1+\exp(-\alpha y)\right\}^{-1} \; \forall \alpha > 0 \end{aligned} \tag{5.10}$$

The *max* function is an integral of the unit-step function and is given by the following equation:

$$\max\{x,0\} = \int_{-\infty}^{x} \sigma(y)dy \tag{5.11}$$

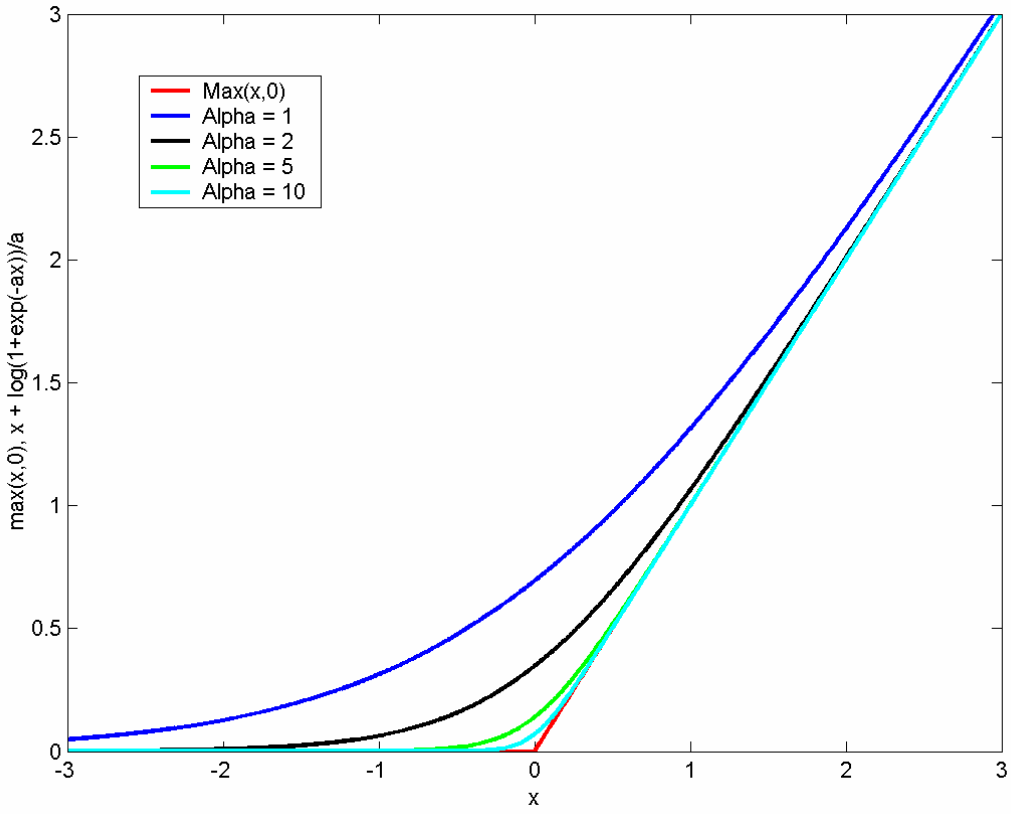


Figure 5-2 The *max* function and its approximations for various values of $\alpha$

Substituting $s(y,\alpha)$ for $\sigma(y)$ in the expression above, it can be shown that the following equation approximates the *max* function [72]:

$$p(x,\alpha) = \int_{-\infty}^{x} s(y,\alpha)dy = x + \frac{1}{\alpha}\log\{1+\exp(-\alpha x)\} \tag{5.12}$$

The function $p(x,\alpha)$ has infinitely many continuous derivatives. Some other properties of the function $p(x,\alpha)$ relevant to its application with the feasible direction algorithm are as follows:

$$\begin{aligned} &p(x,\alpha) > \max\{x,0\} && \forall x \in R \\ &\lim_{|x|\to\infty}\{p(x,\alpha) - \max(x,0)\} = 0 && \forall \alpha > 0 \\ &\lim_{\alpha\to\infty}\{p(x,\alpha) - \max(x,0)\} = 0 && \forall x \in R \end{aligned} \tag{5.13}$$

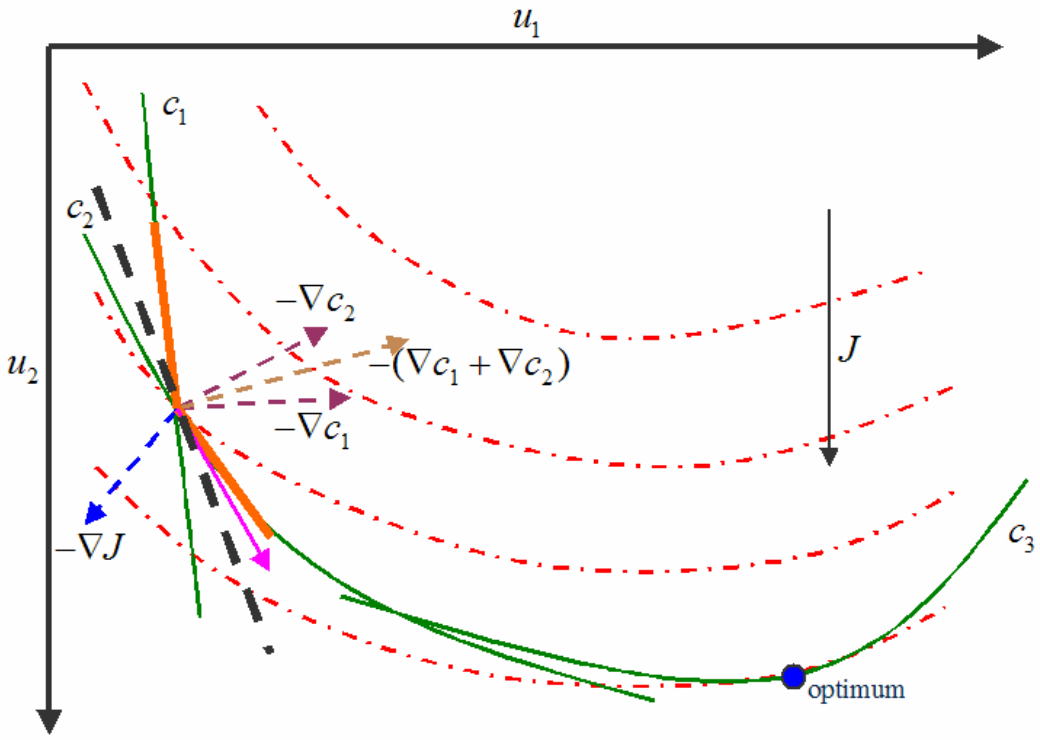


Figure 5-3 Schematic of a simple optimization problem with constraints, illustrating an approximate feasible direction

Due to the above properties, the function $p(x,\alpha)$ can be used as an approximation of the *max* function for constraint lumping. This circumvents the main disadvantage of the *max* function, which is its non-differentiability (this makes it difficult to implement the *max* function with gradient-based algorithms). However, due to the infinite differentiability property mentioned above, $p\left\{c^n\left(\mathbf{x}^{n+1},\mathbf{u}^n\right),\alpha\right\}$ will be as many times differentiable as $c^n\left(\mathbf{x}^{n+1},\mathbf{u}^n\right)$. Figure 5-2 shows the *max* function and its approximation with $p(x,\alpha)$ for various values of $\alpha$. The equivalent constraint replacing the *max* constraint lumping scheme for the path constraints of Equation (5.1) is given as:

$$C=\sum_{n=0}^{N-1}\left[c^n+\frac{1}{\alpha}\log\left\{1+\exp\left(-\alpha c^n\right)\right\}\right]\le\frac{\log 2}{\alpha}\quad\forall\alpha>0 \tag{5.14}$$

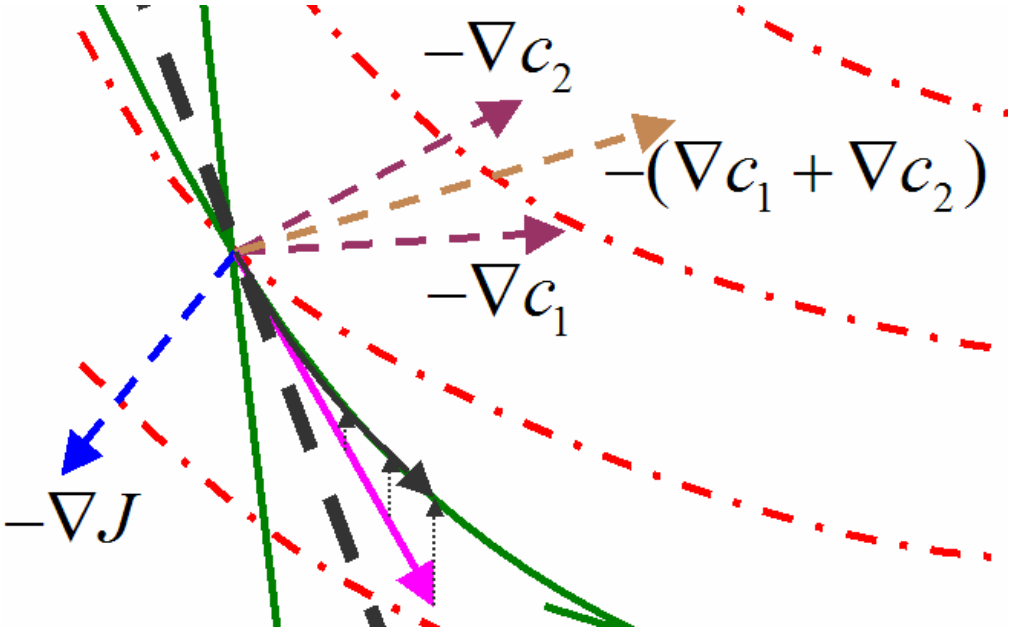

Figure 5-4 Zoomed in version of the above schematic

The $\log 2/\alpha$ term appears in the above equation because $p\left(c^n = 0, \alpha\right) = \log 2/\alpha$ and $p\left(c^n, \alpha\right)$ increases monotonically with $c^n$. With this definition, the optimal control problem equivalent of Equation (5.1) is given by Equation (5.15) below. The nonlinear path constraint has been replaced by the single integral constraint *C*. The bounds on the controls are still present, but they can be easily satisfied using standard techniques like gradient projection.

$$
\begin{aligned}
&\max_{\mathbf{u}^n}\left[ J = \phi\left[\mathbf{x}^N\right] + \sum_{n=0}^{N-1} L^n\left(\mathbf{x}^{n+1}, \mathbf{u}^n\right)\right] \forall n \in (0,..,N-1) \\
&\text{subject to:} \\
&g^n(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{u}^n) = 0 \qquad \forall n \in (0,..,N-1) \\
&\mathbf{x}^0 = \mathbf{x}_0 \qquad \text{(Initial Condition)} \\
&C = \sum_{n=0}^{N-1}\left[ c^n + \frac{1}{\alpha}\log\left\{1 + \exp\left(-\alpha c^n\right)\right\}\right] \le \frac{\log 2}{\alpha} \\
&LB \le \mathbf{u}^n \le UB \qquad \forall n \in (0,..,N-1)
\end{aligned}
\tag{5.15}
$$

From the perspective of the feasible gradient algorithm, since the function $p\left\{c^n\left(\mathbf{x}^{n+1}, \mathbf{u}^n\right), \alpha\right\}$ is an approximation of $\max\left\{c^n\left(\mathbf{x}^{n+1}, \mathbf{u}^n\right), 0\right\}$, it is clear that the

equivalent constraint *C* is essentially a sum of the active constraints, for large enough $\alpha$. Therefore, the gradient of the equivalent constraint *C* is the sum of the gradients of the active constraints. This is demonstrated in Figure 5-3, where the dashed brown arrow is the sum of the gradients of the active constraints $c_1$ and $c_2$, and these gradients themselves are depicted by the dashed purple arrows. Therefore, the implication of using this particular constraint lumping as opposed to directly solving the original problem is that, instead of obtaining the gradients of all the active constraints individually, only a single gradient direction is obtained, which is the sum of the gradients of all active constraints. With only this single direction, the apparent feasibility cone is given by the thick gray line. This apparent feasibility cone will always be equal to or wider than the true feasibility cone (thick orange line). Again, a linear program can be solved to determine the feasible direction:

$$\begin{aligned} &\min_{\mathbf{S}_i} \quad -\gamma \\ &\text{subject to:} \\ &\mathbf{S}_i^T \nabla C\left(\mathbf{u}_i\right) + \theta\gamma \le 0 \\ &\mathbf{S}_i^T \nabla J\left(\mathbf{u}_i\right) + \gamma \le 0 \\ &-1 \le s_{i,k} \le 1 \end{aligned} \tag{5.16}$$

Note that, because the apparent feasibility cone may be wider than the true feasibility cone, the feasible direction obtained may not be actually feasible. For example, using Equation (5.16), a direction such as the solid pink arrow (Figure 5-3) may be obtained as the feasible direction. Although this direction is within the apparent feasibility cone, it is outside the true feasibility cone, and is therefore not truly a feasible direction. Moving in this direction even infinitesimally would result in the violation of constraint $c_2$ (but not $c_1$). This direction is therefore called an approximate feasible direction. However, this direction will usually be better than just the steepest descent direction (negative objective function gradient). For example, as seen in Figure 5-3, moving in the steepest descent direction violates both active constraints.

In order to solve the above problem of approximate feasibility, a feasible line search algorithm is employed. The key idea behind the feasible line search algorithm is to implement the path constraints within the forward model and modify the search direction within the forward model if the path constraints are violated. The approximate feasible direction is thus projected onto the infeasible active constraints during line-search by solving the constraints during the forward simulation. Note that projecting this direction onto the infeasible active constraints during line-search is equivalent to performing a "curved" line search along the infeasible active constraints, as seen in Figure 5-4. Gradient information from previous iterations and our knowledge of the dynamic system can be used to determine which controls need to be modified to satisfy an infeasible constraint. For example, if a path constraint such as a maximum injection rate constraint is violated at a given time step of the forward simulation, the controls associated with the injectors at that time step will have the maximum influence on the constraint, and should therefore be modified to satisfy the constraint. Note that there could be many possible choices of controls or combinations thereof that can be modified to satisfy a constraint, and it is not clear at the moment if a particular "best" strategy exists that could be employed to choose the right controls.

In the current work, a maximum total water injection constraint and a maximum total liquid production constraint have been implemented. For the maximum total water injection rate constraint, if the constraint is violated at a given time step, controls associated with all the injectors (BHPs) at that time step are modified to satisfy the constraint. Similarly, for the maximum total liquid production constraint, the producer BHPs are modified to satisfy the constraint. In the current implementation, a simple, easy to implement iterative approach is used to determine the modified controls if the path constraints are violated at a given time step. The approach assumes that a linear relationship exists between the injection rate of an injector (or liquid production rate of producer) and the pressure difference between its BHP and well block pressure, as depicted by the following equation:

$$p_w^{tgt} = p_b^{n+1} - \omega \frac{q^{tgt}}{q^{cur}} \left( p_b^{n+1} - p_w^{cur} \right) \tag{5.17}$$

To clarify, after solving the forward model at a given time step with the values of the controls as provided by the optimization algorithm (approximate feasible direction algorithm in this case), if a constraint is violated (i.e., $q^{tgt} < q^{cur}$), new values of the controls to be modified ($p_w^{tgt}$) are obtained with the above equation using the current values of the controls ($p_w^{cur}$), and the forward model is solved again at the same time step with the new control values. The process is repeated until the constraint is satisfied. $\omega$ is a relaxation factor used to accelerate convergence. This approach is certainly not the most efficient, and the best approach would be to solve the violated constraints together with the dynamic system.

The main benefits of the feasible line search algorithm are that all iterates obtained are always feasible, implying that any iterate can be considered a useful solution, and large step sizes are possible during the feasible line search, leading to significant reductions in forward model evaluations and possibly also the objective function.

In order to account for the bounds on the controls, a projected gradient algorithm (with the approximate feasible direction as the search direction) is used. Refer to Kelley [73] for more details about gradient projection. The simulator used in this work is the General Purpose Research Simulator (GPRS) developed at Stanford [11], and the adjoint models are built directly from the simulator code, as described in Chapter 2 and [10].

## 5.5. Example 1 – Horizontal Smart Wells

The first case is a simple example adapted from Brouwer and Jansen [34] that effectively demonstrates the applicability of the proposed algorithm to smart well control with nonlinear path constraints. The schematic of the reservoir and well

configuration is shown in Figure 2-3. The model description is the same as in Chapter 2 and is therefore not described here.

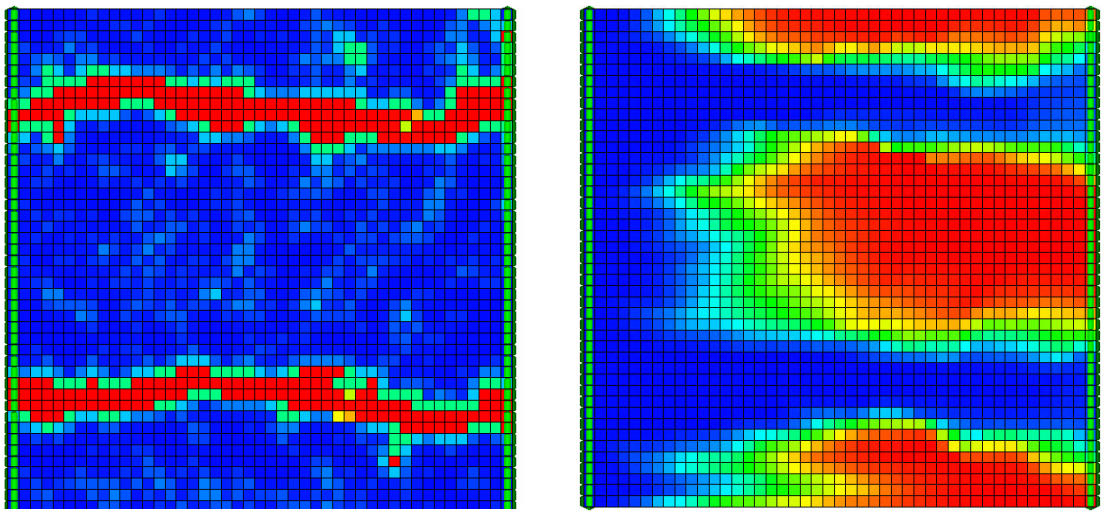

Figure 5-5 Permeability field for first example on left, and final oil saturation for uncontrolled case on right

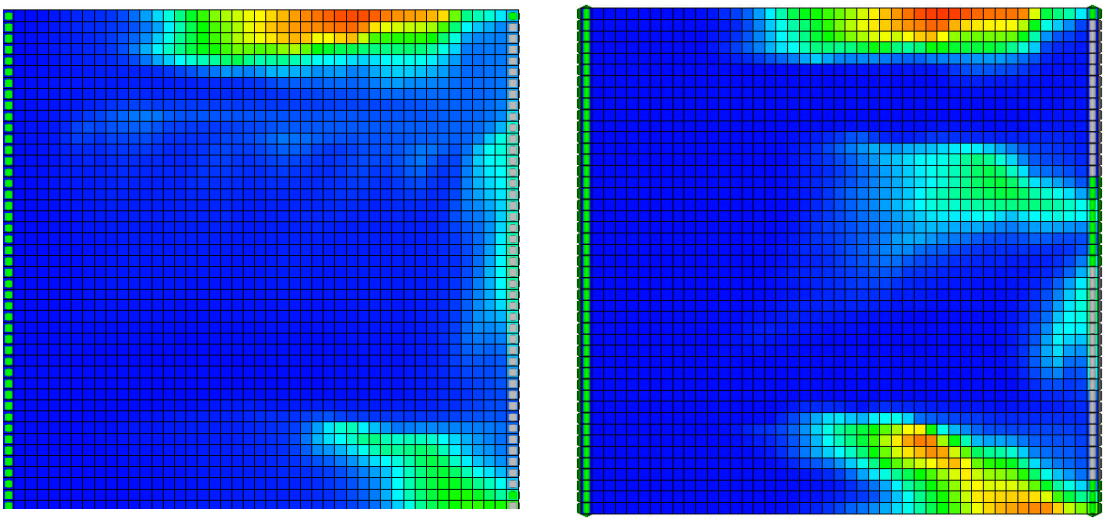

Figure 5-6 Final oil saturation for rate controlled case on left, and for BHP controlled case on right

For purposes of optimization, the injector and producer segments are placed under BHP control. Everything else is the same as in Chapter 2. Because the injector segments were under rate control in Chapter 2, the maximum injection rate constraint of 2710 STBD was a linear constraint. However, because the controls are the BHPs of segments in this case, the same constraint becomes a nonlinear path constraint. Thus, comparing against our earlier results, this case demonstrates the validity and

effectiveness of the proposed approach to handle nonlinear path constraints. The base case is also the same as before.

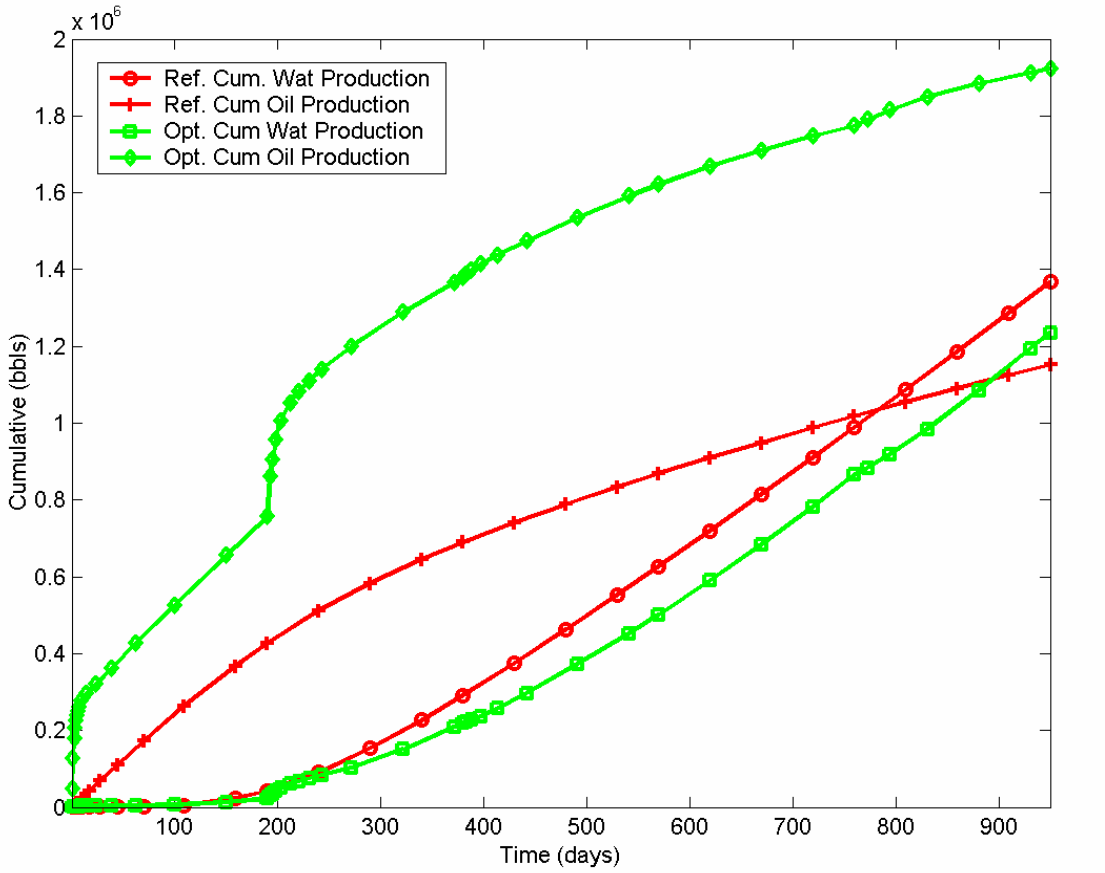


Figure 5-7 Cumulative water and oil production for uncontrolled reference case and optimized case

Starting from an initial oil saturation of 100% throughout the reservoir, Figure 5-5 (right) shows the final oil saturations for the uncontrolled case obtained after 950 days of production. Figure 5-6 shows the final oil saturations after 950 days for the optimized cases, the left image with the injector segments under rate control (from Chapter 2), and the right image with the injector segments under BHP control (performed with the proposed algorithm). As discussed above, rate controlled injectors correspond to the injection rate constraint being a linear path constraint, and for BHP controlled injectors, the rate constraint becomes nonlinear. It is clear that the optimization leads to a large improvement in the sweep efficiency for both cases, and the proposed algorithm with BHP controlled injectors performs almost as well as the

original algorithm [10] with linear constraints only, validating the effectiveness of the approach. The optimization leads to an increase in NPV of approximately 100%. Figure 5-7 shows that there is a substantial increase in cumulative oil production (70%), attributed to the better sweep, and a slight decrease in water production (6%) after the optimization process. The optimization process required 4 iterations and the total number of simulations (including adjoint simulations) required for the optimization was around 15.

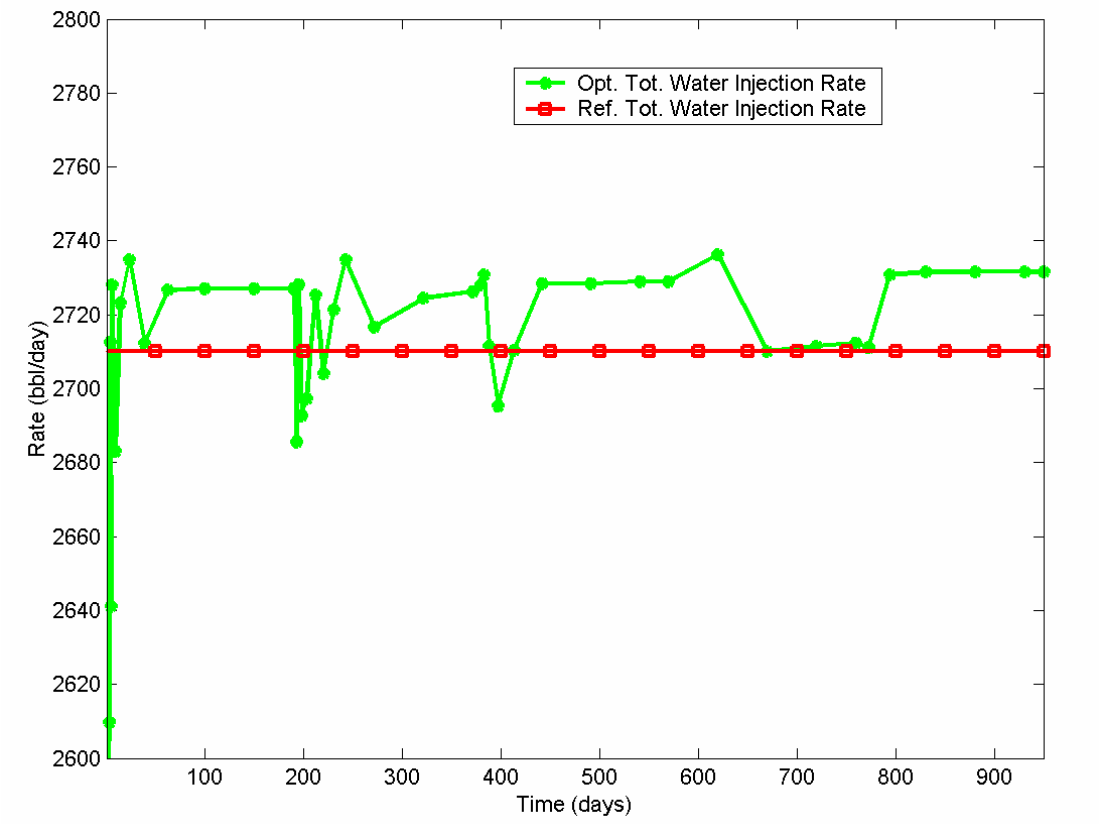


Figure 5-8 Maximum water injection constraint before and after optimization

Figure 5-8 shows the injection rates for the reference case and the optimized case (with BHP controlled injectors). It is clear that the constraint (2710 STBD) is satisfied to within 1% tolerance after optimization. Note that even after the optimization, the water injection rate remains near the maximum for most of the time, thus implying that the optimization essentially results in redistribution of the injected water among the injection segments.

As in Chapter 2, the reasons behind the better sweep in the optimized case can be explained by analyzing the optimized trajectories of the controls as seen in Figure 5-9. The *y*-axis of Figure 5-9 corresponds to 45 segments (injectors on the left and producers on the right) and the *x*-axis corresponds to the 5 control steps. The color scale corresponds to BHPs of the segments. It is obvious that the injector segments completed in or near the high permeability streaks are shut-in most of the time, as they would otherwise force water to move very quickly toward the producers, resulting in early breakthrough and thus poor sweep. For the producers, we again observe that the segments completed in or near the high permeability streaks are shut most of the time, in order to force the injected water into the low permeability regions to improve sweep.

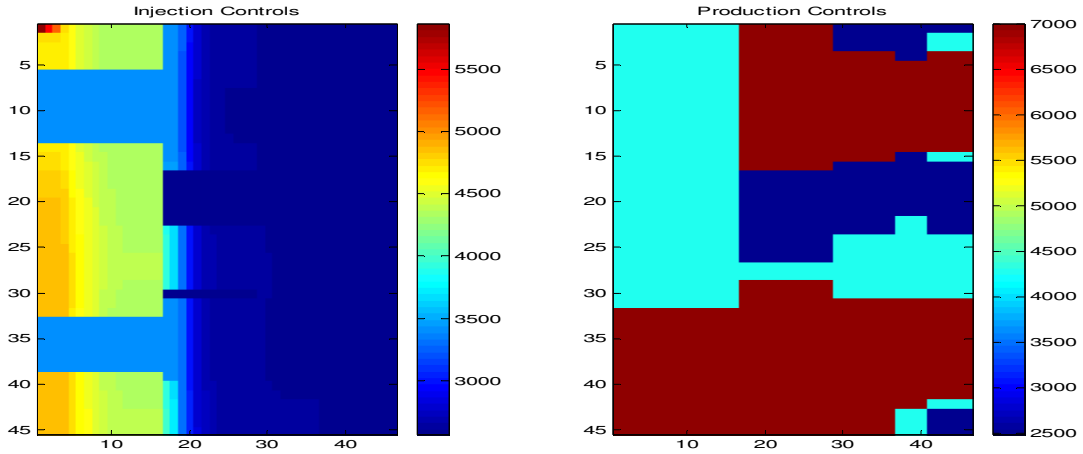


Figure 5-9 Control trajectories for injectors and producers after optimization (*x*-axis represents the control steps, and *y*-axis represents the 45 injector and producer segments)

## 5.6. Example 2 – Arab-D Formation, Ghawar Reservoir

The second example studied is a more realistic example adapted from Yeten [6] and Sengul et al. [74]. The simulation model is a conceptual representation of a small portion of the Arab-D formation of the Ghawar reservoir (see Figure 5-10 left). The 3D simulation model, populated with the initial oil distribution, is shown in Figure 5-11 (left). The red blocks indicate oil and blue blocks indicate water, while the blocks in between indicate the transition zone. The reservoir model consists of $25 \times 33 \times 10$ cells with grid sizes in the *x* and *y* directions of 200 feet. The thickness of each layer

varies, and properties are provided in [6]. The movable oil originally in place is around 18 MMSTB. The model has aquifer support along the east flank. The other boundaries were modeled as no-flow.

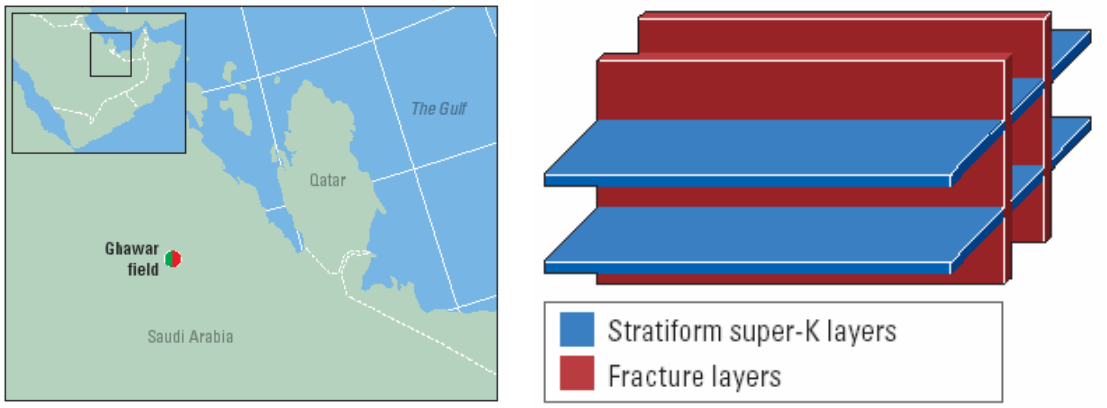


Figure 5-10 The Ghawar oil field with small rectangle depicting area under study on the left, orientation of fractures and Super – K layers in simulation grid on the right [74]

This field is a naturally fractured carbonate reservoir. Fractures act as the fastest fluid flow paths within the reservoir. The matrix also has reasonable permeability and contributes significantly to fluid flow. Two distinct fracture distributions are identified within the field. Here they will be referred to as fractures and stratiform "Super - K" layers. The fractures are modeled as vertical high permeability zones, oriented along the east-west plane, cutting all layers from top to bottom. The stratiform Super - K layers are modeled as thin layers with high permeability. Figure 5-10 (right) shows how the fractures and the stratiform Super - K layers are oriented. The refined grids in the *y* direction in Figure 5-11 (left) represent the vertical fractures, and the thin layers (5 and 9) represent the Super - K layers. Additional properties are provided in [6].

Because the field is operated above the bubble point pressure, the simulation model only includes oil and water phases. A tri-lateral production well is completed in the second layer of the simulation model (Figure 5-11 right). All laterals as well as the main-bore are horizontal. The heel of the main-bore is highlighted with a full white circle on this plot. The branch closest to the heel of the well will be referred to as Branch 1, the one just below it will be referred to as Branch 2, and the last one will be

referred to as Branch 3, as shown in Figure 5-11 (right). Note that Branch 2 intersects a fracture and Branch 1 is very close to a fracture. Branches 1 and 2 are about 2000 ft long and Branch 3 is about 3000 ft long. The branches are spaced approximately 1400 ft apart from each other, and all have open-hole completions. The laterals are fully perforated (no partial perforation) and the main-bore is not perforated.

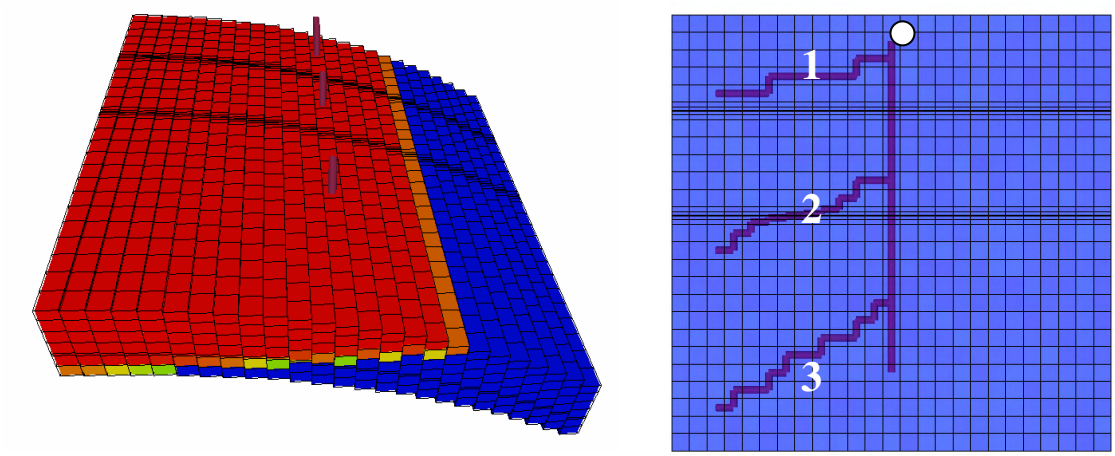


Figure 5-11 3D simulation model of the Ghawar example on the left, and the tri-lateral well on the right

The simulation model was run for 1800 days (approximately 5 years). Production is subject to a maximum liquid production constraint of 6000 STBD, and the controls are the BHPs of each lateral, thereby making the liquid production constraint a nonlinear path constraint. The minimum allowable BHP at the heel of the well was set at 2600 psi (equivalent approximately to a WHP of 250 psi, as in [6]). The objective is to maximize the cumulative oil production of the reservoir. The base case is an uncontrolled case producing at the maximum liquid rate (6000 STBD). Uncontrolled implies that the BHPs of the laterals are not controlled directly, and they adjust themselves according to the production rate and the BHP at the heel of the well. Because the GPRS version used in this work did not have downhole choke models implemented, the tri-lateral well was actually modeled as three separate horizontal wells. To maintain approximate consistency with Yeten's model [6], for the uncontrolled case the BHPs at the junctions of the laterals and the main-bore were

specified based on the ECLIPSE [75] simulation results generated in [6]. This entailed specification of time-varying BHP for each lateral. The resulting oil rates and watercuts generated by GPRS are close to the original ECLIPSE results of Yeten [6], indicating that our base case is consistent with the earlier model.

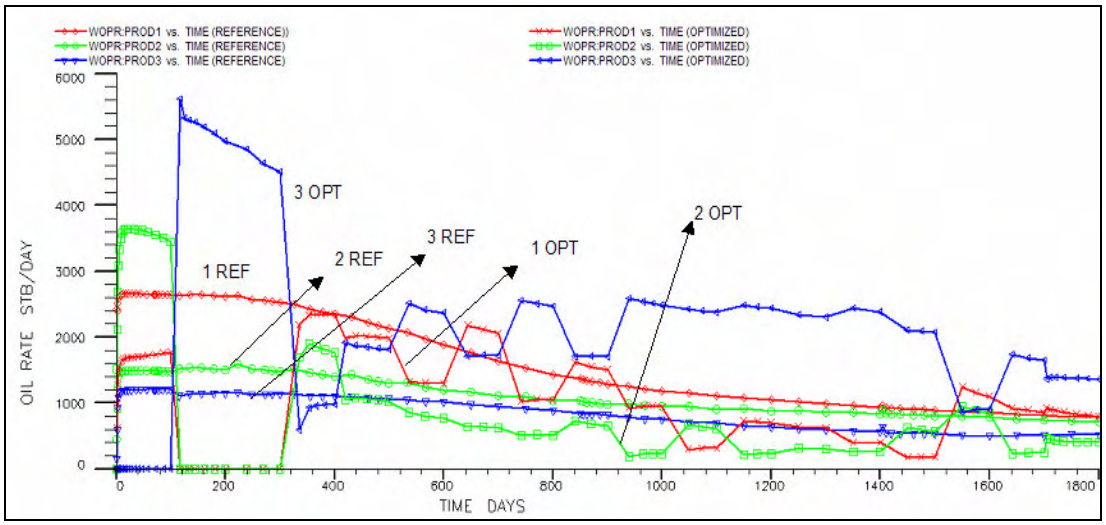


Figure 5-12 Oil production rates for the three branches for the uncontrolled and optimized cases

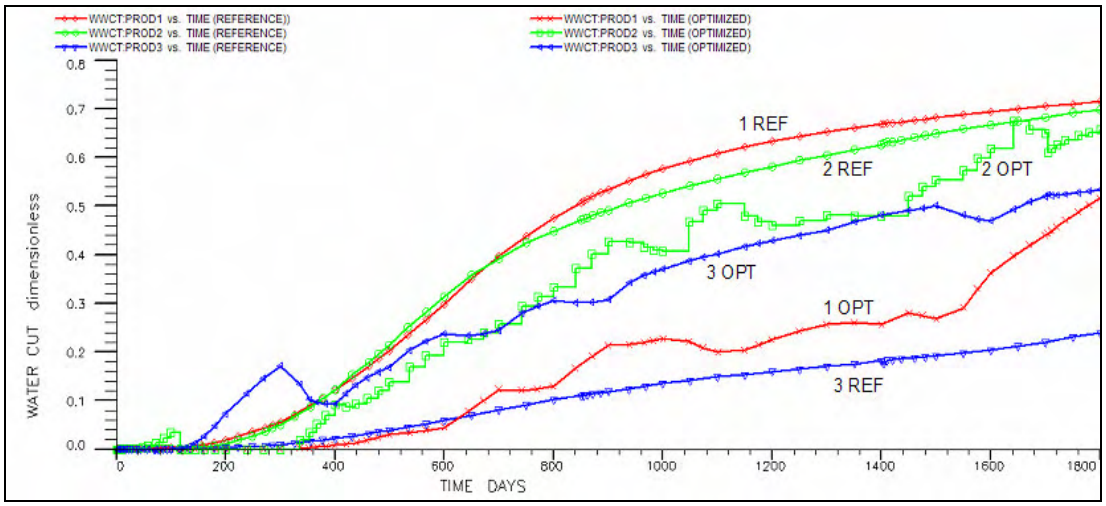


Figure 5-13 Watercuts for the three branches for the uncontrolled and optimized cases

Figure 5-12 shows the oil production profiles for individual branches. As explained in [6], the resulting production is unbalanced. In the uncontrolled (reference) case, Branch 1, which is closest to the heel of the well, produces more oil than the other two

branches. Branch 2 produces significantly more than Branch 3, especially for the early times before the water breaks through, due to its intersection with a fracture.

Figure 5-13 presents the watercut for each branch. The watercut of Branch 3 (reference) is much less compared to the others, likely due to the proximity of Branches 1 and 2 to the fractures and Super – K layers, which act as fast conduits to the aquifer. This results in unbalanced production and unbalanced sweep (Figure 5-14 left) which are detrimental to overall recovery.

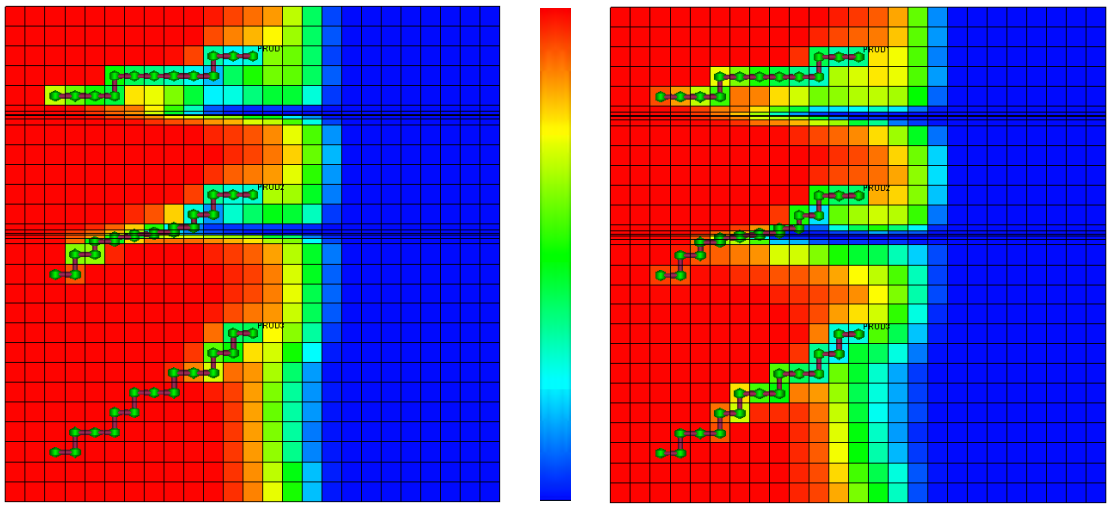

Figure 5-14 Final oil saturation for layer 2 for the uncontrolled case (left) and the optimized case (right)

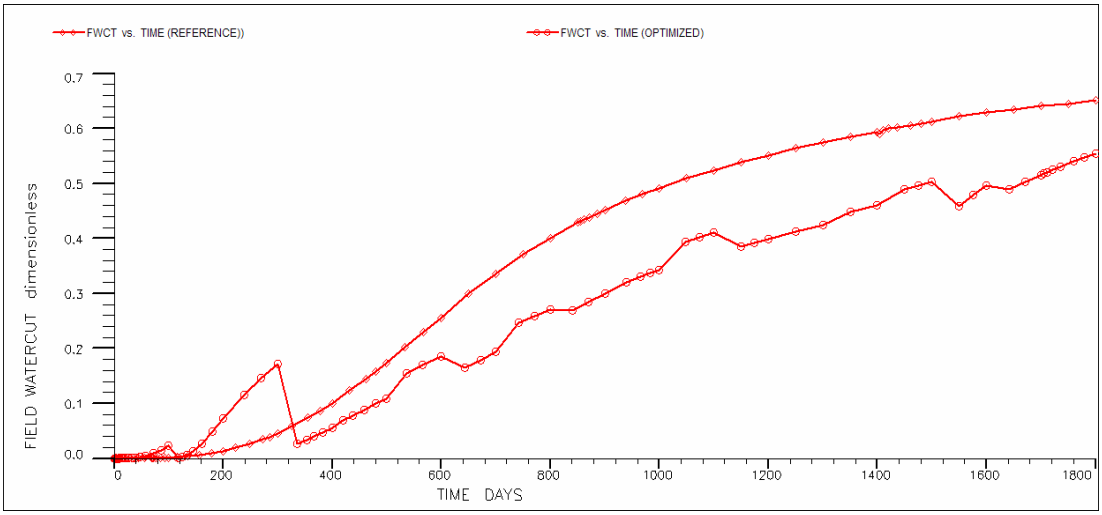


Figure 5-15 Field watercut for the uncontrolled and optimized cases

Figure 5-12 also shows the oil production profiles after optimization with the proposed algorithm. The 1800 days of production was divided into 18 controls steps of 100 days each, resulting in a total of 18x3 = 54 controls. As would be expected, the algorithm allocates more production to Branch 3 than the other branches. It can also be seen that Branch 2 has been allocated the least amount of production due to its direct connection to a fracture.

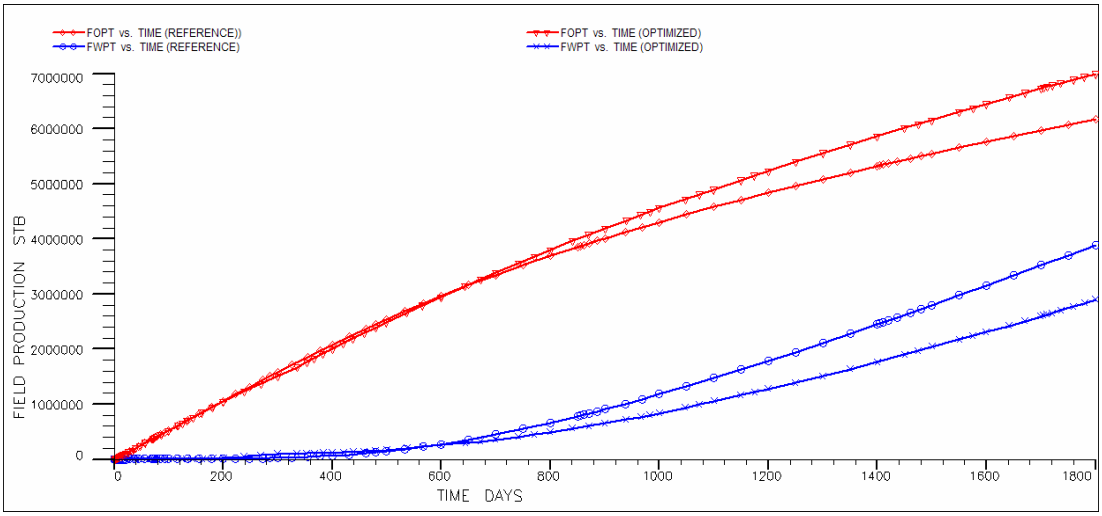


Figure 5-16 Cumulative oil and water production for the uncontrolled and optimized cases

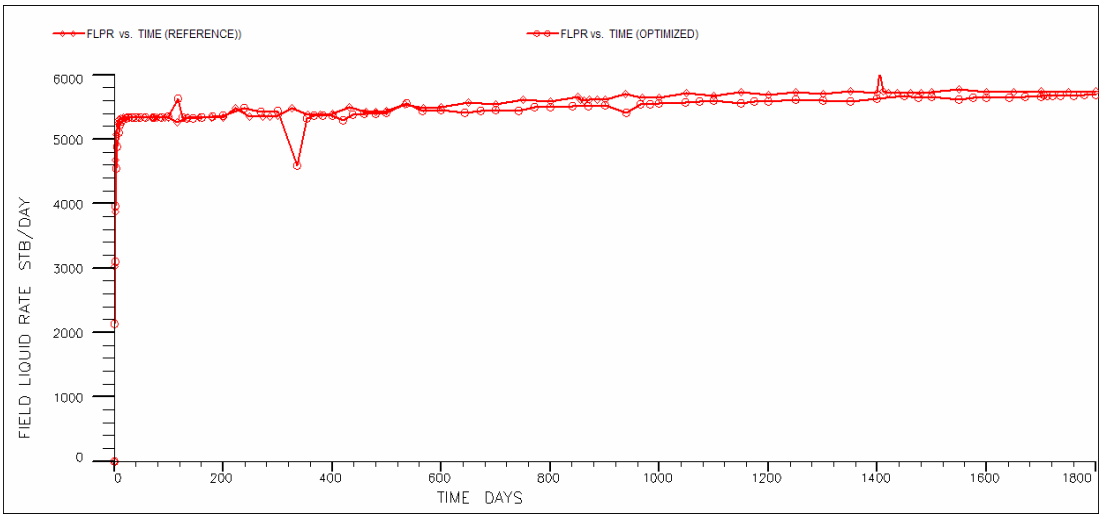


Figure 5-17 Maximum liquid production constraint for the uncontrolled and optimized cases

Figure 5-13 also shows the watercut profiles of the branches after optimization. The water breaks through in Branch 3 earlier. This water comes from the matrix and its watercut does not increase as rapidly as in the other branches. Therefore the breakthrough in Branch 3 does not affect the overall performance as much as the breakthrough in other branches. Figure 5-14 (right) also shows that the final sweep pattern obtained after optimization is more balanced, resulting in more oil being produced.

Figure 5-15 shows the field watercuts for the uncontrolled and optimized cases. It is clear that because there is a maximum liquid production constraint (which is essentially active in the uncontrolled case), the only way to increase cumulative oil production is to reduce the field watercut. It is observed from Figure 5-15 that the field watercut is higher for the optimized case for about 300 days, after which it reduces below the base case. The final watercut after 1800 days is reduced from 0.66 to 0.54, resulting in an increase in cumulative oil production of about 16% over the base case, as seen in Figure 5-16. The liquid production rates for the uncontrolled and the optimized cases are plotted in Figure 5-17. The optimization clearly satisfies the maximum constraint of 6000 STBD at all times, again validating the effectiveness of the proposed algorithm. The total number of iterations of the optimization algorithm was 11, and the total number of simulations required including adjoint simulations was 68. Yeten [6] used a conjugate gradient algorithm with numerical gradients for the optimization, with ECLIPSE as the simulator, and he also found a similar improvement in cumulative oil production. However, the number of simulations required with his approach will in general be much more due to the use of numerical gradients.

## 5.7. Summary

In this chapter, an approximate feasible direction algorithm combined with a feasible line search was proposed to handle production optimization problems with nonlinear path inequality constraints, with the following major benefits:

1. Due to constraint lumping, only two adjoint evaluations are required at each iteration of the optimization algorithm, which is one of the reasons behind the efficiency of the algorithm.

2. All iterates obtained are always feasible, implying that the optimization may be stopped at any iteration and the final iterate can be considered a useful solution.

3. Large step sizes are possible during the modified line search, leading to significant reductions in forward model evaluations during the line search, and may also result in significant reduction of the objective function.

4. Problems associated with starting close to the boundary of the feasible region, which may limit the effectiveness of penalty function methods, are avoided. Such starting points often occur in production optimization problems.

The effectiveness and applicability of the algorithm was demonstrated through two examples: the first was a dynamic waterflood optimization problem with a maximum injection rate constraint, and the second was a tri-lateral well optimization problem with aquifer support and a maximum liquid production rate constraint. Both optimizations resulted in significant improvement in the objective functions (NPV and cumulative oil production), implying that model-based optimization has considerable potential for practical reservoir management.

# Chapter 6

# 6. Kernel PCA for Parameterizing Geology

As described in Chapter 3 and Chapter 4, a continuous and differentiable parameterization of input random fields of simulation or geological models relating the output realizations and input parameters is required because such a parameterization may be used in conjunction with, or as a subset of, other gradient-based algorithms that constitute the reservoir modeling and management workflow, such as history matching or uncertainty propagation algorithms. The emphasis here on gradient-based algorithms is due to their efficiency compared to stochastic algorithms, because in general, gradient-based algorithms require many fewer evaluations of the reservoir flow simulation model compared to stochastic algorithms. This is essential for many practical applications, because a single evaluation of a large-scale reservoir simulation model can require many hours. Furthermore, as seen in earlier chapters, efficiency has become even more critical with the advent of closed-loop reservoir management [8, 16], which requires continuous realtime application of various algorithms such as model updating, uncertainty propagation and optimization.

To elaborate, a differentiable mathematical parameterization of the form $\mathbf{y} = f(\boldsymbol{\xi})$ is desirable, where vector $\mathbf{y}$ is a realization of a discrete random field, $\mathbf{y} \in R^{N_C}$, ($N_C$ = number of cells in geological model) and vector $\boldsymbol{\xi}$ (input parameters), of dimension much smaller than $N_C$, is a set of independent random variables with a specified distribution (for example standard Gaussian). Such a mathematical model can be used for geostatistical simulation just as standard geostatistical algorithms, because by drawing a random vector $\boldsymbol{\xi}$ from the specified distribution, a realization $\mathbf{y}$ of the given random field can be generated. In comparison, a key issue with both the traditional two-point [63] and the more recent multi-point geostatistical algorithms

[76, 57, 77] is that the algorithms do not provide a differentiable functional relationship between the output realizations (of the simulated random field) and the input parameters of the algorithms. Further, the input parameterization is not very general or flexible, for example, a parameterization in terms of a single random variable (random seed) is used in most geostatistical algorithms to generate the realizations of the random field.

As seen in Chapter 3, the Karhunen-Loeve (K-L) expansion or linear principal component analysis (PCA) was used within the closed-loop for a differentiable parameterization of subsurface properties in terms of a small set of independent random variables. However, the K-L expansion has two major drawbacks limiting its application to practical geological or simulation models. First, as is the case with standard two-point geostatistical algorithms, it only preserves the covariance of the random field (two-point statistics), and therefore is suitable only for multi-Gaussian random fields. It cannot be applied to model complex geological structures such as channels. Second, the K-L expansion requires a computationally expensive eigen decomposition of a large covariance matrix of size $N_C \times N_C$, and this is exceedingly demanding for large-scale simulation models even with current computational capability. To our knowledge, there are no differentiable mathematical models applied in the petroleum industry for parameterizing non-Gaussian random fields (multi-point statistics) that are capable of representing complex geological structures.

Both problems of the K-L expansion discussed above can be solved elegantly with a recently developed theory from the field of neural computing and pattern recognition, known as kernel principal component analysis (KPCA), which is a nonlinear form of PCA [22, 78]. The basic idea is that the kernel method can be applied to create different nonlinear versions of any algorithm (such as the K-L expansion) that can be written exclusively in terms of dot products [22]. Kernel PCA has proven to be a powerful tool as a nonlinear feature extractor for classification algorithms [79] from

high dimensional data sets, which is similar to what multi-point geostatistical algorithms attempt to accomplish using training images.

In this chapter, we first apply the kernel formulation of the eigenvalue problem associated with the standard K-L expansion (this corresponds to a polynomial kernel of order one), which results in a much more efficient representation of the K-L expansion compared to a direct solution of the standard eigenvalue problem. Using the kernel formulation of the eigenvalue problem, instead of performing an eigen decomposition of an $N_C \times N_C$ covariance matrix, an eigen decomposition of a different matrix, the so-called kernel matrix of dimension $N_R \times N_R$ ($N_R$ is the number of realizations required to achieve a converged covariance matrix), is performed. This can be accomplished very efficiently, as $N_R$ is usually quite small compared to $N_C$.

Next, kernel principal component analysis with high order polynomial kernels is applied to preserve multi-point statistics of non-Gaussian random fields, thereby creating a differentiable mathematical model capable of simulating random fields representing complex geological structures. The kernel PCA parameterization is not created from analytical multi-point statistical descriptions but rather from realizations of the random field generated using existing multi-point geostatistical algorithms. The kernel PCA parameterization thus provides a differentiable model that enables the use of gradient-based algorithms for a variety of applications. Kernel PCA is a nonlinear generalization of the K-L expansion, and the basic idea is that instead of performing the K-L expansion in the input space $R^{N_C}$ of the original random field, the K-L expansion is performed in a possibly high order feature space $F$, which is nonlinearly related to the input space. Application of $d^{th}$ order polynomial kernels implies that the feature space $F$ is a $d^{th}$ order product space of the original input space $R^{N_C}$. Therefore, a K-L expansion in the feature space $F$ results in preserving moments up to order $2d$ ($2d$-point statistics) of the original random field in $R^{N_C}$.

Because the K-L expansion is performed in the feature space, the resulting realizations lie in the feature space, and therefore an appropriate "pre-image problem" (inversion) is performed [79, 80] to generate realizations in the input space $R^{N_C}$. This results in a nonlinear, implicit, and differentiable parameterization of the input random field in terms of a small number of independent random variables. Again, because the kernel formulation of the eigenvalue problem is solved in the feature space, an eigen decomposition of only a small kernel matrix of size $N_R \times N_R$ is required even though the dimension of the feature space could be extremely large. The approach is therefore essentially as efficient as the kernel formulation of the standard K-L expansion.

This chapter proceeds as follows. We first describe the basic K-L expansion and then formulate this expansion as a kernel eigenvalue problem. Next, kernel PCA procedures are applied to handle multi-point geostatistics, after which our approach for the pre-image problem is described. We then apply the kernel PCA approach for the solution of a history matching problem involving a channelized subsurface model. We note that many of the kernel PCA ideas and approaches described here are due to Scholkopf et al. [22] and Scholkopf and Smola [78] and have been described in detail previously within the machine learning literature. Although our presentation in some places closely follows these earlier expositions, we present the approaches in full detail in this chapter as these ideas appear to be new within the context of subsurface characterization and history matching. We note further that several of the algorithms applied in this work were modified from the Statistical Pattern Recognition Toolbox for Matlab [81].

## 6.1. The Karhunen-Loeve Expansion of Random Fields

The basic theory of the Karhunen-Loeve expansion has been described in Chapter 3 and will not be discussed here. This and the next section focus on the eigenvalue problem that has to be solved in order to perform the K-L expansion. Given a set of discrete centered (i.e., mean = 0) conditioned or unconditioned realizations of a

random field $\mathbf{y}_k$, $k = 1,..,N_R$, $\left(\mathbf{y}_k \in R^{N_C}\right)$, the covariance matrix can be calculated as [22]:

$$\mathbf{C} = \frac{1}{N_R}\sum_{j=1}^{N_R}\mathbf{y}_j\mathbf{y}_j^T \tag{6.1}$$

The number of realizations $N_R$ should be large enough to yield a converged covariance matrix **C**. This approach of calculating the covariance matrix numerically from a set of realizations of the random field as above, instead of using some analytical covariance model, is the most general approach for calculating the covariance. This is because for any arbitrary random field, an analytical covariance model may not exist, but the numerical approach can always be applied. For example, if a number of realizations are created using a multi-point geostatistical algorithm such as *snesim* [76], the covariance matrix can always be calculated numerically from those realizations; however, it is very likely that an analytical covariance model associated with those realizations may not exist.

The discrete K-L expansion that generates realizations with the above covariance **C** is given as [56]:

$$\mathbf{y} = \mathbf{E}\boldsymbol{\Lambda}^{1/2}\boldsymbol{\xi} \quad \equiv \quad \mathbf{y} = f\left(\boldsymbol{\xi}\right) \tag{6.2}$$

In the above, **E** is the matrix of eigenvectors of the covariance matrix **C**, $\boldsymbol{\Lambda}$ is a diagonal matrix of the eigenvalues of **C**, and $\boldsymbol{\xi}$ (vector) is a set of uncorrelated random variables, and these random variables are also independent if the random field is multi-Gaussian. In other words, if the elements of $\boldsymbol{\xi}$ are drawn from the standard Gaussian distribution, the $\mathbf{y}$ obtained using Equation (6.2) will be multi-Gaussian correlated with covariance **C**. Thus, the K-L expansion can be used for simulation of multi-Gaussian random fields. On the other hand, if the covariance matrix **C** is obtained from a set of realizations having higher order statistics, Equation (6.2) can still be used

to obtain realizations $\mathbf{y}$, but those realizations will only have the same covariance $\mathbf{C}$ as the original realizations used to calculate $\mathbf{C}$. The higher order moments in the original realizations will in general not be reproduced.

The K-L expansion is thus a parameterization of the form $\mathbf{y} = f(\xi)$, where the functional relationship is linear. Here, matrix $\mathbf{C}$ is of size $N_C \times N_C$. The maximum size of the matrices $\mathbf{E}$ and $\mathbf{\Lambda}$ is $N_C \times N_C$, and that of vector $\xi$ is $N_C \times 1$. Note that the word maximum is used because we may choose to retain only the largest $N_M$ of the total $N_C$ eigenvalues, in which case $\mathbf{E}$ is of size $N_C \times N_M$, $\mathbf{\Lambda}$ is of size $N_M \times N_M$, and $\xi$ is of size $N_M \times 1$. Further, as we will see later, the maximum number of non-zero eigenvalues is actually the minimum of $N_C$ and $N_R$, implying that $N_C$ non-zero eigenvalues do not exist if $N_R < N_C$. Thus, the random field $\mathbf{y}$ finally is parameterized in terms of $N_M$ independent random variables $\xi$.

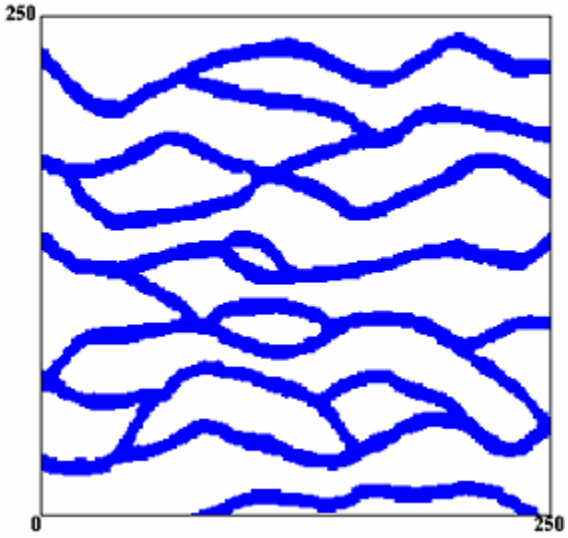


Figure 6-1 Channel training image used to create the original realizations [77]

Discarding the ($N_C$ - $N_M$) smallest eigenvalues implies that we are discarding the shortest correlation lengths. However, because the K-L expansion is an optimal expansion for multi-Gaussian random fields in a least-square sense [56], this implies that of all possible parameterizations of the multi-Gaussian random field with $N_M$

random variables, the K-L expansion minimizes the least-square approximation error. Even when the random field is not multi-Gaussian, the K-L expansion minimizes the least-square approximation error of representing the realizations used to create the covariance matrix **C**. This is a very favorable property of the K-L expansion for the applications discussed earlier, because a relatively small $N_M$ is essential for efficiency of these applications, though this still provides accurate approximations of multi-Gaussian random fields.

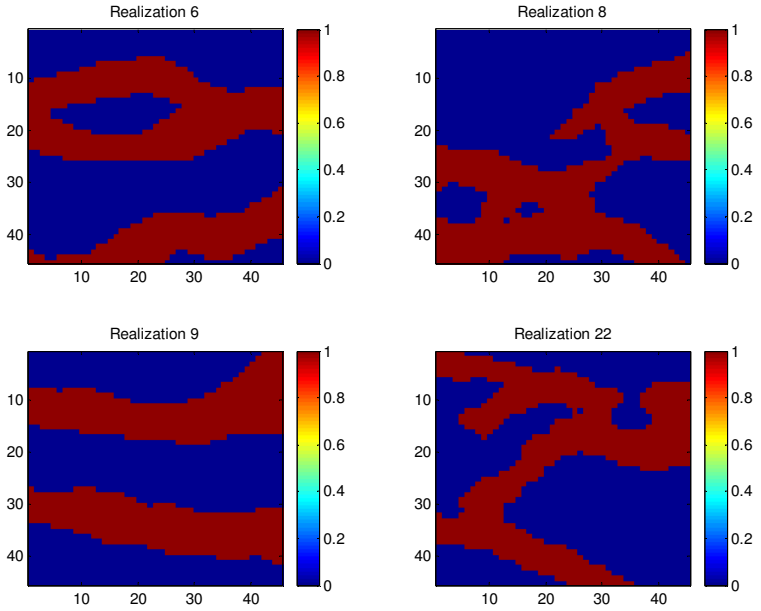


Figure 6-2 Some of the realizations created using *snesim*

In order to determine the K-L expansion, the following eigenvalue problem is solved [22, 78]:

$$\lambda \mathbf{v} = \mathbf{C}\mathbf{v} \tag{6.3}$$

Here, $\lambda$ are the eigenvalues of **C** and $\mathbf{v}$ are the eigenvectors of **C**. However, solving this problem directly with standard algorithms such as singular value decomposition (SVD) is a very expensive process of $O(N_C^3)$ complexity [82]. It is thus almost impossible to solve this problem for a large-scale simulation model with a few hundred thousand or more cells, and real simulation models are usually of this size. However, as introduced earlier, an alternative but exactly equivalent formulation of the

same problem, called the kernel eigenvalue problem, can be solved much more efficiently to determine the non-zero eigenvalues $\lambda$ and eigenvectors $\mathbf{v}$ of covariance matrix $\mathbf{C}$. The kernel eigenvalue formulation of the K-L expansion will be motivated with the following example.

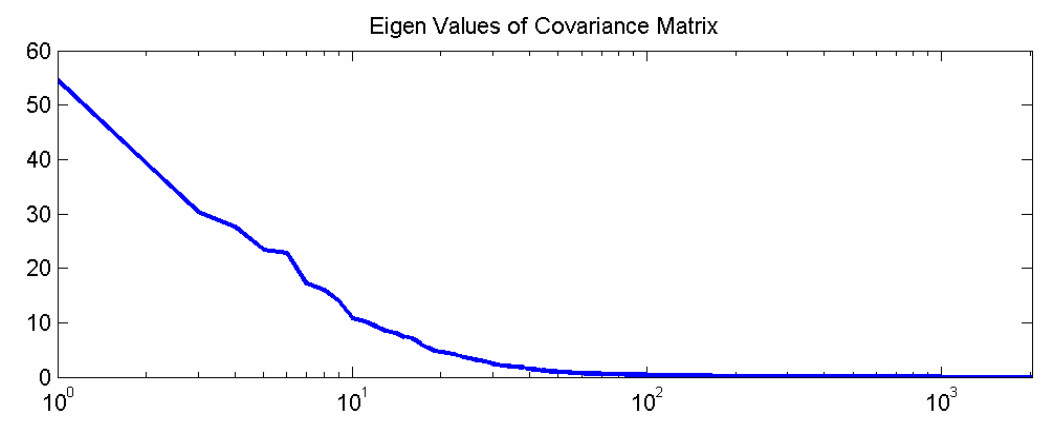


Figure 6-3 Eigenvalues (total = $N_C$ = 2025) of $\mathbf{C}$ arranged according to their magnitude

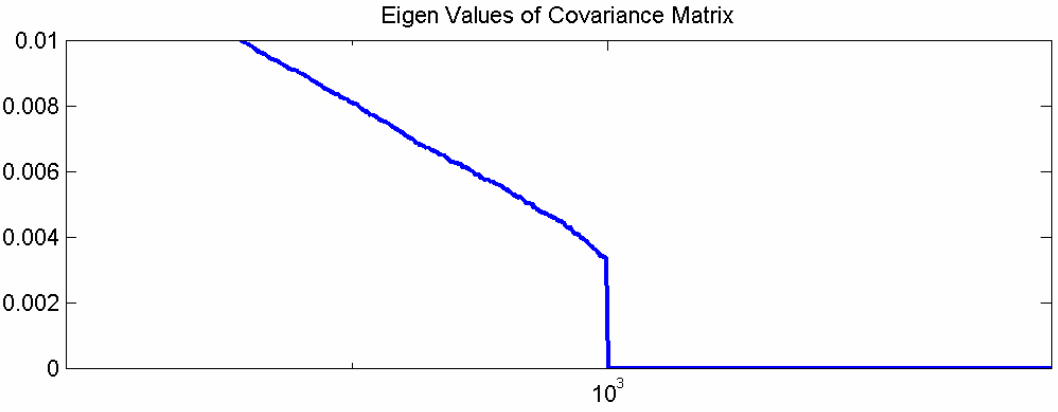


Figure 6-4 Neighborhood of the 1000$^{th}$ eigenvalue ($N_R$ = 1000)

Using the channel training image shown in Figure 6-1, 1000 ($N_R$ = 1000) 2D unconditional realizations of a channel permeability field, of dimension 45 × 45 ($N_C$ = 2025), are created using the *snesim* software [76]. Some of the realizations created are shown in Figure 6-2. These realizations are used to calculate the covariance matrix $\mathbf{C}$, and Figure 6-3 shows the eigenvalues of matrix $\mathbf{C}$ arranged according to their magnitude. Note that the total number of possible eigenvalues is equal to $N_C$ (2025). We observe that the eigenvalues reduce very rapidly and become close to zero at

around the 100th eigenvalue, but are not exactly zero. More importantly, if we focus on the neighborhood of the 1000th eigenvalue (Figure 6-4), we observe that the eigenvalues become exactly zero after the 1000th eigenvalue. This is precisely equal to the number of realizations $N_R$ used to create **C**. This occurrence is always the case (as proved in [22]), indicating that the maximum number of non-zero eigenvalues of **C** is equal to $N_R$ if $N_R < N_C$ and to $N_C$ if $N_C < N_R$. This key observation serves as the basis for the kernel eigenvalue problem.

### 6.2. The K-L Expansion as a Kernel Eigenvalue Problem

Using the definition of the covariance matrix **C** from Equation (6.1) and the eigenvalue problem given by Equation (6.3), the following equation is obtained [22, 78]:

$$\mathbf{C}\mathbf{v} = \frac{1}{N_R}\sum_{j=1}^{N_R}\left(\mathbf{y}_j.\mathbf{v}\right)\mathbf{y}_j \tag{6.4}$$

From elementary linear algebra, this implies that all solutions $\mathbf{v}$ with $\lambda \neq 0$ must lie in the span of the $N_R$ realizations $\mathbf{y}_1,....,\mathbf{y}_{N_R}$. Since eigenvectors $\mathbf{v}$ must lie in the span of $N_R$ realizations, and there cannot be more than $N_R$ orthogonal directions in the span of $N_R$ realizations, therefore, there can only be $N_R$ non-zero eigenvalues associated with these $N_R$ eigenvectors. Thus, as indicated above, **C** can have a maximum of only $N_R$ non-zero eigenvalues if $N_R < N_C$. This has two important consequences [22, 78]. First, Equation (6.3) can be written in the following equivalent manner:

$$\lambda\left(\mathbf{y}_k.\mathbf{v}\right) = \left(\mathbf{y}_k.\mathbf{C}\mathbf{v}\right) \quad \forall \quad k = 1,...,N_R \tag{6.5}$$

and second, there exist coefficients $\alpha_j$ such that:

$$\mathbf{v} = \sum_{j=1}^{N_R}\alpha_j\mathbf{y}_j \tag{6.6}$$

Combining Equations (6.5) and (6.6), we obtain [22, 78]:

$$\lambda \sum_{i=1}^{N_R} \alpha_i \left( \mathbf{y}_k . \mathbf{y}_i \right) = \frac{1}{N_R} \sum_{i=1}^{N_R} \alpha_i \left( \mathbf{y}_k . \sum_{j=1}^{N_R} \mathbf{y}_j \right) \left( \mathbf{y}_j . \mathbf{y}_i \right) \quad \forall \quad k = 1, ..., N_R \tag{6.7}$$

Now, defining an $N_R \times N_R$ matrix $\mathbf{K}$ where $K_{ij} = \left( \mathbf{y}_i . \mathbf{y}_j \right)$, that is, $K_{ij}$ is the dot product of realizations *i* and *j*, Equation (6.7) can be written as [22, 78]:

$$N_R \lambda \mathbf{K} \boldsymbol{\alpha} = \mathbf{K}^2 \boldsymbol{\alpha} \tag{6.8}$$

$\mathbf{K}$ is called the kernel matrix, and it is of size $N_R \times N_R$, whereas, the covariance matrix $\mathbf{C}$ is of size $N_C \times N_C$. Here $K_{ij} = \left( \mathbf{y}_i . \mathbf{y}_j \right)$ is called the polynomial kernel of order 1 [22, 78]. In the next section, this will be generalized with higher order polynomial kernels that will allow us to preserve higher order moments (multi-point statistics). It can be shown that the eigenvalues and eigenvectors of the eigenvalue problem of Equation (6.8) are equivalent to those of the following eigenvalue problem [22, 78]:

$$N_R \lambda \boldsymbol{\alpha} = \mathbf{K} \boldsymbol{\alpha} \tag{6.9}$$

Equation (6.9) is known as the kernel eigenvalue problem. The eigenvalues of this are given by $N_R \lambda$ and the eigenvectors are given by $\boldsymbol{\alpha}$. Solving this problem is exactly equivalent to solving Equation (6.3), because the non-zero eigenvalues of Equation (6.3) are just that of Equation (6.9) scaled by $N_R$, and the eigenvectors $\mathbf{v}$ associated with the non-zero eigenvalues $\lambda$ of Equation (6.3) can be obtained from $\boldsymbol{\alpha}$ by using Equation (6.6). The rest of the $N_C - N_R$ eigenvalues of $\mathbf{C}$ are equal to zero, as was also seen in the earlier example. The attractiveness of solving the kernel eigenvalue problem of Equation (6.9), instead of the original problem of Equation (6.3), is that for practical problems $N_R \ll N_C$, and therefore the eigen decomposition of the kernel matrix $\mathbf{K}$ can be done extremely efficiently compared to that of the covariance matrix $\mathbf{C}$, which may not even be possible with current computing technology.

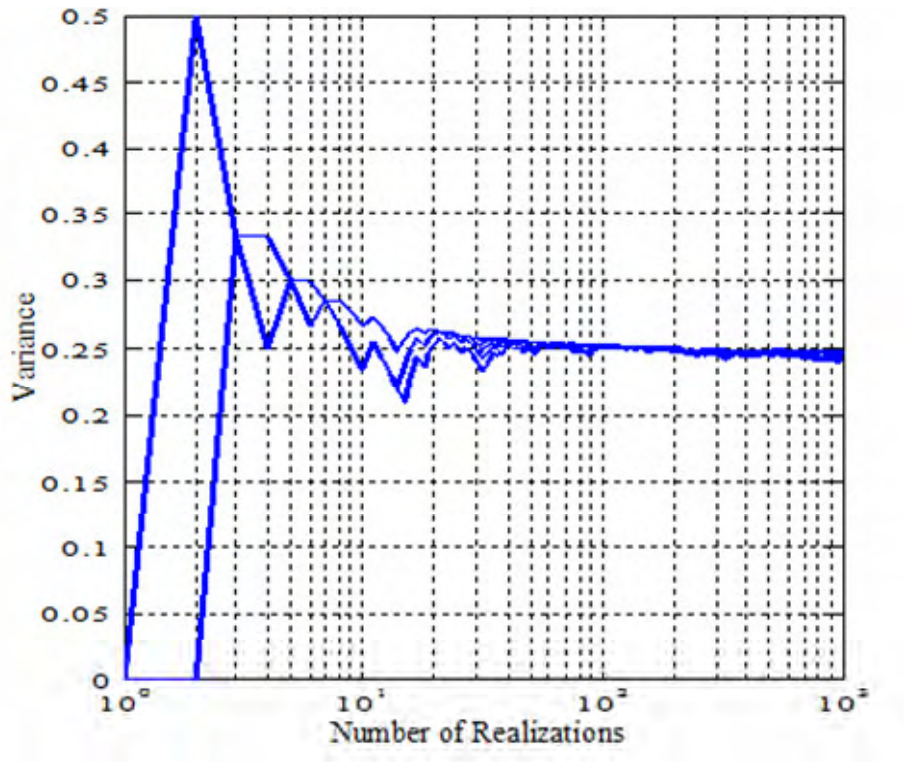


Figure 6-5 Convergence of variance of permeability of a few cells

Consider the example discussed in the previous section. We can calculate the number of realizations that would be required in this problem to obtain a converged covariance matrix (of permeability). In order to determine the convergence of the covariance matrix and therefore the appropriate $N_R$, the convergence of the variance of permeability of a few cells of the permeability field (chosen randomly) are calculated using an increasing number of realizations. Figure 6-5 shows the variance of these cells calculated using increasing numbers of realizations (10 cells were chosen, but most of the curves overlap, thus we see only 2 curves). We see that about 100 realizations are required for the variance of the cell permeabilities to converge. This implies that $N_R = 100$ is sufficient for this problem, whereas $N_C = 2025$, corroborating the fact that $N_R \ll N_C$. Thus the kernel matrix $\mathbf{K}$ is of size $100 \times 100$ in this problem and is much smaller than the covariance matrix $\mathbf{C}$ of size $2025 \times 2025$. In general, an $N_R$ of the order $10^{2\text{-}3}$ should be sufficient even for problems with $N_C$ of the order $10^{5\text{-}6}$, which is the usual size of practical simulation models.

## 6.3. Preserving Multi-point Statistics using Kernel PCA

In the previous section, the eigenvalue problem associated with the Karhunen-Loeve expansion (linear PCA) was expressed in terms of the kernel eigenvalue problem with a polynomial kernel of order one, resulting in significant improvement in the efficiency of the solution of the eigenvalue problem. However, kernel methods have a much wider range of applicability than this. In this section, polynomial kernels of higher orders are applied to perform the Karhunen-Loeve expansion not in the original space of the realizations $R^{N_C}$, but in a high order space called feature space $F$, thereby preserving higher order moments (multi-point statistics) rather than just the second moment (two-point statistics). Recall that the standard K-L expansion only preserves the two-point statistics of the random field, however, multi-point statistics have to be preserved in order to accurately model complex geological structures like channels. Since the feature space $F$ is nonlinearly related to the original space $R^{N_C}$, this is essentially a form of nonlinear PCA. The efficiency of performing nonlinear PCA with kernels is similar to that of linear PCA with kernels, as will be demonstrated later. Note that other types of kernels also exist, such as radial kernels and exponential kernels, which are used in other applications (e.g., support vector machines [22, 78]).

In order to motivate the necessity of creating a parameterization capable of preserving multi-point statistics, consider a permeability field represented by the training image of Figure 6-1. As explained before, 1000 realizations are created using the *snesim* software [76], some of which are shown in Figure 6-2. These 1000 realizations are used to create the standard K-L expansion using the kernel formulation with a polynomial kernel of order one. Although there are 1000 non-zero eigenvalues ($N_R$ = 1000), only the largest 30 are retained for the K-L expansion ($N_M$ = 30), which corresponds to about 75% of the energy of the random field as seen in Figure 6-6.

This K-L expansion can now be used for geostatistical simulation by drawing 30 standard normal random variables and applying the K-L expansion to generate a realization $\mathbf{y}$ having the same covariance as the random field depicted by the training

image of Figure 6-1. Some of these realizations are shown in Figure 6-7. It is clear that although the realizations show a longer correlation length in the horizontal direction (direction of the channels), they do not reproduce channelized models. Further, the marginal distribution is not reproduced, which is clearly a binary bimodal distribution for the original realizations. This is because only the covariance (two-point statistics) of the original realizations is preserved by the K-L expansion. However, the original realizations are clearly non-Gaussian with higher order statistics, which should be preserved by the parameterization if the original channel structure is to be reproduced to a reasonable degree of accuracy.

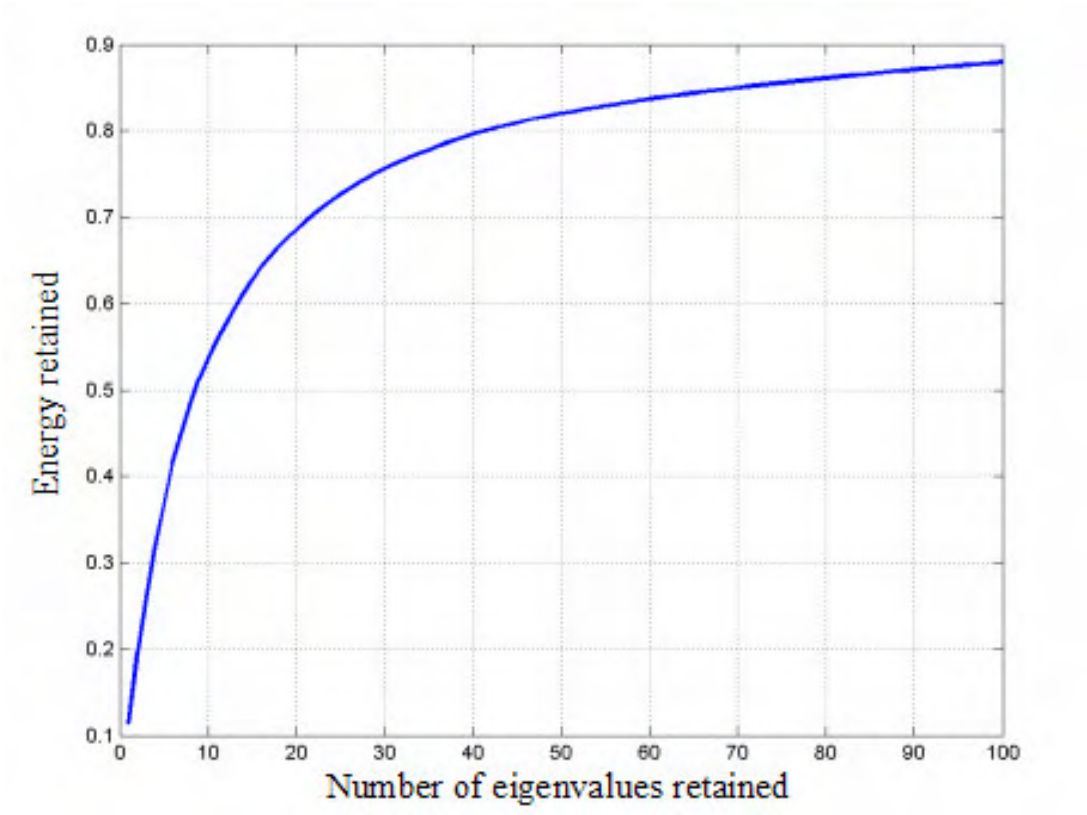


Figure 6-6 Energy retained in the first 100 eigenpairs

Our solution to the above problem, as mentioned earlier, is to apply nonlinear PCA with high order polynomial kernels. Figure 6-8 demonstrates the basic idea behind nonlinear kernel PCA [22, 78]. Consider an arbitrary random field in $R^2$ ($N_C = 2$), that is, each realization is a vector of dimension two, $\mathbf{y} = (y_1, y_2)^T$. Each realization can

thus be easily plotted as a point on a 2D graph, as shown in the left graph of Figure 6-8. Note that the realizations are nonlinearly related to each other. However, if linear PCA or the standard Karhunen-Loeve expansion were used, the major principal component obtained would be in the direction drawn as an arrow in the graph, and it is clearly not able to capture the nonlinear relationship among the realizations. Thus, the Karhunen-Loeve expansion is only able to capture linear relationships among the realizations in the space where it is performed, in this case $R^2$.

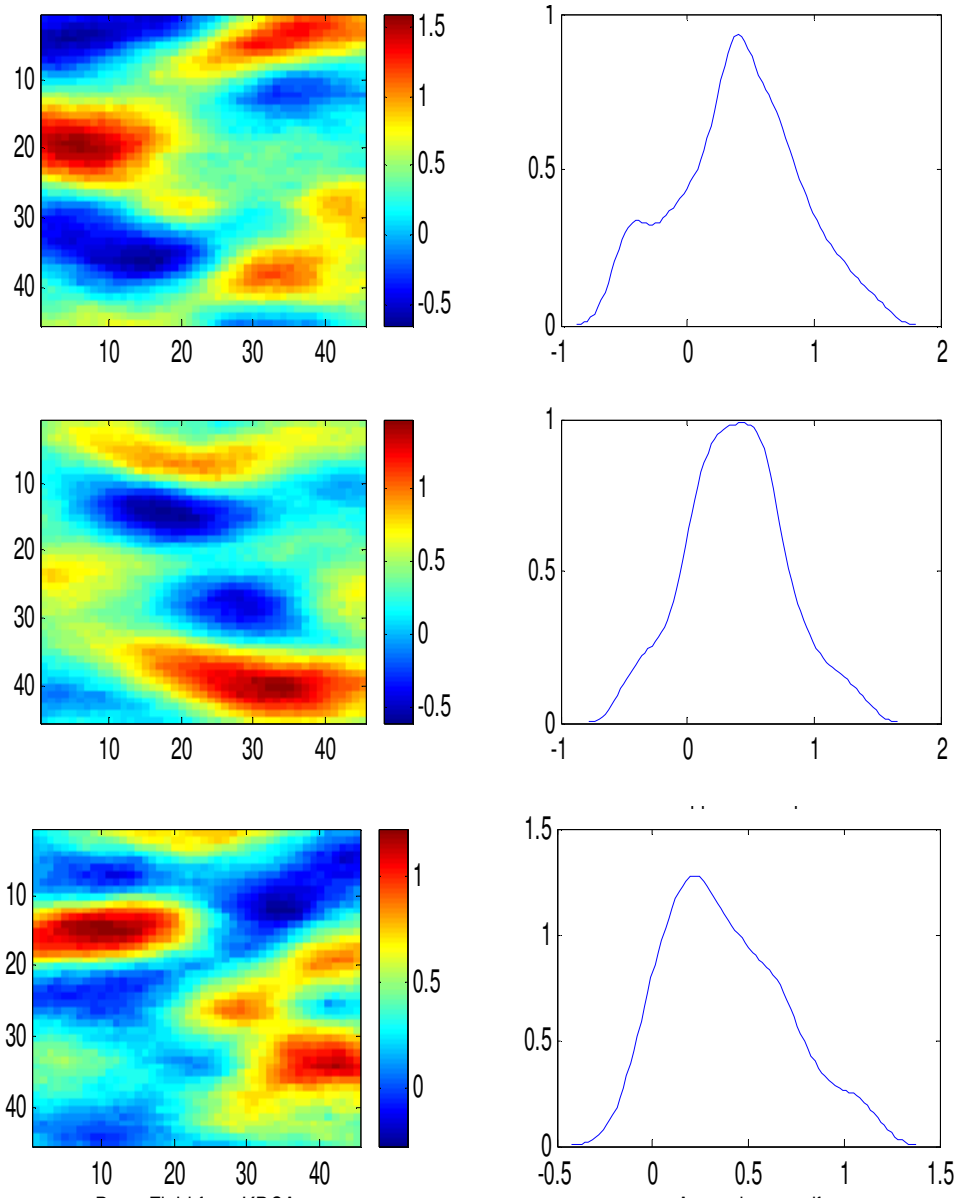


Figure 6-7 Some realizations and their marginal distributions with the standard K-L expansion

Now, consider a nonlinear mapping $\Phi$ that relates the input space $R^{N_C}$ to another space $F$. That is:

$$\Phi: \quad R^{N_C} \to F; \quad \mathbf{Y} = \Phi(\mathbf{y}); \quad \mathbf{y} \in R^{N_C}, \mathbf{Y} \in F \tag{6.10}$$

$F$ is called the feature space, and it could have an arbitrarily large dimensionality. The definition will become clearer when space $F$ is associated with a kernel function below. Now, as seen in the right graph of Figure 6-8, after this $\Phi$ transform, the realizations that were nonlinearly related in $R^2$ become linearly related in the feature space $F$. Thus, standard linear PCA or the Karhunen-Loeve expansion can now be performed in $F$ in order to determine the principal eigenvectors in this space.

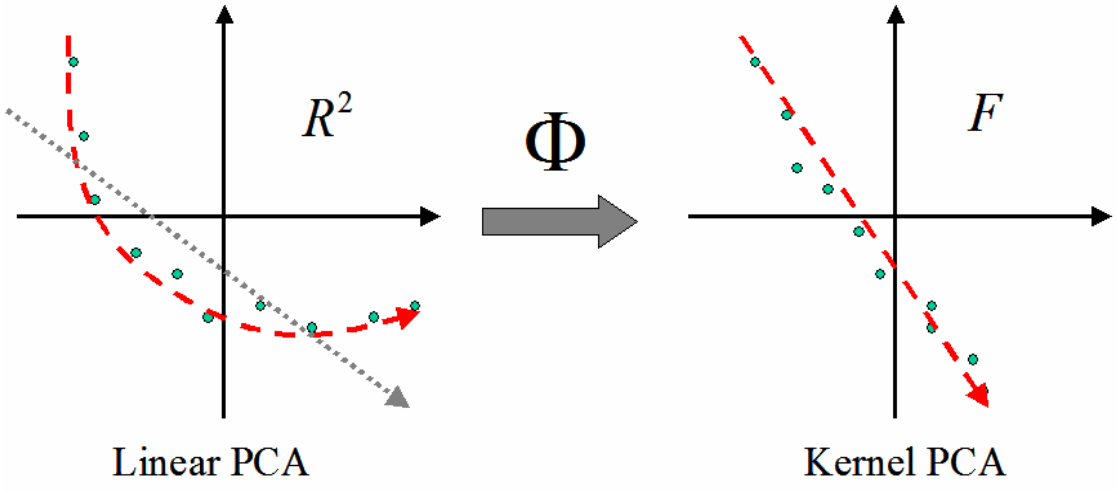


Figure 6-8 Basic idea behind kernel PCA (modified from [22])

To clarify further, in the context of geostatistical simulation, consider that a large number of realizations (each of length $N_C$) of a channelized permeability field have been obtained using some geostatistical software. Thus, the input space is $R^{N_C}$, and the realizations can be thought of as points in this space. These points (realizations) are nonlinearly related in $R^{N_C}$, however, because if they were linearly related, standard Karhunen-Loeve expansion (linear PCA) could be performed in $R^{N_C}$ to obtain channelized realizations (which does not occur as seen in the pervious example). Kernel PCA captures the nonlinear relationship between these realizations, thereby

allowing us to form new (channelized) realizations that satisfy this nonlinear relationship. Thus we create images that "look like" those generated using existing multi-point geostatistical software. The advantage, however, of the kernel PCA representation is that it provides a parameterized (differentiable) mathematical model that can be used with gradient-based methods.

Realizing that nonlinear kernel PCA is essentially linear PCA in a high dimensional feature space $F$ (as opposed to the input space $R^{N_C}$), all results discussed in the first section on linear PCA can be readily generalized for general kernel PCA. That is, the maps of the realizations $\mathbf{y}_k$, $k = 1,..,N_R$ in the feature space $F$ are $\Phi(\mathbf{y}_k)$, $k = 1,..,N_R$, and assuming $\Phi(\mathbf{y}_k)$ are centered (if not, they can be centered as in [79, 80]), the covariance matrix in the feature space $F$ is given as [22, 78]:

$$\bar{\mathbf{C}} = \frac{1}{N_R}\sum_{j=1}^{N_R}\Phi(\mathbf{y}_j)\Phi(\mathbf{y}_j)^T \tag{6.11}$$

Note that the dimension of this covariance matrix is not $N_C \times N_C$, but $N_F \times N_F$, where $N_F$ is the length of $\Phi(\mathbf{y})$, which could be extremely large. Similar to linear PCA, an eigenvalue problem again must be solved, but using $\bar{\mathbf{C}}$ instead of $\mathbf{C}$:

$$\lambda\mathbf{V} = \bar{\mathbf{C}}\mathbf{V} \tag{6.12}$$

Here, $\lambda$ are the eigenvalues of $\bar{\mathbf{C}}$ and $\mathbf{V}$ are the eigenvectors of $\bar{\mathbf{C}}$. Again, instead of solving this problem directly (which is impossible with current computing technology due to extremely large $N_F$), a kernel eigenvalue problem associated with Equation (6.12) is formulated [22, 78]:

$$N_R\lambda\boldsymbol{\alpha} = \mathbf{K}\boldsymbol{\alpha} \tag{6.13}$$

Here, the kernel matrix $\mathbf{K}$ is different from the kernel matrix applied for linear PCA, and is defined as [22, 78]:

$$\mathbf{K}: \quad K_{ij} = \left(\Phi(\mathbf{y}_i).\Phi(\mathbf{y}_j)\right) \tag{6.14}$$

Note that now each element of the kernel matrix is a dot product of vectors in the feature space *F*, and not the dot product of vectors of the input space of the realizations, $R^{N_C}$. However, the dimension of the kernel matrix is still $N_R \times N_R$, just as for linear PCA. The kernel formulation of Equation (6.13) is again exactly equivalent to the eigenvalue problem of Equation (6.12) without any assumptions. That is, as in linear PCA, we can obtain all the non-zero eigenvalues $\lambda$ and eigenvectors $\mathbf{V}$ of $\bar{\mathbf{C}}$ from the eigenvalues $N_R\lambda$ and eigenvectors $\boldsymbol{\alpha}$ of $\mathbf{K}$. There are only $N_R$ non-zero eigenvalues of $\bar{\mathbf{C}}$, if $N_R < N_F$, and in general $N_R \square N_F$.

Observe that in order to calculate the kernel matrix $\mathbf{K}$, only the dot product of vectors in the feature space *F* are required; the explicit calculation of the map $\Phi(\mathbf{y})$ is not required. This is extremely important, because it may not even be possible to calculate and store $\Phi(\mathbf{y})$ in the memory of current computers due to its very large dimension (as will be illustrated below). Since only the dot products in the space *F* are required for application of kernel PCA, and not $\Phi(\mathbf{y})$ itself, this can be calculated very efficiently with what is known as a kernel function [22, 78]:

$$\left(\Phi(\mathbf{x}).\Phi(\mathbf{y})\right) = k(\mathbf{x},\mathbf{y}) \tag{6.15}$$

The kernel function $k(\mathbf{x},\mathbf{y})$ calculates the dot product in space *F* directly from the elements of the input space $R^{N_C}$. That is, the right hand side of Equation (6.15) does not directly involve the mapping $\Phi(\mathbf{y})$. As mentioned before, there are various kinds of kernel functions available, but the kernel function of interest in this application is the polynomial kernel defined as [22, 78]:

$$\left(\Phi(\mathbf{x}).\Phi(\mathbf{y})\right) = k(\mathbf{x},\mathbf{y}) = (\mathbf{x}.\mathbf{y})^d \tag{6.16}$$

Here, $d$ is the order of the polynomial kernel. Every kernel function (satisfying Mercer's theorem) is uniquely associated to a mapping $\Phi$ [22, 78]. The polynomial kernel of order $d$, $(\mathbf{x}.\mathbf{y})^d$, corresponds to a feature space $F$ of $d^{th}$ order monomials of $R^{N_C}$. For example, if we take $d = 2$ and the input space is $R^3$ [22, 78]:

$$\begin{aligned} &k(\mathbf{x},\mathbf{y}) = (\mathbf{x}.\mathbf{y})^2; \quad \mathbf{x} = (x_1, x_2, x_3)^T; \mathbf{y} = (y_1, y_2, y_3)^T \\ &k(\mathbf{x},\mathbf{y}) = x_1^2 y_1^2 + x_2^2 y_2^2 + x_3^2 y_3^2 + 2x_1 x_2 y_1 y_2 + 2x_1 x_3 y_1 y_3 + 2x_2 x_3 y_2 y_3 \\ &\Phi(\mathbf{x}) = \left(x_1^2, x_2^2, x_3^2, \sqrt{2}x_1 x_2, \sqrt{2}x_1 x_3, \sqrt{2}x_2 x_3\right)^T \end{aligned} \tag{6.17}$$

Notice that $\Phi(\mathbf{x})$ contains the product of 2 elements of $\mathbf{x}$ at a time (for $d = 3$, we would take the product of 3 elements of $\mathbf{x}$ at a time, etc.). Thus, for $d = 2$, the covariance matrix in $F$, given by $\bar{\mathbf{C}}$, corresponds to fourth order moments or four-point statistics of the input space $R^{N_C}$. In general, for the polynomial kernel $(\mathbf{x}.\mathbf{y})^d$ of the order $d$, $\bar{\mathbf{C}}$ corresponds to the $2d^{th}$ order moment of $R^{N_C}$. Therefore, performing linear PCA or the Karhunen-Loeve expansion in the feature space $F$ corresponding to the polynomial kernel $(\mathbf{x}.\mathbf{y})^d$ corresponds to preserving the $2d^{th}$ order moment or $2d$-point statistics of $R^{N_C}$. However, since we are interested not only in preserving the $2d^{th}$ order moment but all the moments up to the $2d^{th}$ order moment, the following kernel is used:

$$(\Phi(\mathbf{x}).\Phi(\mathbf{y})) = k(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{d} (\mathbf{x}.\mathbf{y})^i \tag{6.18}$$

A similar result can be obtained using an inhomogeneous polynomial kernel as described in [78]. Note that the first moment or mean is easily preserved by directly adding it to the parameterization as in the standard K-L expansion.

It was mentioned earlier that the dimensionality of the feature space $F$ could be very large. For the polynomial kernel $(\mathbf{x}.\mathbf{y})^d$ and an input space of realizations $R^{N_C}$, the dimension of $F$ is given as [22, 78]:

$$N_F = \frac{(N_C + d - 1)!}{d!(N_C - 1)!} \tag{6.19}$$

which can be readily seen to scale as $(N_C)^d$. Thus if $N_C$ (length of realization) is $10^5$ (a relatively small value for geological models), and $d$ is moderately small, say 3 (corresponding to preserving the 6th order moments of the realizations in $R^{N_C}$), $N_F \approx 10^{15}$, which is extremely large.

The physical argument why we are able to perform the Karhunen-Loeve expansion in such a large dimensional space is because we are not working in the full feature space $F$, but in a comparably small linear subspace of it, whose dimension equals at most the number of realizations $N_R$. The method automatically chooses this subspace and provides a means of taking advantage of the lower dimensionality. Kernel PCA does not explicitly compute all dimensions of $F$ (all possible features), but works only in a relevant subspace of $F$, which is spanned by $\Phi(\mathbf{y}_k)$, $k$ = 1,..,$N_R$, and therefore only takes the features present in the $\Phi(\mathbf{y}_k)$ realizations into account. The numerical argument is that we are calculating the dot products using the kernel function directly on elements of $R^{N_C}$, which can be done very efficiently, and explicit calculation of the mapping $\Phi(\mathbf{x})$ is not required.

### 6.4. The Pre-image Problem for Parameterizing Geology

It was mentioned in the last section that because linear K-L expansion is performed in the high order feature space $F$, the results of the K-L expansion (realizations) thus lie in the feature space, that is, a simulated realization $\mathbf{Y} \in F$. We are, however, interested in obtaining realizations in the original space of the input random field $R^{N_C}$,

because the goal is to obtain a parameterization of this input random field. In order to obtain a realization $\mathbf{y}$ in the original space of the realizations $R^{N_C}$ that corresponds to this simulated realization $\mathbf{Y} \in F$, an inverse $\Phi$ map of $\mathbf{Y}$ is required, that is, $\mathbf{y} = \Phi^{-1}(\mathbf{Y})$. This is known as the pre-image problem [79, 80]. However, due to the very large dimensionality of the feature space *F*, it may not be possible to calculate this pre-image, and further, such a pre-image may not even exist, or, if it exists, it may be non-unique [79, 80]. All these issues can be resolved by solving a minimization problem, in which a vector $\mathbf{y}$ is sought such that the least-square error between $\Phi(\mathbf{y})$ and $\mathbf{Y}$ is minimized [79, 80]:

$$\min_{\mathbf{y}} \ \rho(\mathbf{y}) = \left\| \Phi(\mathbf{y}) - \mathbf{Y} \right\|^2 = \Phi(\mathbf{y}).\Phi(\mathbf{y}) - 2\mathbf{Y}.\Phi(\mathbf{y}) + \mathbf{Y}.\mathbf{Y} \tag{6.20}$$

Now, using the counterparts of Equations (6.2) and (6.6) in the feature space *F*, it can be shown that any realization $\mathbf{Y}$ is a linear combination of the $\Phi$ maps $\Phi(\mathbf{y}_k)$ of the input realizations $\mathbf{y}_k$, $k = 1,..,N_R$. That is:

$$\mathbf{Y} = \sum_{i=1}^{N_R} \beta_i \Phi(\mathbf{y}_i) \tag{6.21}$$

Note that the coefficients $\beta_i$ are functions of the independent random variables $\xi$ and are given as:

$$\beta_i = \frac{1}{\sqrt{N_R}} \sum_{j=1}^{N_R} \alpha_{ij} \delta_j^{1/2} \xi_j \tag{6.22}$$

Here, $\boldsymbol{\alpha}_i = \left[ \alpha_{1i}, \alpha_{2i}, ..., \alpha_{N_R i} \right]^T$ and $\delta_i$ are the $i^{th}$ eigenvector and eigenvalue of the kernel matrix $\mathbf{K}$ respectively. Using Equation (6.21) and replacing the dot products in Equation (6.20) with the kernel function, the objective function of Equation (6.20) can be written as:

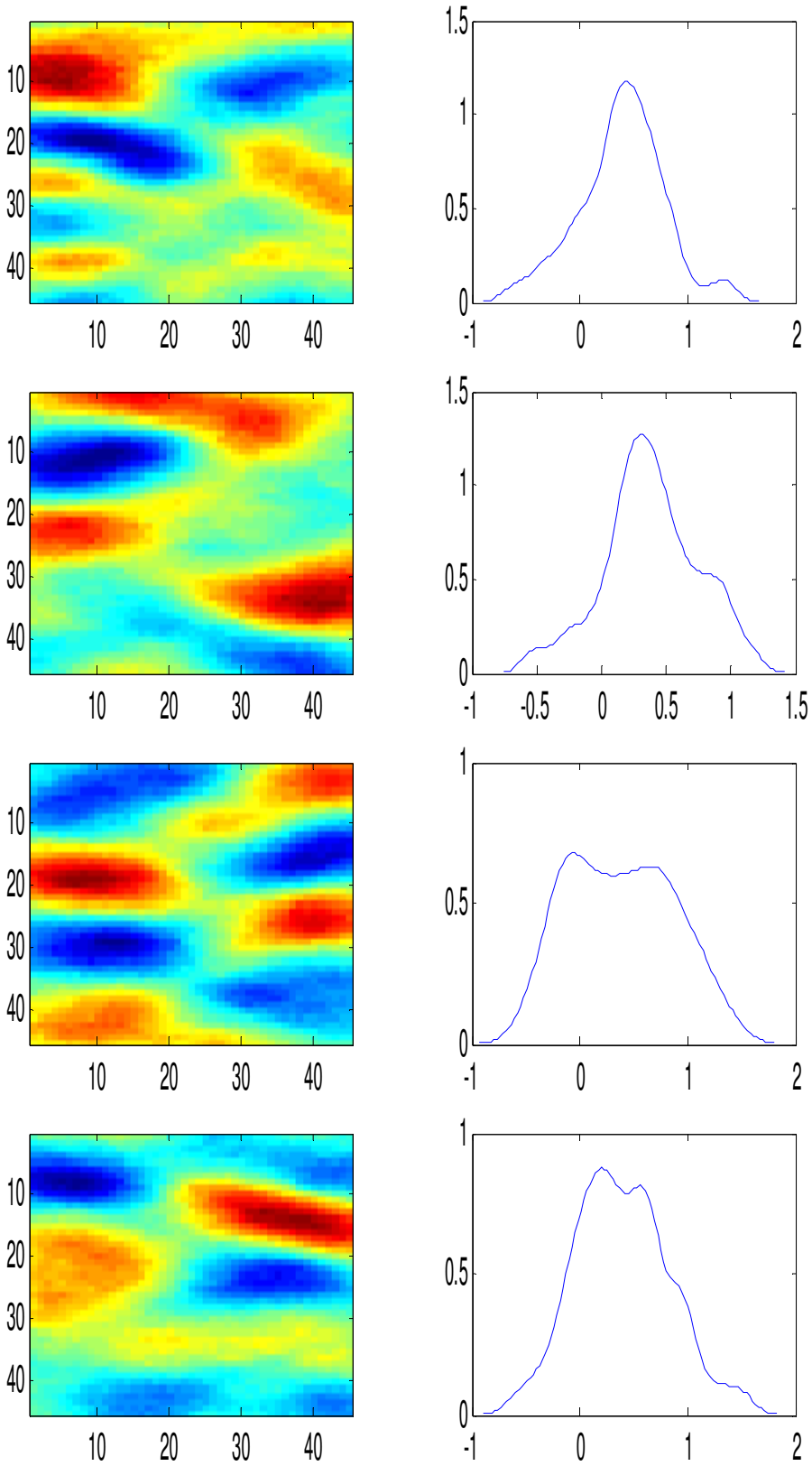


Figure 6-9 Typical realizations obtained with linear PCA and their marginal *pdfs*

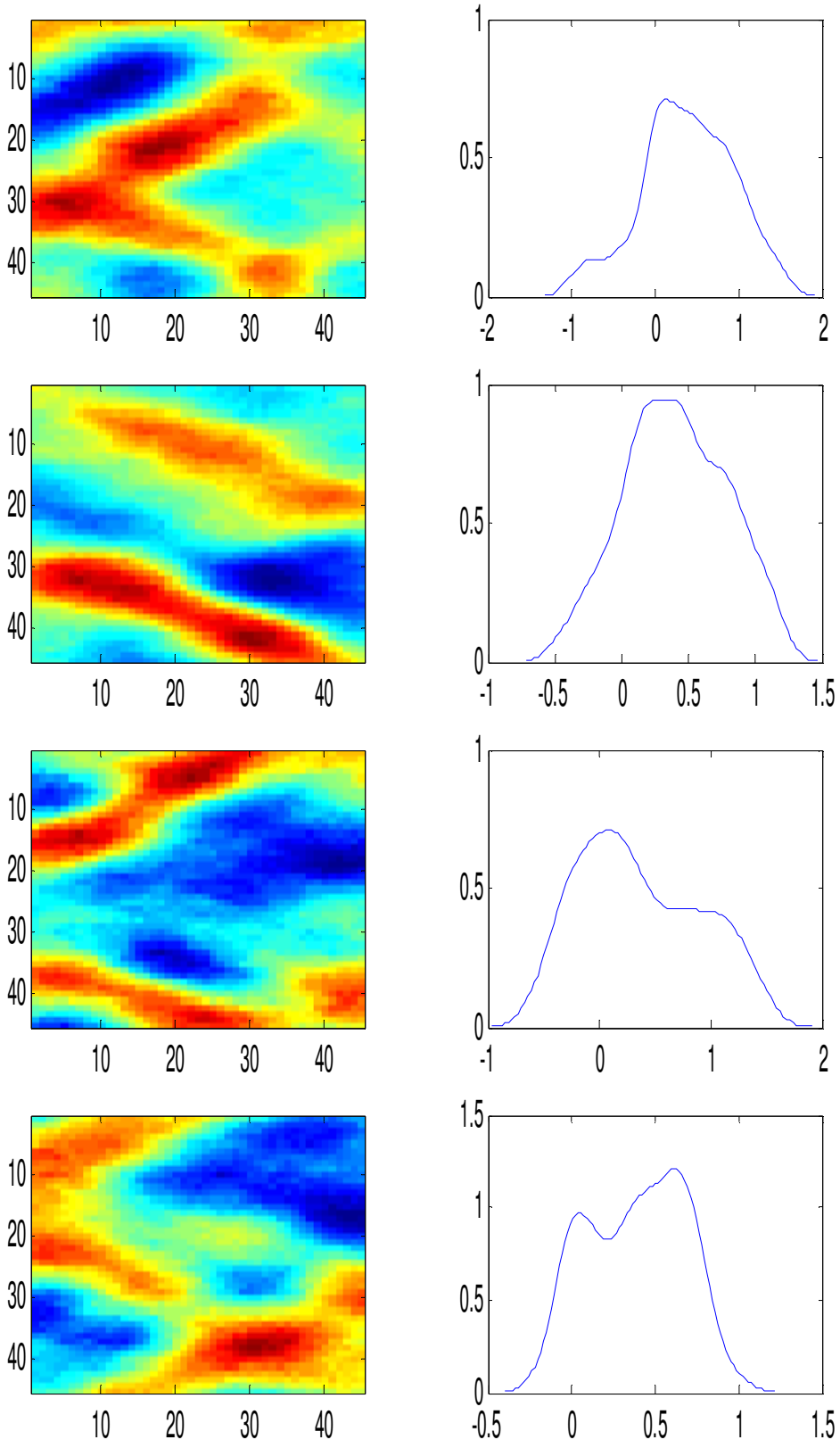


Figure 6-10 Typical realizations obtained with kernel PCA of order 2 and their marginal *pdfs*

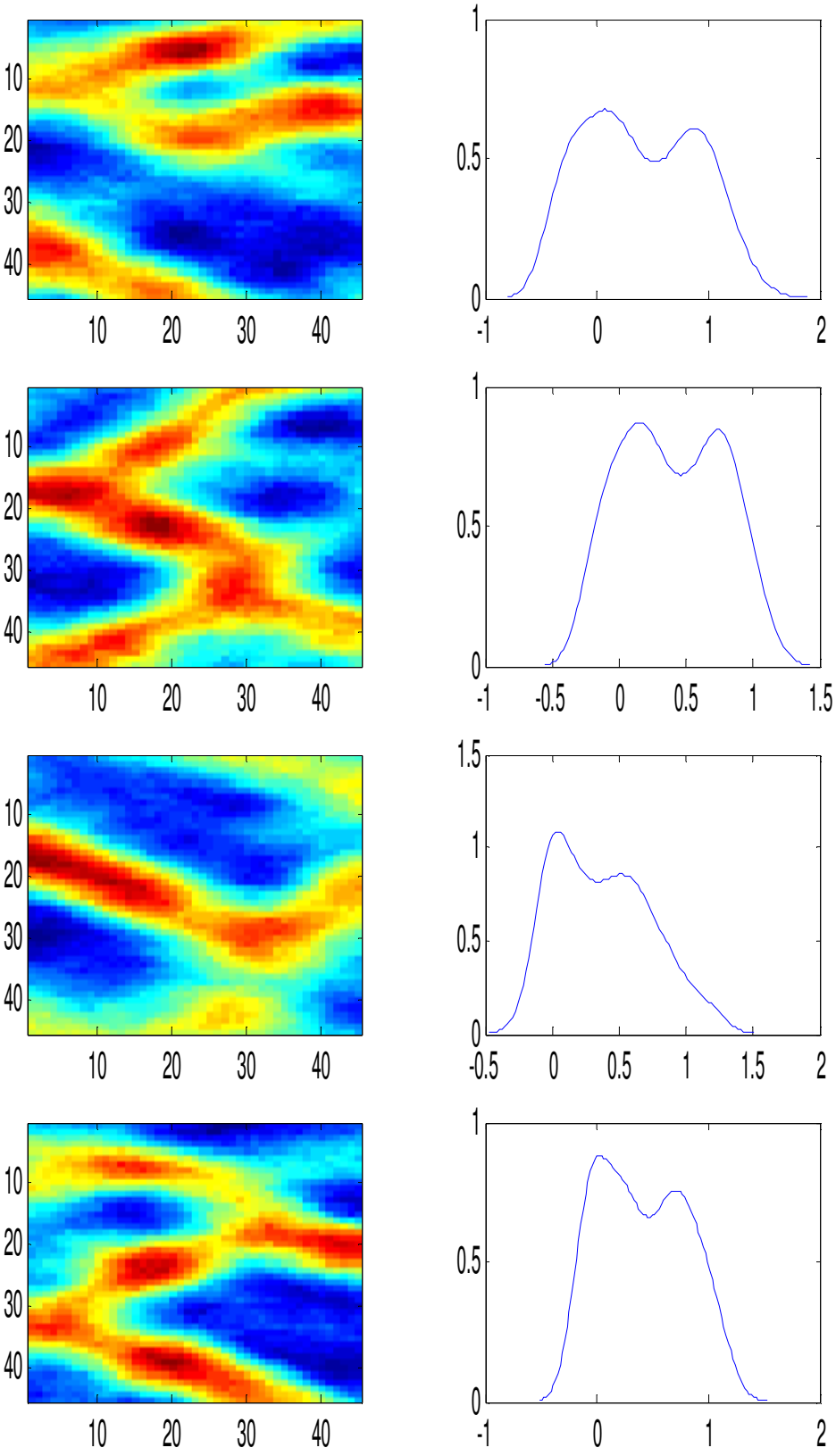


Figure 6-11 Typical realizations obtained with kernel PCA of order 3 and their marginal *pdfs*

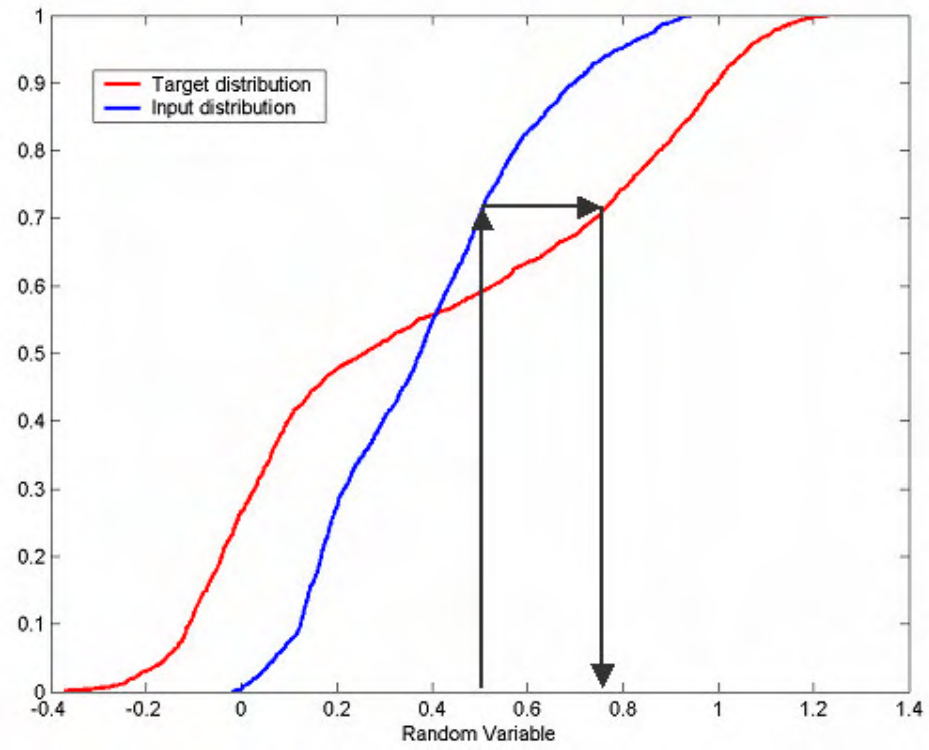


Figure 6-12 Pictorial representation of cdf transform

$$\rho(\mathbf{y}) = k(\mathbf{y},\mathbf{y}) - 2\sum_{i=1}^{N_R}\beta_i k(\mathbf{y}_i,\mathbf{y}) + \sum_{i=1}^{N_R}\sum_{j=1}^{N_R}\beta_i\beta_j k(\mathbf{y}_i,\mathbf{y}_j) \tag{6.23}$$

The minimum of the objective function $\rho(\mathbf{y})$ can be obtained by setting its gradient to zero, resulting in the following equation:

$$\frac{dk(\mathbf{y},\mathbf{y})}{d\mathbf{y}} - 2\sum_{i=1}^{N_R}\beta_i\frac{dk(\mathbf{y}_i,\mathbf{y})}{d\mathbf{y}} = 0 \tag{6.24}$$

This equation can be further reduced to the following specific equation for the kernel given by Equation (6.18):

$$\sum_{j=1}^{d} j(\mathbf{y}.\mathbf{y})^{j-1}\mathbf{y} - \sum_{i=1}^{N_R}\beta_i(\xi)\sum_{j=1}^{d} j(\mathbf{y}_i.\mathbf{y})^{j-1}\mathbf{y}_i = 0 \tag{6.25}$$

This is a differentiable parameterization relating a realization $\mathbf{y}$ in the input space $R^{N_C}$ to a set of independent random variables $\xi$, because $\beta_i$ are functions of $\xi$. Note that this is a nonlinear implicit parameterization of the form $f(\mathbf{y},\xi)=0$. An efficient method to solve for $\mathbf{y}$ for a given $\xi$ using Equation (6.25) is to apply a fixed-point iteration method [79], wherein the iteration scheme is given as:

$$\mathbf{y}^{k+1}=\frac{\sum_{i=1}^{N_R}\beta_i\sum_{j=1}^{d}j\left(\mathbf{y}_i.\mathbf{y}^k\right)^{j-1}\mathbf{y}_i}{\sum_{j=1}^{d}j\left(\mathbf{y}^k.\mathbf{y}^k\right)^{j-1}} \tag{6.26}$$

Unfortunately, this iteration scheme is not very stable for the above kernel, although the method is quite stable for other kernels such as the Gaussian kernel [79]. However, by noticing that the denominator of the above equation is a function of the magnitude of $\mathbf{y}$ and therefore essentially acts as a normalization term, and also by comparison to the iteration scheme using the Gaussian kernel [79], the denominator is modified to obtain the final iteration scheme:

$$\mathbf{y}^{k+1}=\frac{\sum_{i=1}^{N_R}\beta_i\sum_{j=1}^{d}j\left(\mathbf{y}_i.\mathbf{y}^k\right)^{j-1}\mathbf{y}_i}{\sum_{i=1}^{N_R}\beta_i\sum_{j=1}^{d}j\left(\mathbf{y}_i.\mathbf{y}^k\right)^{j-1}} \tag{6.27}$$

Not only is the scheme quite stable, but even more importantly, it results in a nonlinear combination of the original realizations with the weights summing up to one. This implies that if the original realizations all honor certain hard data, any realization obtained with Equation (6.27) will also honor the same hard data. Thus, any hard data conditioning incorporated into the original realizations is maintained with the above scheme. The final implicit parameterization consistent with Equation (6.27) is thus given as:

$$f(\mathbf{y},\boldsymbol{\xi}) = \sum_{i=1}^{N_R} \beta_i \sum_{j=1}^{d} j\left(\mathbf{y}_i.\mathbf{y}\right)^{j-1} \mathbf{y} - \sum_{i=1}^{N_R} \beta_i \sum_{j=1}^{d} j\left(\mathbf{y}_i.\mathbf{y}\right)^{j-1} \mathbf{y}_i = 0 \tag{6.28}$$

Application of this parameterization in conjunction with other gradient-based algorithms would usually require the gradient of $\mathbf{y}$ with respect to $\boldsymbol{\xi}$. This is obtained by setting $df = 0$.

$$\frac{d\mathbf{y}}{d\boldsymbol{\xi}} = -\left(\frac{\partial f}{\partial \mathbf{y}}\right)^{-1} \frac{\partial f}{\partial \boldsymbol{\xi}} \tag{6.29}$$

Note that the above gradient is obtained analytically using Equation (6.28) and no optimization or any other numerical problem has to be solved in order to do so.

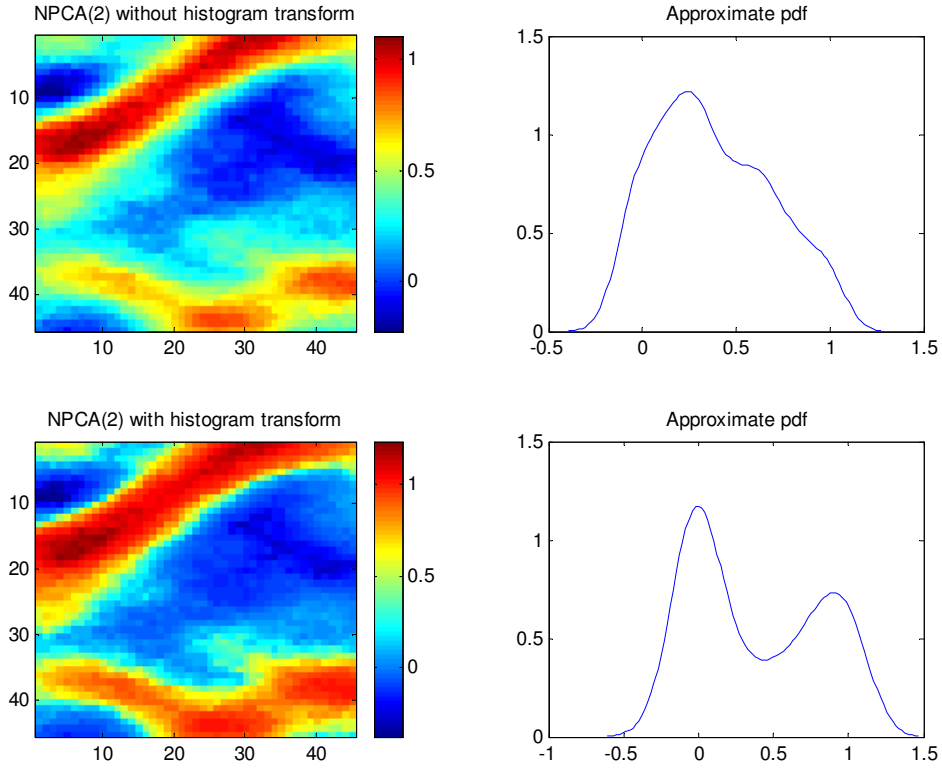


Figure 6-13 Realizations before and after histogram transform

In order to demonstrate the validity of kernel PCA for parameterizing multi-point geostatistics, Equation (6.28) is applied for geostatistical simulation of the channel sand example discussed earlier. Using the channel training image shown in Figure 6-1,

1000 ($N_R$ = 1000) 2D unconditional realizations of a channelized permeability field of dimension 45 × 45 ($N_C$ = 2025) are created using the *snesim* software [76]. Some of the realizations are shown in Figure 6-2. These realizations are used to create kernel matrices of orders 1, 2 and 3, and kernel PCA is then carried out to create the parameterization given by Equation (6.28) corresponding to these different orders. The Statistical Pattern Recognition Toolbox for Matlab [81] is modified and used for this purpose. Geostatistical simulation can now be accomplished by drawing $\xi$ from the standard normal distribution and obtaining a realization **y**.

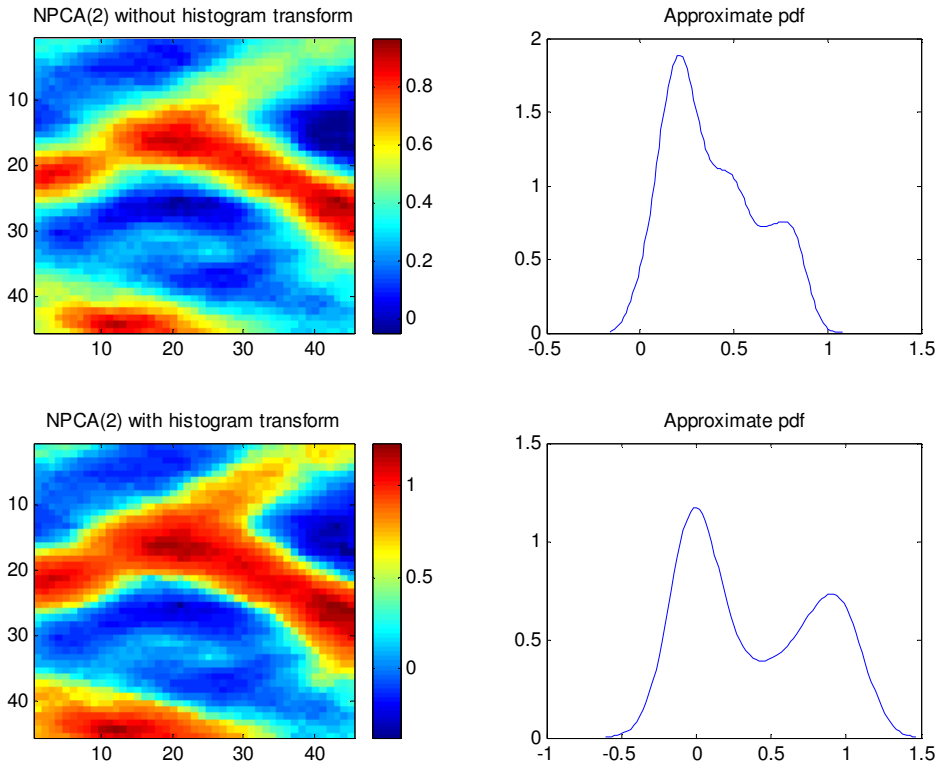


Figure 6-14 Realizations before and after histogram transform

Figure 6-9 to Figure 6-11 illustrate the results of the simulation. In each figure, the left images are simulated realizations and the right plots are the histogram of the corresponding realization. Figure 6-9 shows typical realizations obtained using standard Karhunen-Loeve expansion (kernel matrix of order 1) in the original input space of the realizations $R^{N_C}$, Figure 6-10 shows typical realizations obtained using kernel PCA with the polynomial kernel of order 2 (preserving up to 4$^{th}$ order

statistics), and Figure 6-11 shows that obtained using kernel PCA with the polynomial kernel of order 3 (preserving up to $6^{th}$ order statistics). The number of eigenvalues retained for the parameterization was 30 for all simulations (the maximum number of eigenvalues in this case is $N_R$ = 1000). The same numerical values of the 30 random variables $(\xi)$ drawn from the standard Gaussian distribution were used to create the corresponding realizations in all three figures.

The results clearly indicate that although the standard Karhunen-Loeve expansion produces images that do have a longer correlation length in the direction of the channels, they do not look like channelized models. However, with kernel PCA, the features clearly have a better resemblance to channels, and further, as the order of the polynomial kernel is increased, the channel structure becomes more prominent as expected. Note that although the original realizations are binary and discontinuous in nature (the original realizations as seen in Figure 6-2 consist of only 0s and 1s, where 0 is shale and 1 sand), the realizations constructed with kernel PCA are continuous and smooth due to the fact that the original realizations have higher order statistics than just the $6^{th}$ order, and only 30 eigenvalues have been retained for kernel PCA. In a numerical context, this continuity implies that the parameterization is indeed differentiable with smooth derivatives.

The above parameterization can be further improved to also provide a better approximation of the histograms if necessary. The basic idea is to perform an additional transform on a realization $\mathbf{y}$ obtained using Equation (6.28) to obtain another realization $\tilde{\mathbf{y}} = h(\mathbf{y})$ having a specified target histogram. One approach is to use polynomial chaos expansions, as explained in [83]. Another simpler approach is to apply a *cdf* transform where, if $f_T(\mathbf{y})$ is the target *cdf* and $f_I(\mathbf{y})$ is the initial *cdf*, then $\tilde{\mathbf{y}}$ is obtained as $\tilde{\mathbf{y}} = f_T^{-1} f_I(\mathbf{y})$. This is depicted pictorially in Figure 6-12. The polynomial chaos approach provides the analytical functional relationship $h(\mathbf{y})$ directly, but for this method to work properly, the nature of the input histogram must

be known accurately (for example whether it is Gaussian, uniform etc.). On the other hand, the second approach does not require knowledge of the nature of the input histogram, but it does not provide any analytical functional relationship $h(\mathbf{y})$ (which is required for example to calculate gradients $d\tilde{\mathbf{y}}/d\boldsymbol{\xi}$), and therefore, curve-fitting techniques (like splines, polynomials etc.) have to be applied to obtain the functional form $h(\mathbf{y})$. Figure 6-13 and Figure 6-14 demonstrate the application of the *cdf* transform method on realizations obtained with kernel PCA of order 2, and the modified histograms are clearly better approximations of the original binary histogram. Note that the target histogram in this case is a smooth approximation of the actual binary histogram, obtained using a limited number of eigenpairs for the reconstruction of a typical realization.

### 6.5. Applications to the History Matching Problem

One immediate application of the kernel PCA parameterization of multi-point geostatistics is to the solution of the history matching (or model updating) problem with gradient-based methods. A gradient-based approach was discussed in Chapter 3, in which the history matching problem was defined as a minimization problem, wherein a better estimate of the unknown random field was obtained by minimizing the data mismatch error between actual observed data and data calculated by the simulation model subject to dynamic and geological constraints. The random field to be estimated (permeability field in Chapter 3) was parameterized with the standard Karhunen-Loeve expansion, which allowed for the application of gradient-based minimization algorithms.

If the geological constraints correspond to complex geological structures, however, the Karhunen-Loeve expansion cannot be used for an accurate parameterization, and in such circumstances a kernel PCA parameterization would be more appropriate. Note that there are other existing methods able to preserve geological constraints, such as the probability perturbation method [84] and the gradual deformation method [85], but

these methods, unlike the approach described here, are stochastic rather than gradient-based. Although in general more time consuming, stochastic methods have some advantageous features (e.g., avoidance of local minima). For some applications it may therefore be useful to consider a hybrid algorithm with gradient (kernel PCA) and stochastic components for the history matching problem.

Our interest here is in the application of the kernel PCA parameterization to the history matching problem, which we now consider. The procedure described in Chapter 3 can be followed exactly with the exception that the Karhunen-Loeve expansion is replaced with a kernel PCA parameterization. The final form of the minimization problem is (for additional details, refer to Chapter 3):

$$\min_{\boldsymbol{\xi}} \left[ S = \left\{ \mathbf{y}(\boldsymbol{\xi}) - \mathbf{y}_{prior}(\boldsymbol{\xi}) \right\}^T C_M^{-1} \left\{ \mathbf{y}(\boldsymbol{\xi}) - \mathbf{y}_{prior}(\boldsymbol{\xi}) \right\} + \sum_{n=0}^{N-1} L^n \left( \mathbf{x}^{n+1}, \mathbf{y}(\boldsymbol{\xi}) \right) \right]$$
$$\text{subject to:} \qquad (6.30)$$
$$g^n(\mathbf{x}^{n+1}, \mathbf{x}^n, \mathbf{y}(\boldsymbol{\xi})) = 0 \qquad \forall n \in (0,..,N-1)$$

In the above equation, $S$ is the objective function to be minimized, $\mathbf{C}_M$ is the covariance matrix of the random field $\mathbf{y}$ (obtained numerically from initial realizations of $\mathbf{y}$), $L$ is the usual least-square error between observed and calculated data, $\mathbf{x}$ now represents the dynamic states, the system of equations $g(.)$ represents the dynamic system (i.e., flow simulation), and $n$ is the time step index. The random field $\mathbf{y}$ is a function of independent random variables $\boldsymbol{\xi}$, and this relationship is obtained using kernel PCA. The geological constraints are implicitly honored through this parameterization, and therefore do not appear directly in the minimization problem. Note that the objective function $S$ is obtained from Bayesian inverse theory under the assumption that $\mathbf{y}$ is multi-Gaussian and the dynamics $g(.)$ is linear. Thus, minimizing $S$ in general does not guarantee that the maximum likelihood estimate of $\mathbf{y}$ would be obtained when $\mathbf{y}$ is non-Gaussian or $g(.)$ is highly nonlinear [13].

The applicability of kernel PCA to the history matching problem is demonstrated on a dynamic waterflooding example. The simulation model is that of a simple 2D horizontal square reservoir with one horizontal "smart" water injector on the left edge and one horizontal "smart" producer on the right edge, each having 45 independently controllable segments. The reservoir covers an area of 450×450 $m^2$ and has a thickness of 10 m and is modeled by a 45×45×1 horizontal 2D grid. It is essentially an incompressible two-phase unit mobility oil-water system, with zero connate water saturation and zero residual oil saturation. The injector segments are placed under rate control, and the producer segments are under bottom hole pressure (BHP) control with predefined rates and BHPs. The objective is to estimate the unknown permeability field using observed data that consists of the injector segment BHPs and producer segment watercuts.

Although the permeability field is unknown, it is assumed that we have some prior knowledge of the reservoir which informs us that the reservoir is a fluvial channelized reservoir as depicted by the training image shown in Figure 6-1, with the sand permeability being about 10 Darcy and the background permeability about 500 mD. The contrast in permeability between the high permeability sand and the background reservoir is about a factor of 20. Since this is a validation study, a "true" realization is required, against which the history matching results can be compared. Realization 9 from Figure 6-2 is arbitrarily taken to be the true realization that will provide the observed data.

A kernel PCA parameterization of the permeability field with a polynomial kernel of order 2 is created using the 1000 realizations obtained from the training image as discussed earlier. Only 30 eigenpairs are retained for the parameterization, which corresponds to about 75% of the energy. Histogram transform is not applied in this problem. Note that as the order of the kernel PCA is increased, the parameterization becomes more and more nonlinear, and therefore it becomes more difficult for the gradient-based minimization algorithm to converge. Thus the minimum order kernel

should be chosen that is sufficient to appropriately represent the structures present in the geological model.

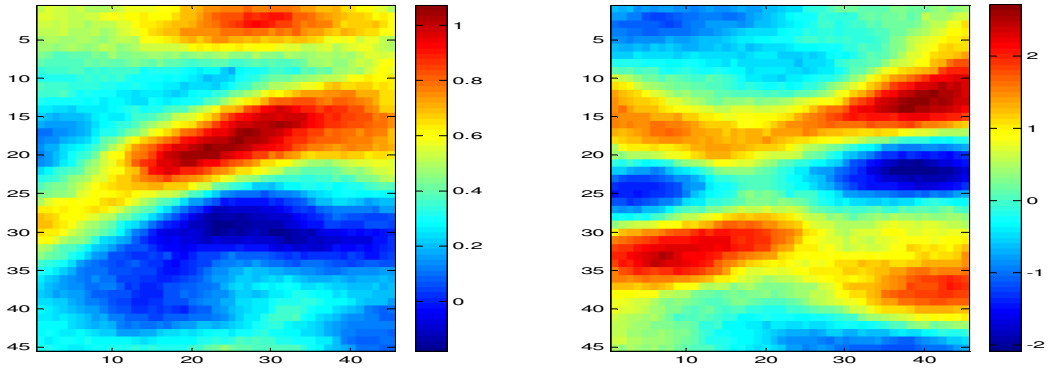

Figure 6-15 Initial guess realization (left) and converged realization (right)

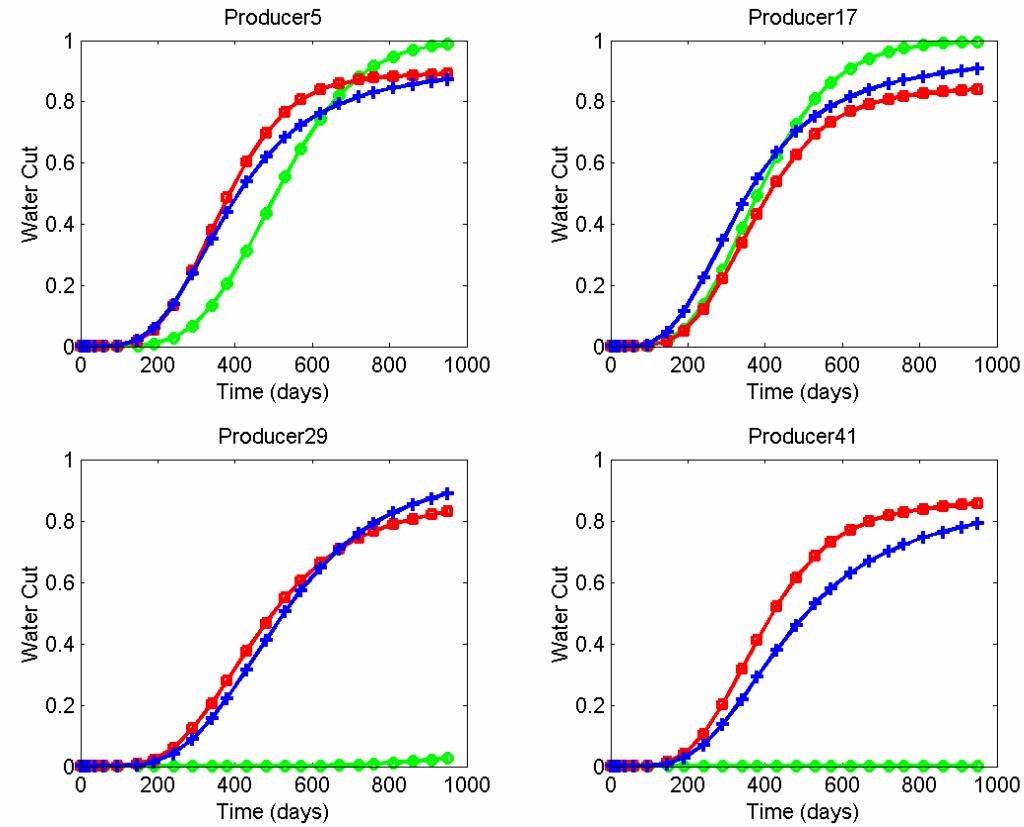


Figure 6-16 Watercut profiles using true (red with squares), initial guess (green with circles) and converged (blue with plus) realizations

Starting from an arbitrary initial guess (Figure 6-15 left), the final converged realization obtained is shown in Figure 6-15 (right). About 10 iterations were required to obtain this final realization, and this corresponds to about 30-40 simulations if

adjoint models are used to calculate the gradients. The final realization obtained clearly resembles the true realization, although this realization is continuous (for reasons discussed earlier) whereas the original realization is binary and discontinuous. The watercut profiles of some of the producer segments are shown in Figure 6-16. The red curves (with squares) are obtained from the true permeability field, the green curves (with circles) are obtained from the initial guess realization, and the blue curves (with plus signs) are obtained from the history matched realization. It is clear that the history match is quite reasonable, thus demonstrating the applicability and efficiency of kernel PCA to the history matching problem.

Further application of kernel PCA within the closed-loop optimal control approach for long-term production optimization is demonstrated in Sarma et al. [86] and also in Chapter 7.

## 6.6. Summary

In this chapter, a novel differentiable parameterization of non-Gaussian random fields (characterized by multi-point geostatistics) was proposed. This parameterization is capable of representing complex geological structures and has many applications including but not limited to closed-loop control, where it can be used within the model updating and uncertainty propagation algorithms. In Chapter 3, the Karhunen-Loeve expansion was used for creating differentiable parameterizations of random fields. This, however, has two major disadvantages rendering it incapable of application to practical large-scale random-fields and associated reservoir simulation models. First, it only preserves the covariance of the random field (two-point statistics), and therefore is only suitable for multi-Gaussian random fields, and cannot be applied to model complex geological structures such as channels. Second, the K-L expansion requires a computationally expensive eigen decomposition of a large covariance matrix of size $N_C \times N_C$, and this is prohibitive for large-scale simulation models even with current computational capability.

It was shown that both these problems could be solved elegantly and efficiently with kernel principal component analysis, which is a nonlinear form of principal component analysis. Using higher order polynomial kernels, multi-point geostatistics can be preserved instead of two-point statistics, thereby creating a differentiable parameterization capable of preserving complex geology. Further, the approach requires an eigen decomposition of a small kernel matrix instead of a covariance matrix, which can be readily accomplished even for large-scale random fields. The application of the parameterization to history matching (using a gradient-based procedure) was demonstrated using a simple dynamic waterflooding example, with the results clearly indicating the efficiency and applicability of the kernel PCA representation. The approach is very general and can be applied to realistic three-dimensional reservoir problems.

# Chapter 7

# 7. Application to a Gulf of Mexico Reservoir

In Chapter 4, a closed-loop control approach was proposed which employed adjoint models and standard gradient-based algorithms for optimization, the Karhunen-Loeve (K-L) expansion and Bayesian inversion theory for model updating, and the K-L expansion and polynomial chaos expansions for uncertainty propagation. There are two main issues limiting the application of that procedure to practical problems. The first is the difficulty in handling nonlinear path constraints during optimization. The second limitation is that the K-L expansion is not able to represent complex geology and is also computationally very expensive. New algorithms to address these issues were discussed in Chapter 5 and Chapter 6.

This chapter modifies the closed-loop algorithm of Chapter 4 with the new algorithms of Chapters 5 and 6 to provide a closed-loop algorithm applicable to practical, large-scale simulation models. Specifically, we apply the approximate feasible direction optimization algorithm of Chapter 5 to efficiently handle path constraints, and kernel principal component analysis (PCA) discussed in Chapter 6 to resolve the issues associated with the K-L expansion. Except for these modifications, the integration of the various algorithms within the closed-loop remains the same as in Chapter 4. The modified closed-loop algorithm is then applied on a sector of the simulation model of a Gulf of Mexico reservoir being waterflooded. The objective is to maximize the expected net present value (NPV) of the sector over a period of eight years by controlling the bottom hole pressures (BHPs) of the injectors and producers. The optimization is subject to production constraints (path constraints) and an uncertain reservoir description. The closed-loop procedure is shown to provide substantial

improvement in NPV over the base case, and the results are very close to those obtained when the reservoir description is known a priori.

## 7.1. Model Description

The simulation model used for this study represents a reservoir located in the Gulf of Mexico, USA. As seen in Figure 7-1, the model consists of a 3D structured grid with 68,800 (86×100×8) cells, out of which about 40,000 are active. The reservoir spans an area of approximately 25 square miles and is located at a depth of about 10,000 ft with an initial reservoir pressure at datum of 5,280 psi. The fluid system is represented by a three-phase black oil model, though it acts as a two-phase system because pressure stays above the bubble point. The reservoir is flanked by an aquifer as seen in Figure 7-1. The relative permeability curves used are the usual Corey type. Capillary pressures are assumed to be zero for modeling purposes. The closed-loop approach is applied on a 46×32×8 sector of this model highlighted by the black rectangle in Figure 7-1.

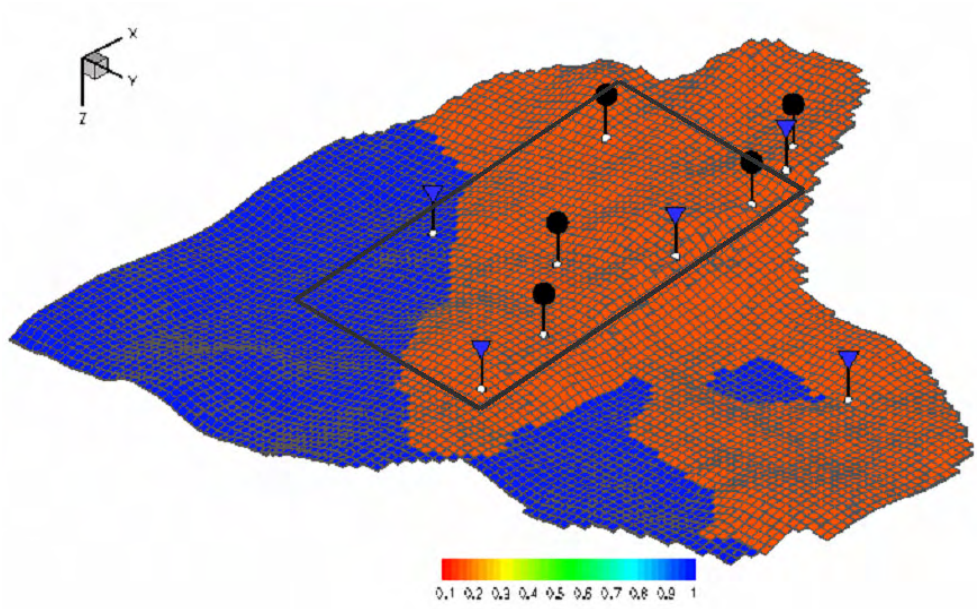


Figure 7-1 Reservoir model with the sector under study highlighted by black rectangle

In order to demonstrate the closed-loop approach, the permeability field is assumed to be unknown and will be updated by assimilating production data. Further, it is

assumed that we have some prior knowledge of the reservoir (for example from outcrop studies), which informs us that the permeability field is moderately heterogeneous, ranging from 10 md to 10 D, with the lower end of the scale corresponding to low permeability shale bodies depicted by the training image shown in Figure 7-2. A multi-point geostatistical software application called *filtersim* [87] is used to generate 120 realizations using this training image. Note that *filtersim* is applied because the training image used in this study is continuous, and other MPS software are not able to handle continuous training images. These realizations are then used to create a kernel PCA parameterization of the permeability field with a polynomial kernel of order 1 (note that, for this case, the geological model is sufficiently simple that we only need to maintain two-point statistics).

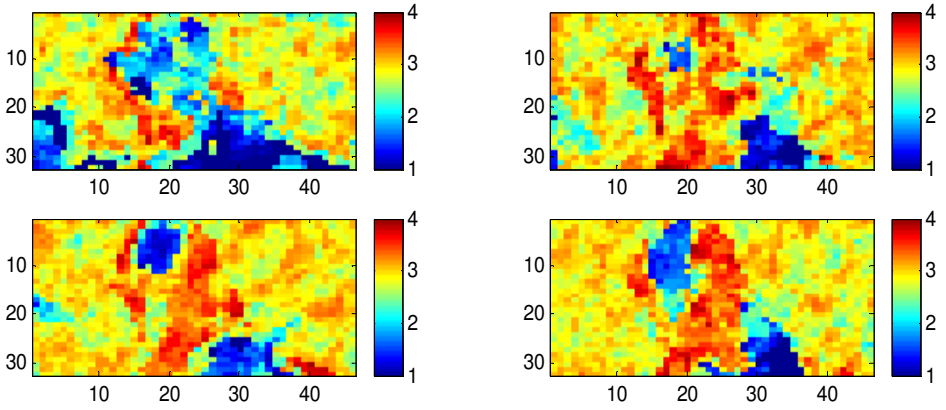


Figure 7-2 Top four layers of the training image used to create kernel PCA parameterization

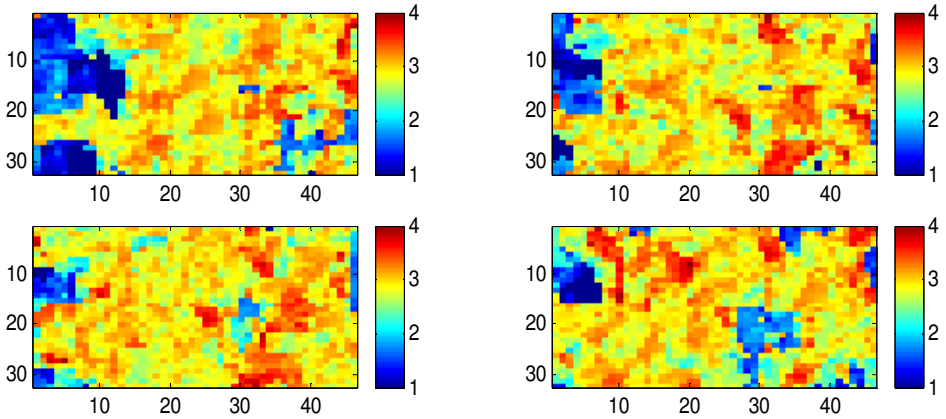


Figure 7-3 Top four layers of the true permeability field of the sector model

In order to validate our closed-loop approach, a "true" realization is required, against which the optimization and model updating results can be compared. The realization shown in Figure 7-3 is taken to be the true model. Note that the training image and the true realization look structurally similar, but the locations of the low permeability bodies are quite different, so the two models will display different flow behavior.

## 7.2. Production Scenario and Constraints

The sector under study has 3 vertical injectors (blue triangles) and 4 vertical producers (black circles), with locations as depicted in Figure 7-1. All wells are perforated in all eight layers of the model. For the purpose of closed-loop optimization, the BHPs of the injectors and producers are used as controls and these are varied in time to maximize the expected NPV of the sector over a period of about eight years (3060 days). This time period is divided into 17 control steps of 180 days each. Thus the total number of controls is equal to $17 \times 7 = 119$. The NPV discounting factor is set at zero, meaning that the effect of discounting is neglected. Thus, maximizing NPV is essentially maximizing cumulative oil production and minimizing cumulative water production. As in Chapter 4, for uncertainty propagation, a Hermite chaos expansion is used to represent NPV. The oil price is conservatively set at \$30/bbl, water injection costs at \$3/bbl, and water production costs at \$6/bbl. The injection rates of the injectors and the watercuts of the producers are the observed data used to update the permeability field. The model updates are performed at 180, 360, 720, 1260 and 2160 days.

Most optimization problems are associated with a set of constraints on the controls. In the absence of such constraints, the solution of the optimization problem is often straightforward. In this case, the BHPs of the wells are bounded below at 4500 psi and above at 9000 psi, which could for example correspond to bubble point pressure and fracture pressure. More importantly, there are two nonlinear inequality path constraints, namely a maximum total injection rate constraint of 20,000 bbl/d and a maximum watercut constraint of 95%.

## 7.3. Base Case Production Strategy

In order to quantify the benefit of any optimization process, it is usual to compare the optimization results against a base or reference case. For production optimization, such a base case would be a reasonable production strategy that an engineer might devise given a simulation model and a set of constraints.

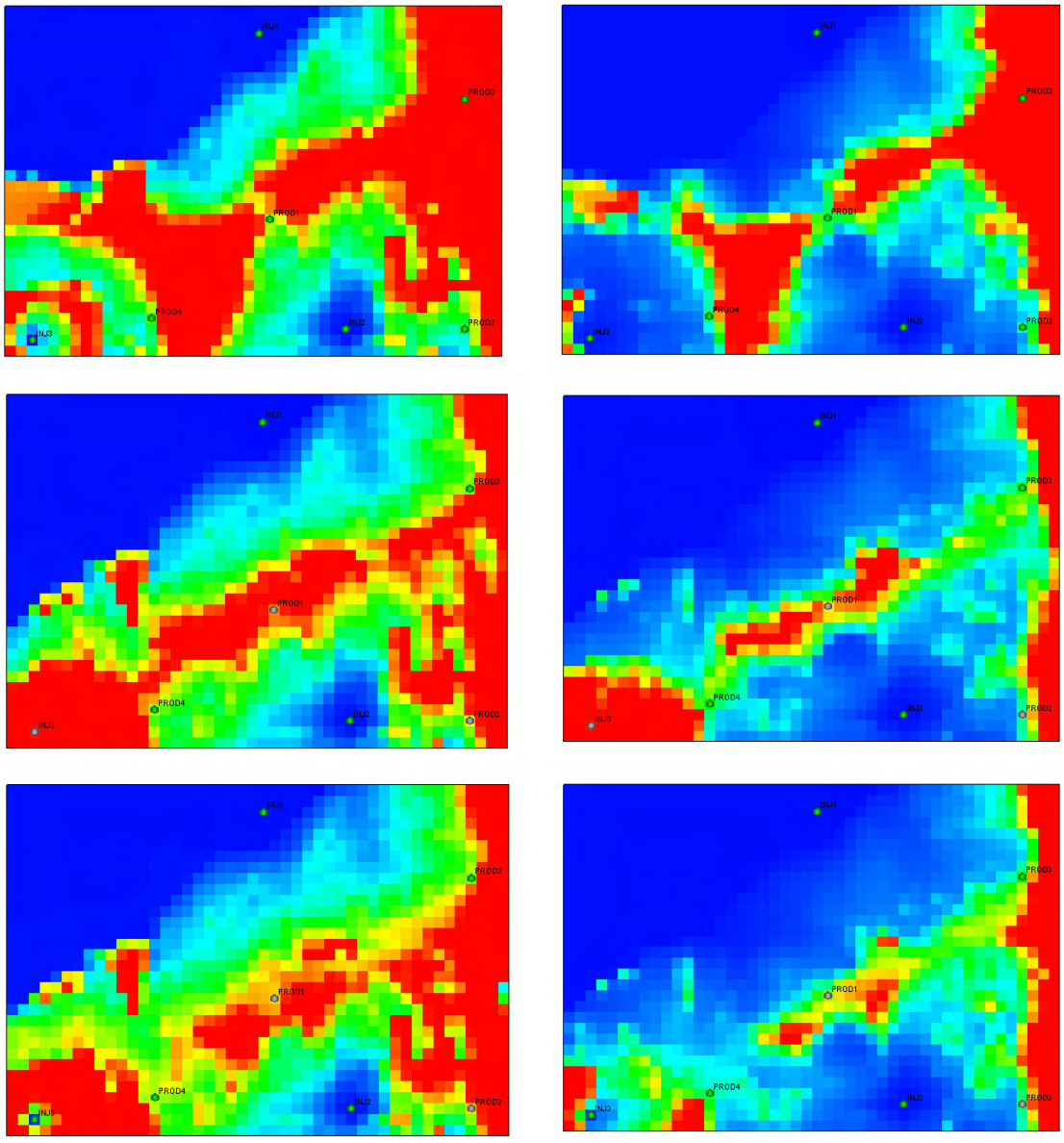


Figure 7-4 Final oil saturations of layers 1 (left) and 2 (right) of the reference case (top), open-loop case (middle) and closed-loop case (bottom); red is oil and blue is water

For the purpose of this case study, the base case is a constant rate/constant BHP production strategy. The 20,000 bbl/d of available injection water is distributed among the three injectors according to their *kh* (product of permeability and pay-thickness), which corresponds to an uncontrolled case. This actually turns out to be a good strategy, as it results in a relatively high sweep (see Figure 7-4 top) and late breakthroughs. The producers are kept fully open, that is, at the minimum allowed BHP of 4,500 psi. In the absence of any other information, this is often a viable approach as it maximizes initial oil production. Overall, the base case represents a reasonable production strategy as demonstrated by the simulation results.

### 7.4. Closed-loop Optimization Results

Figure 7-4 shows the final oil saturation after eight years for layer 1 (left) and layer 2 (right) for the base case (top), open-loop case (middle), and closed-loop case (bottom) starting from an initial condition as seen in Figure 7-1. The open-loop case constitutes an approach wherein the optimization is directly performed on the true permeability field. This approach cannot be applied in reality, as we never have complete knowledge of the reservoir, and thus the closed-loop approach must be used. However, the results of the open-loop approach can be thought of as essentially the best that can be achieved by a closed-loop approach, and it can thus be used as a benchmark against which closed-loop algorithms can be compared.

It is clear that in this case the closed-loop and open-loop sweep efficiencies are very similar, and both lead to a significant improvement in sweep efficiency over the base case. This is also confirmed by the cumulative oil and water production profiles as seen in Figure 7-5. Both the closed-loop and open-loop approaches result in an increase in oil production of approximately 16%, a decrease in water production of approximately 50-55%, and an increase in NPV of 25%. The total water injection remains close to the base case. However, the optimal BHPs obtained with the closed-loop are not exactly the same as those from the open-loop, and this can be understood

by comparing the field watercuts of the two cases as in Figure 7-6. They are however close enough to provide similar sweeps and similar increases in NPV.

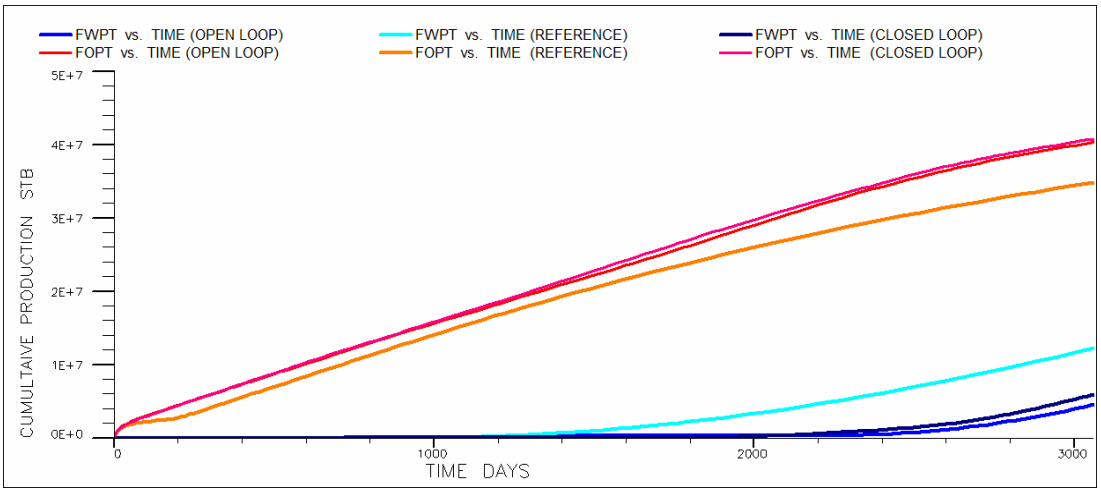


Figure 7-5 Cumulative oil and water production profiles of the reference, open-loop and closed-loop cases

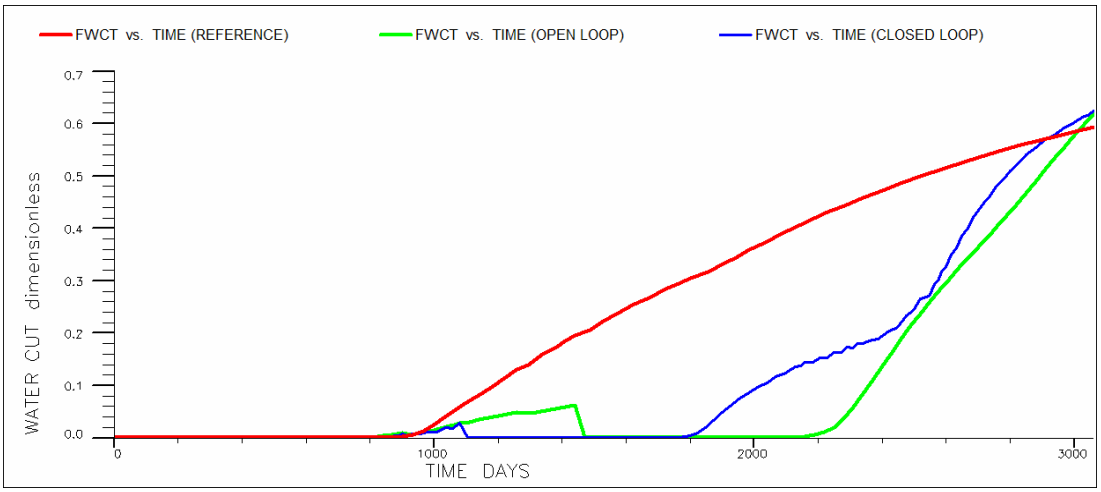


Figure 7-6 Field watercut profiles of the reference, open-loop and closed-loop cases

Starting with a nearly homogeneous permeability field as seen in Figure 7-7, the final updated permeability field is shown in Figure 7-8. Comparing to Figure 7-3, it is clear that the highs and lows in the permeability field are obtained at the correct locations, demonstrating the validity of kernel PCA for model updating. However, the variation in the magnitude of permeability is not as high as in the true permeability field. It is interesting to note that even this approximate permeability field results in nearly correct optimal trajectories. A similar result was also observed in Chapter 4.

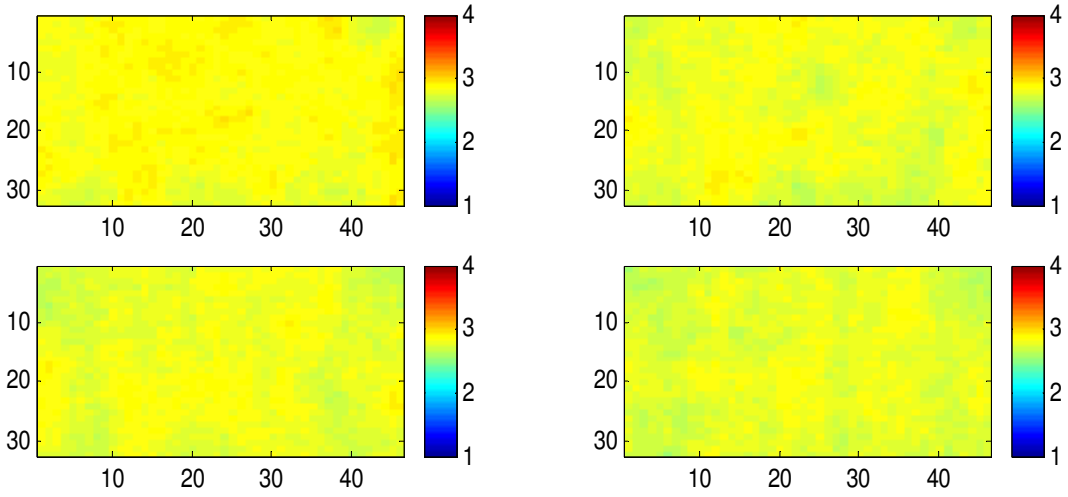


Figure 7-7 Top four layers of the initial permeability field, which is nearly homogeneous

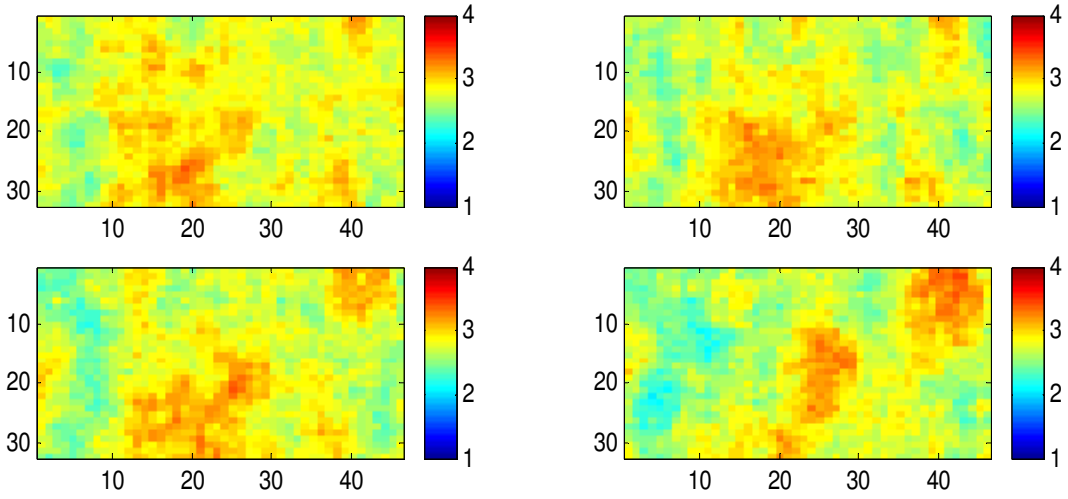


Figure 7-8 Top four layers of the final permeability field obtained after the fifth model update

Figure 7-9 shows the total injection rate (normalized) after closed-loop optimization. It is clear that this constraint is satisfied to within 1% tolerance after optimization, demonstrating the validity of the approximate feasible direction algorithm. Note that after the optimization, the water injection rate remains near the maximum for most of the simulation time, indicating that the optimization essentially results in the time-dependent redistribution of the injected water among the injectors. The watercut constraint never becomes active over the eight years, and therefore does not affect the optimization.

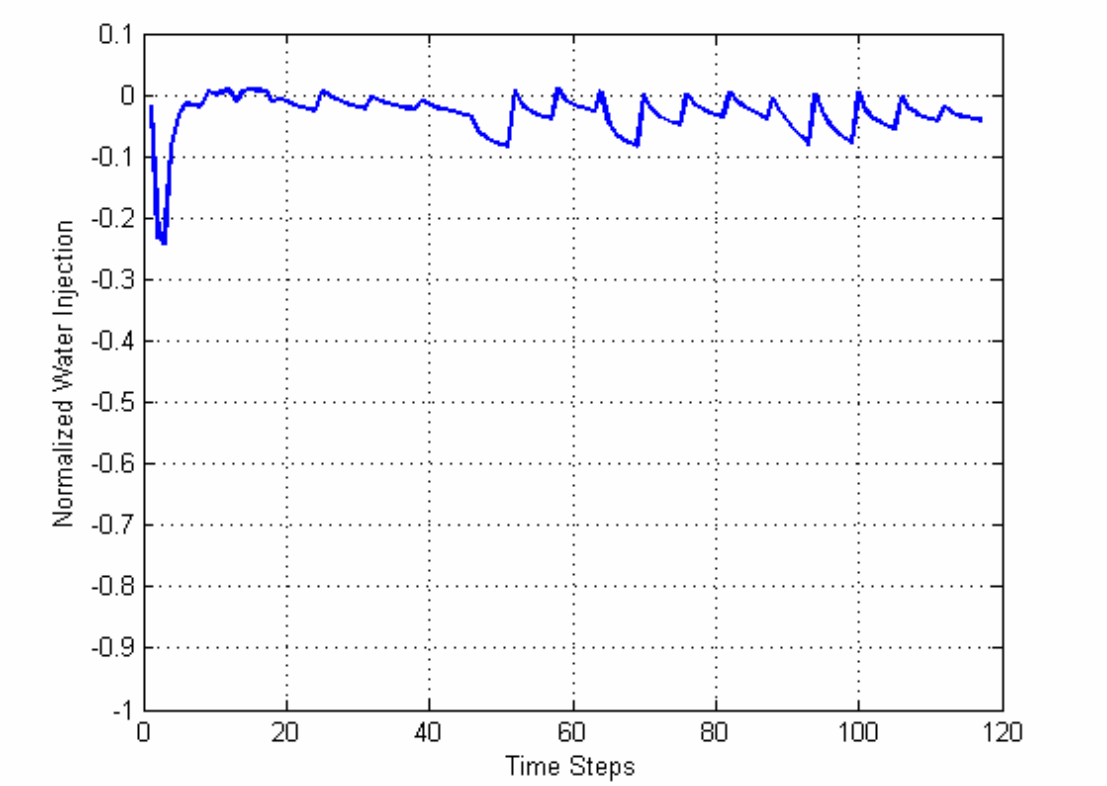


Figure 7-9 Normalized maximum injection rate constraint after optimization

Taken in total, the results presented here demonstrate the applicability of the overall closed-loop procedure to practical reservoir management. This is achieved as a result of the efficiency and robustness of the optimization, history matching and uncertainty propagation algorithms developed and applied in this work.

# Chapter 8

# 8. Conclusions and Recommendations

Since detailed conclusions are provided at the end of each chapter, here we only list the major accomplishments of this work:

1. It was demonstrated that adjoint models provide an efficient method for calculating gradients for the production optimization problem, thereby enabling efficient optimization, especially when the number of controls is large.

2. A modified approach for the construction of the adjoint model applicable to arbitrary level of implicitness of the forward model was also described, and it was shown that with this approach, it is relatively easy to code the adjoint as compared to the standard approach.

3. An efficient model updating algorithm that is able to preserve geological constraints was described. This approach applies the Karhunen-Loeve expansion for parameterizing uncertain reservoir properties, combined with Bayesian inversion theory and adjoint models.

4. Polynomial chaos expansions were applied for uncertainty propagation within the closed-loop. This approach is straightforward to integrate with the other algorithms that comprise the closed-loop, is very efficient compared to standard Monte Carlo techniques, but still allows a “black box” approach.

5. An efficient approximate feasible direction algorithm for handling path inequality constraints during optimization was proposed. The method has many benefits, including guaranteed feasibility and large step sizes during line search.

6. A novel differentiable parameterization of non-Gaussian random fields (characterized by multi-point geostatistics) using Kernel Principal Component Analysis was proposed. This approach addresses the two main limitations associated with the standard Karhunen-Loeve expansion.

7. Application of the proposed closed-loop approach (and its various components) was demonstrated on several dynamic waterflood optimization problems including a realistic example. It was seen that there is a significant potential to maximize production and enhance recovery using the methods developed in this thesis.

Although many of the issues associated with the closed-loop production optimization approach have been addressed in this work, further research is certainly required in many areas, including:

1. The modified adjoint construction approach has only been tested for the FIM approach, and it should be further developed and tested for other implicit levels.

2. The adjoint code and construction procedure should be tested for three-phase and compositional models.

3. Automatic differentiation techniques for constructing the adjoint could be investigated.

4. The efficiency of the proposed model updating algorithm against other methods like the ensemble Kalman filter requires further testing, especially in conjunction with uncertainty propagation.

5. Only production data was assimilated in the examples described in this work. Necessary modifications to the algorithms for assimilating other kinds of data such as 4D seismic have to be investigated.

6. The Probabilistic Collocation Method loses efficiency as the number of input parameters increases, and therefore modifications or other methods may be necessary for efficient uncertainty propagation.

7. It may be useful to test more sophisticated feasible direction algorithms for handling nonlinear path constraints, such as feasible arc Sequential Quadratic Programming.

8. The Kernel PCA parameterization should be applied to other geological models. Its applicability for large-scale history matching and other applications should also be further assessed.

9. One issue with kernel PCA or any other continuous parameterization is that such parameterizations cannot be applied when the input properties are scenario-based and cannot be represented as random fields. Other methods should be investigated to handle such scenario-based properties.

10. This work only addressed the optimization of continuous control variables; however, the more general problem of combined optimization of discrete variables (such as well locations, type etc.) and continuous controls is a more difficult problem and needs further research.

11. The particular closed-loop approach described in this work does not take value of information and risk attitude into account during optimization. Further research is required to develop modifications or other methods to address these issues.

# Appendix A

## A. A General Adjoint for Arbitrary Implicit Level

The formulation of the adjoint system of equations is directly dependent on the forward system of equations, and thus, for the adjoint system to be consistent with the forward system, it has to be derived for the exact set of equations comprising the forward system. For example, an adjoint model derived for the fully implicit (FIM) forward equations cannot be used with an IMPES forward model. In this appendix, an adjoint system of equations is derived that is suitable for any level of implicitness of the forward model including Adaptive Implicit Method (AIM). The sequential implicit schemes are of a different form and are not considered here.

For a general adaptive implicit forward model, the optimal control problem to be solved can be written as follows:

$$\max_{\mathbf{u}^n}\left[ J = \phi\left(\mathbf{x}_p^N, \mathbf{x}_s^N\right) + \sum_{n=0}^{N-1} L^n\left(\mathbf{x}_p^{n+1}, \mathbf{x}_s^{n+1}, \mathbf{u}^n\right)\right] \forall n \in (0,..,N-1)$$

subject to:

$$\begin{aligned} & f_p^n(\mathbf{x}_p^{n+1}, \mathbf{x}_p^n, \mathbf{x}_s^n, \mathbf{u}^n) = 0 && \forall n \in (0,..,N-1) \\ & \mathbf{x}_s^{n+1} = f_s^n(\mathbf{x}_p^{n+1}, \mathbf{x}_p^n, \mathbf{x}_s^n, \mathbf{u}^n) = 0 && \forall n \in (0,..,N-1) \\ & \mathbf{x}_p^0 = \mathbf{x}_{p0};\ \mathbf{x}_s^0 = \mathbf{x}_{s0} && \text{(Initial Condition)} \end{aligned} \qquad \text{(A.1)}$$

Note that additional path constraints are not discussed here, as the modifications necessary to implement them are similar to that of the dynamic system. Here, the dynamic states are $\mathbf{x} = \left[\mathbf{x}_p, \mathbf{x}_s\right]$, where, $\mathbf{x}_p$ are the implicit variables and $\mathbf{x}_s$ are the explicit variables, $f_p$ are the implicit equations and $f_s$ are the explicit equations. In the forward simulation, the implicit equations (for example, the pressure equation in

IMPES) are first solved implicitly for $\mathbf{x}_p^{n+1}$, and then $\mathbf{x}_s^{n+1}$ is solved explicitly using the explicit equations (for example, the saturation equations). As in the FIM approach of Chapter 2, we would also like to perform all the calculations in the forward simulation itself, so that the adjoint can be constructed easily. This is possible for the general adaptive implicit forward system if it is created using the General Formulation Approach [11], as in this approach, the fully implicit Jacobian is always created, and terms are then discarded and the equations decoupled in order to arrive at the IMPES and IMPSAT equations. Note that $L^n$ consists of well terms for the production optimization problem, and thus it is written in the fully implicit form. This is consistent with GPRS [11], in which the wells are always fully implicit.

The augmented cost function for Equation (A.1) is given as:

$$J_A = \phi\left(\mathbf{x}_p^N, \mathbf{x}_s^N\right) + \sum_{n=0}^{N-1}\left[\begin{array}{l} L^n\left(\mathbf{x}_p^{n+1}, \mathbf{x}_s^{n+1}, \mathbf{u}^n\right) + \boldsymbol{\lambda}^{T(n+1)} f_p^n(\mathbf{x}_p^{n+1}, \mathbf{x}_p^n, \mathbf{x}_s^n, \mathbf{u}^n) \\ +\boldsymbol{\mu}^{T(n+1)}\left\{\mathbf{x}_s^{n+1} - f_s^n(\mathbf{x}_p^{n+1}, \mathbf{x}_p^n, \mathbf{x}_s^n, \mathbf{u}^n)\right\} \end{array}\right] \tag{A.2}$$

Note that the terms that are zero by definition are not shown here. $\boldsymbol{\lambda}^{n+1}$ are the Lagrange multipliers associated with the implicit equations $f_p$ and $\boldsymbol{\mu}^{n+1}$ are the Lagrange multipliers associated with the explicit equations $f_s$. The first variation of the augmented cost function $J_A$ is given as:

$$\begin{aligned} \delta J_A = & \left[\frac{\partial \phi}{\partial \mathbf{x}_p^N} + \frac{\partial L^{N-1}}{\partial \mathbf{x}_p^N} + \boldsymbol{\lambda}^{TN}\frac{\partial f_p^{N-1}}{\partial \mathbf{x}_p^N} - \boldsymbol{\mu}^{TN}\frac{\partial f_s^{N-1}}{\partial \mathbf{x}_p^N}\right]\delta\mathbf{x}_p^N + \left[\frac{\partial \phi}{\partial \mathbf{x}_s^N} + \frac{\partial L^{N-1}}{\partial \mathbf{x}_s^N} + \boldsymbol{\mu}^{TN}\right]\delta\mathbf{x}_s^N \\ & + \sum_{n=1}^{N-1}\left[\frac{\partial L^{n-1}}{\partial \mathbf{x}_p^n} + \boldsymbol{\lambda}^{T(n+1)}\frac{\partial f_p^n}{\partial \mathbf{x}_p^n} + \boldsymbol{\lambda}^{Tn}\frac{\partial f_p^{n-1}}{\partial \mathbf{x}_p^n} - \boldsymbol{\mu}^{T(n+1)}\frac{\partial f_s^n}{\partial \mathbf{x}_p^n} - \boldsymbol{\mu}^{Tn}\frac{\partial f_s^{n-1}}{\partial \mathbf{x}_p^n}\right]\delta\mathbf{x}_p^n \\ & + \sum_{n=1}^{N-1}\left[\frac{\partial L^{n-1}}{\partial \mathbf{x}_s^n} + \boldsymbol{\lambda}^{T(n+1)}\frac{\partial f_p^n}{\partial \mathbf{x}_s^n} + \boldsymbol{\mu}^{Tn} - \boldsymbol{\mu}^{T(n+1)}\frac{\partial f_s^n}{\partial \mathbf{x}_s^n}\right]\delta\mathbf{x}_s^n \\ & + \sum_{n=0}^{N-1}\left[\frac{\partial L^n}{\partial \mathbf{u}^n} + \boldsymbol{\lambda}^{T(n+1)}\frac{\partial f_p^n}{\partial \mathbf{u}^n} - \boldsymbol{\mu}^{T(n+1)}\frac{\partial f_s^n}{\partial \mathbf{u}^n}\right]\delta\mathbf{u}^n \end{aligned} \tag{A.3}$$

Again, the terms that are zero by definition are not shown. The adjoint equations can be derived from the above equation, and are given as:

$$\boldsymbol{\mu}^{TN} = -\left[\frac{\partial \phi}{\partial \mathbf{x}_s^N} + \frac{\partial L^{N-1}}{\partial \mathbf{x}_s^N}\right]; \boldsymbol{\lambda}^{TN} = \left[\boldsymbol{\mu}^{TN}\frac{\partial f_s^{N-1}}{\partial \mathbf{x}_p^N} - \frac{\partial \phi}{\partial \mathbf{x}_p^N} - \frac{\partial L^{N-1}}{\partial \mathbf{x}_p^N}\right]\left[\frac{\partial f_p^{N-1}}{\partial \mathbf{x}_p^N}\right]^{-1} \text{(Final Cond.)}$$

$$\boldsymbol{\mu}^{Tn} = \left[\boldsymbol{\mu}^{T(n+1)}\frac{\partial f_s^n}{\partial \mathbf{x}_s^n} - \frac{\partial L^{n-1}}{\partial \mathbf{x}_s^n} - \boldsymbol{\lambda}^{T(n+1)}\frac{\partial f_p^n}{\partial \mathbf{x}_s^n}\right] \forall n = 1,...,N-1 \tag{A.4}$$

$$\boldsymbol{\lambda}^{Tn} = \left[\boldsymbol{\mu}^{T(n+1)}\frac{\partial f_s^n}{\partial \mathbf{x}_p^n} + \boldsymbol{\mu}^{Tn}\frac{\partial f_s^{n-1}}{\partial \mathbf{x}_p^n} - \frac{\partial L^{n-1}}{\partial \mathbf{x}_p^n} - \boldsymbol{\lambda}^{T(n+1)}\frac{\partial f_p^n}{\partial \mathbf{x}_p^n}\right]\left[\frac{\partial f_p^{n-1}}{\partial \mathbf{x}_p^n}\right]^{-1} \forall n = 1,...,N-1$$

After the Lagrange multipliers have been obtained using Equation (A.4), the required gradient of the cost function can be obtained as follows:

$$\frac{dJ}{d\mathbf{u}^n} = \frac{dJ_A}{d\mathbf{u}^n} = \frac{\partial L^n}{\partial \mathbf{u}^n} + \boldsymbol{\lambda}^{T(n+1)}\frac{\partial f_p^n}{\partial \mathbf{u}^n} - \boldsymbol{\mu}^{T(n+1)}\frac{\partial f_s^n}{\partial \mathbf{u}^n} \qquad \forall n = 1,...,N-1 \tag{A.5}$$

This adjoint model can be used for any level of implicitness of the forward model as in an AIM method. It is interesting to note that in the adjoint model, first $\boldsymbol{\mu}^n$ has to be solved before $\boldsymbol{\lambda}^n$ can be solved. As defined before, $\boldsymbol{\mu}^n$ is associated with the explicit equations, and $\boldsymbol{\lambda}^n$ is associated with the implicit equations, and therefore the order of solution is opposite to that of the forward model. This is consistent as the adjoint is solved backwards in time.

As seen from Equations (A.4) and (A.5), in order to calculate the Lagrange multipliers and the final gradient, the following Jacobian matrices are required:

$$\frac{\partial f_p^{n-1}}{\partial \mathbf{x}_p^n}; \frac{\partial f_s^{n-1}}{\partial \mathbf{x}_p^n}; \frac{\partial f_p^n}{\partial \mathbf{x}_p^n}; \frac{\partial f_s^n}{\partial \mathbf{x}_p^n}; \frac{\partial f_p^n}{\partial \mathbf{x}_s^n}; \frac{\partial f_s^n}{\partial \mathbf{x}_s^n}; \frac{\partial f_p^n}{\partial \mathbf{u}^n}; \frac{\partial f_s^n}{\partial \mathbf{u}^n}; \frac{\partial L^{n-1}}{\partial \mathbf{x}_p^n}; \frac{\partial L^{n-1}}{\partial \mathbf{x}_s^n}; \frac{\partial L^n}{\partial \mathbf{u}^n} \tag{A.6}$$

In order to determine how to construct these from the forward equations, it is essential to determine how $f_p$ and $f_s$ are related to the conservation equations.

$$g^n(\mathbf{x}_p^{n+1},\mathbf{x}_p^n,\mathbf{x}_s^{n+1},\mathbf{x}_s^n,\mathbf{u}^n) = F^n(...)+W\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^{n+1},\mathbf{u}^n\right) - \frac{1}{\Delta t^n}\left[A^{n+1}\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^{n+1}\right)-A^n\left(\mathbf{x}_p^n,\mathbf{x}_s^n\right)\right] \tag{A.7}$$

Here $F^n$ refers to the flux terms, $W^n$ refers to the source terms and $A^n$ refers to the accumulation terms. The functionality of $F^n$ is not shown in the above equation, as it depends on the level of implicitness of the grid block. $F^n$ comprises of four terms that are dependent on $\mathbf{x}$.

$$F^n(...) = \Upsilon(...)\mathrm{P}(...)\Psi(...)\Delta\Phi(...) \tag{A.8}$$

$\Upsilon(...)$ represents the transmissibility terms, $\mathrm{P}(...)$ represents the density and mobility terms, $\Psi(...)$ represents the mole fraction terms and $\Delta\Phi$ represents the potential difference terms. The functionality of each term depends on the implicit level of the conservation equations. The equation below demonstrates this functionality for IMPES, IMPSAT, and FIM systems.

$$\begin{aligned}
F^n\left(\mathbf{x}_p^{n+1},\mathbf{x}_p^n,\mathbf{x}_s^n\right) &= \Upsilon\left(\mathbf{x}_p^n,\mathbf{x}_s^n\right)\mathrm{P}\left(\mathbf{x}_p^n,\mathbf{x}_s^n\right)\Psi\left(\mathbf{x}_s^n\right)\Delta\Phi\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^n\right) && (\mathrm{IMPES})\\
F^n\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^n\right) &= \Upsilon\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^n\right)\mathrm{P}\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^n\right)\Psi\left(\mathbf{x}_s^n\right)\Delta\Phi\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^n\right) && (\mathrm{IMPSAT})\\
F^n\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^{n+1}\right) &= \Upsilon\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^{n+1}\right)\mathrm{P}\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^{n+1}\right)\Psi\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^{n+1}\right)\Delta\Phi\left(\mathbf{x}_p^{n+1},\mathbf{x}_s^{n+1}\right) && (\mathrm{FIM})
\end{aligned} \tag{A.9}$$

In GPRS, Local Inversion [11] is used to obtain $f_p^n$ and $f_s^n$ from $g^n$. Thus,

$$f_p^n = g_p^n - \frac{\partial g_p^n}{\partial \mathbf{x}_s^{n+1}}\left(\frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}}\right)^{-1} g_s^n \tag{A.10}$$

Since the Jacobians appearing in the above equations are constant at any given iteration, $f_p^n$ is therefore just a linear combination of the conservation equations $g^n$.

For example, $\partial g_p^n / \partial \mathbf{x}_s^{n+1} \left( \partial g_s^n / \partial \mathbf{x}_s^{n+1} \right)^{-1}$ is a vector for the IMPES model, and is also known as the IMPES Reduction Factor [11, 88]. Also from Cao [11]:

$$f_s^n = \mathbf{x}_s^n - \left( \frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} \right)^{-1} g_s^n - \left( \frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} \right)^{-1} \frac{\partial g_s^n}{\partial \mathbf{x}_p^{n+1}} \left( \mathbf{x}_p^{n+1} - \mathbf{x}_p^n \right) \tag{A.11}$$

Again, the Jacobians in the above equation are constant at any given iteration, and therefore $f_s^n$ is also a linear combination of the conservation equations and $\mathbf{x}$. Thus, the required Jacobians can be obtained as follows:

$$\frac{\partial f_p^n}{\partial \mathbf{x}_p^{n+1}} = \frac{\partial g_p^n}{\partial \mathbf{x}_p^{n+1}} - \frac{\partial g_p^n}{\partial \mathbf{x}_s^{n+1}} \left( \frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} \right)^{-1} \frac{\partial g_s^n}{\partial \mathbf{x}_p^{n+1}} \tag{A.12}$$

This is the simplest Jacobian to construct, as it is the converged implicit Jacobian (for example, of the pressure equation only for IMPES) at the $n^{th}$ time step, and is already decoupled and extracted in the forward run.

$$\frac{\partial f_s^n}{\partial \mathbf{x}_p^{n+1}} = -2 \left( \frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} \right)^{-1} \frac{\partial g_s^n}{\partial \mathbf{x}_p^{n+1}} \tag{A.13}$$

The Jacobians on the right hand side (RHS) of the above equation can be obtained from the converged full Jacobian of the conservations equations at the $n^{th}$ time step. The full Jacobian and the Newton-Raphson iteration used to solve the system are given as:

$$\begin{bmatrix} \dfrac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} & \dfrac{\partial g_s^n}{\partial \mathbf{x}_p^{n+1}} \\ \dfrac{\partial g_p^n}{\partial \mathbf{x}_s^{n+1}} & \dfrac{\partial g_p^n}{\partial \mathbf{x}_p^{n+1}} \end{bmatrix} \begin{bmatrix} \delta \mathbf{x}_s \\ \delta \mathbf{x}_p \end{bmatrix} = - \begin{bmatrix} g_s^n \\ g_p^n \end{bmatrix} \tag{A.14}$$

The remaining four Jacobians of $f_p^n$ and $f_s^n$ with respect to $\mathbf{x}$ are obtained as follows:

$$\frac{\partial f_p^n}{\partial \mathbf{x}_p^n} = \frac{\partial g_p^n}{\partial \mathbf{x}_p^n} - \frac{\partial g_p^n}{\partial \mathbf{x}_s^{n+1}} \left( \frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} \right)^{-1} \frac{\partial g_s^n}{\partial \mathbf{x}_p^n}; \frac{\partial f_p^n}{\partial \mathbf{x}_s^n} = \frac{\partial g_p^n}{\partial \mathbf{x}_s^n} - \frac{\partial g_p^n}{\partial \mathbf{x}_s^{n+1}} \left( \frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} \right)^{-1} \frac{\partial g_s^n}{\partial \mathbf{x}_s^n}$$
$$\frac{\partial f_s^n}{\partial \mathbf{x}_p^n} = -\left( \frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} \right)^{-1} \frac{\partial g_s^n}{\partial \mathbf{x}_p^n} + \left( \frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} \right)^{-1} \frac{\partial g_s^n}{\partial \mathbf{x}_p^{n+1}}; \frac{\partial f_s^n}{\partial \mathbf{x}_s^n} = \mathbf{I} - \left( \frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} \right)^{-1} \frac{\partial g_s^n}{\partial \mathbf{x}_s^n} \tag{A.15}$$

As seen from Equation (A.14), most of the Jacobians on the right hand side of the above equations are obtained from the converged full Jacobian of the conservations equations. However, there are a few terms on the RHS of Equation (A.15) that still remain to be calculated:

$$\frac{\partial g_p^n}{\partial \mathbf{x}_p^n}; \frac{\partial g_s^n}{\partial \mathbf{x}_p^n}; \frac{\partial g_p^n}{\partial \mathbf{x}_s^n}; \frac{\partial g_s^n}{\partial \mathbf{x}_s^n} \tag{A.16}$$

Since $g_p^n$ and $g_s^n$ both represent conservation equations, their form is the same, and therefore, the subscript will be dropped in the following derivation of the above Jacobians. The procedure to calculate $\partial g^n / \partial \mathbf{x}_p^n$ and $\partial g^n / \partial \mathbf{x}_s^n$ depends on the level of implicitness of the grid block. For example, for the FIM case, as seen in Chapter 2:

$$\frac{\partial g^n}{\partial \mathbf{x}_p^n} = \frac{1}{\Delta t^n} \frac{\partial A^n \left( \mathbf{x}^n \right)}{\partial \mathbf{x}_p^n}; \frac{\partial g^n}{\partial \mathbf{x}_s^n} = \frac{1}{\Delta t^n} \frac{\partial A^n \left( \mathbf{x}^n \right)}{\partial \mathbf{x}_s^n} \tag{A.17}$$

These are the derivatives of the accumulation terms of the previous time step. For the IMPES system:

$$\frac{\partial g^n}{\partial \mathbf{x}_p^n} = \Psi \left( \mathbf{x}_s^n \right) \Delta \Phi \left( \mathbf{x}_p^{n+1}, \mathbf{x}_s^n \right) \frac{\partial}{\partial \mathbf{x}_p^n} \left\{ \Upsilon \left( \mathbf{x}_p^n, \mathbf{x}_s^n \right) \mathrm{P} \left( \mathbf{x}_p^n, \mathbf{x}_s^n \right) \right\} + \frac{1}{\Delta t^n} \frac{\partial A^n \left( \mathbf{x}^n \right)}{\partial \mathbf{x}_p^n}$$
$$\frac{\partial g^n}{\partial \mathbf{x}_s^n} = \frac{\partial}{\partial \mathbf{x}_s^n} \left\{ \Psi \left( \mathbf{x}_s^n \right) \Delta \Phi \left( \mathbf{x}_p^{n+1}, \mathbf{x}_s^n \right) \Upsilon \left( \mathbf{x}_p^n, \mathbf{x}_s^n \right) \mathrm{P} \left( \mathbf{x}_p^n, \mathbf{x}_s^n \right) \right\} + \frac{1}{\Delta t^n} \frac{\partial A^n \left( \mathbf{x}^n \right)}{\partial \mathbf{x}_s^n} \tag{A.18}$$

The second terms on the RHS are the same as in the FIM case, and the first terms are the terms that are discarded before IMPES decoupling of the previous time step [11].

The approach remains the same for the IMPSAT grid blocks as the IMPES case. The Jacobians with respect to the controls $\mathbf{u}^n$ can be calculated as follows:

$$\frac{\partial f_p^n}{\partial \mathbf{u}^n} = \frac{\partial g_p^n}{\partial \mathbf{u}^n} - \frac{\partial g_p^n}{\partial \mathbf{x}_s^{n+1}} \left( \frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} \right)^{-1} \frac{\partial g_s^n}{\partial \mathbf{u}^n}$$
$$\frac{\partial f_s^n}{\partial \mathbf{u}^n} = -\left( \frac{\partial g_s^n}{\partial \mathbf{x}_s^{n+1}} \right)^{-1} \frac{\partial g_s^n}{\partial \mathbf{u}^n} \tag{A.19}$$

The only new Jacobians on the RHS of the above equation are $\partial g_p^n / \partial \mathbf{u}^n$, and since $\mathbf{u}^n$ only appears in the well terms:

$$\frac{\partial g^n}{\partial \mathbf{u}^n} = \frac{\partial W^n}{\partial \mathbf{u}^n} \tag{A.20}$$

These terms can be obtained exactly as in the FIM case discussed in Chapter 2. Furthermore, the terms $\partial L^{n-1} / \partial \mathbf{x}_p^n$, $\partial L^{n-1} / \partial \mathbf{x}_s^n$ and $\partial L^n / \partial \mathbf{u}^n$ can also be obtained just as in the FIM case, as *L* is only a function of the well terms.

To conclude, we have seen that all information necessary to construct the adjoint is calculated in the forward run even in the AIM case, provided the General Formulation Approach for constructing the forward system is followed. Also, very importantly, if all these Jacobians are stored from the forward run, it is not necessary to keep track of the implicit levels of the grid blocks for the adjoint run. Further, variable switching is also not an issue as none of the terms of the adjoint is a direct function of $\mathbf{x}$.

# Nomenclature

| | |
|---|---|
| $a$ | Coefficients of polynomial chaos, slack variable |
| $A$ | Accumulation terms of simulation equations |
| $\mathbf{b}$ | RHS of linear constraints |
| $\mathbf{B}$ | Matrix of coefficients for linear constraints |
| $c$ | Nonlinear constraints |
| $C$ | Equivalent constraint |
| $C_A$ | Augmented cost function of constraint |
| $\mathbf{C}$ | Covariance matrix |
| $\overline{\mathbf{C}}$ | Covariance matrix in feature space |
| $C_{w,p}$ | Water production costs per Bbl |
| $d$ | Order of polynomial kernel |
| $\mathbf{d}$ | RHS of linear constraints, observed data calculated by model |
| $\mathbf{d}_{obs}$ | Actual observed data |
| $\mathbf{E}$ | Matrix of eigenvectors |
| $F$ | Flux terms of simulation equations, feature space |
| $g$ | Dynamic system equations |
| $G$ | Jacobian of misfit with respect to parameters |
| $H_n$ | Polynomial chaos functionals |
| $J$ | Cost function |
| $J_A$ | Augmented cost function |
| $\mathbf{K}$ | Kernel matrix |
| $L$ | Lagrangian |
| $LB$ | Lower bounds on controls |
| $\mathbf{m}$ | Uncertain parameters in geological model |
| $N$ | Number of control steps |

| | |
|---|---|
| $N_c$ | Number of components |
| $N_C$ | Number of grid cells of simulations model |
| $N_D$ | Physical dimensions of the model |
| $N_F$ | Dimension of feature space |
| $N_g$ | Number of grid blocks |
| $N_P$ | Number of producers |
| $N_I$ | Number of injectors |
| $N_M$ | Number of eigenpairs retained in the K-L expansion |
| $N_R$ | Number of realizations |
| $N_t$ | Number of time steps |
| $p$ | Number of active constraints |
| $p_b$ | Well block pressure |
| $p_w$ | Well BHP |
| $P$ | Number of terms in polynomial chaos |
| $P_{o,p}$ | Oil price per Bbl |
| $q_{wi}$ | Water injection rate |
| $q$ | Fluid flow rate |
| $r_k$ | Penalty parameter at $k^{th}$ iteration |
| $\mathbf{S}$ | Usable feasible direction, model updating error |
| $s$ | Slack variable |
| $t$ | Time |
| $\mathbf{u}$ | Control vector |
| $UB$ | Upper bounds on controls |
| $\mathbf{v}$ | Eigenvector of covariance matrix |
| $\mathbf{V}$ | Eigenvector in feature space |
| $W$ | Well terms of simulation equations, weighting function |
| $x$ | Spatial variable |

| | |
|---|---|
| $\mathbf{x}$ | Dynamic states |
| $X$ | Output random variable |
| $y$ | Forward model output, spatial variable |
| $\tilde{y}$ | Output calculated by polynomial chaos, approximate realization |
| $\mathbf{y}$ | Realization of random field |
| $Y$ | Output random variable |
| $\mathbf{Y}$ | Feature space vector |

***Greek***

| | |
|---|---|
| $\alpha$ | Discounting factor, tolerance for *max* function approximation |
| $\boldsymbol{\alpha}$ | Eigenvector of kernel matrix |
| $\beta$ | Line search step length, coefficients in pre-image parameterization |
| $\gamma$ | Parameter to maximize |
| $\varepsilon$ | Tolerance for max function approximation |
| $\Delta\Phi$ | Potential difference terms of flux terms |
| $\boldsymbol{\Lambda}$ | Tolerance for max function approximation, diagonal matrix of eigenvalues |
| $\phi$ | Part of cost function |
| $\Phi$ | Nonlinear mapping to feature space |
| $\Phi_k$ | Penalty function at $k^{th}$ iteration |
| $\boldsymbol{\Phi}$ | Matrix of eigenvectors |
| $\lambda$ | Eigenvalues |
| $\boldsymbol{\lambda}$ | Lagrange multipliers |
| $\boldsymbol{\mu}$ | Lagrange multipliers |
| $\rho$ | Fluid density, least square error in pre-image problem |
| $\rho_D$ | Prior observed data distribution |
| $\rho_M$ | Prior model parameters distribution |
| P | Mobility terms of flux terms |
| $\sigma$ | Data standard deviations |

| | |
|---|---|
| $\sigma_M$ | Posterior model parameters distribution |
| $\theta$ | Random event, weight on constraints |
| $\boldsymbol{\Sigma}$ | Diagonal matrix of standard deviations |
| $\Upsilon$ | Function of constraints, transmissibility terms of flux terms |
| $\omega$ | Relaxation factor |
| $\xi$ | Parameters from K-L expansion |
| $\boldsymbol{\xi}$ | Vector of parameters from K-L expansion |
| $\psi$ | Hermite polynomials |
| $\Psi$ | Mole fraction terms of flux terms |

***Subscripts***

| | |
|---|---|
| $D$ | Observed data |
| $H$ | Number of original parameters |
| $i$ | Summation index, iteration index |
| $j$ | Summation index |
| $k$ | Vector component index |
| $K$ | Number of parameters in K-L expansion |
| $M$ | Model parameters |
| $o$ | Oil |
| $p$ | Implicit terms |
| $prior$ | Prior term |
| $s$ | Explicit terms |
| $SC$ | Standard conditions |
| $w$ | Water |
| $\boldsymbol{\xi}$ | K-L expansion parameters |

***Superscripts***

| | |
|---|---|
| $d$ | Polynomial kernel order |
| $j$ | Polynomial kernel order index |

| | |
|---|---|
| $k$ | Newton-Raphson iteration index |
| $m$ | Time level |
| $n$ | Time level |
| $T$ | Transpose of matrix or vector |

***Abbreviations***

| | |
|---|---|
| AIM | Adaptive implicit method |
| BHP | Well bottom hole pressure |
| FIM | Fully implicit method |
| FLPR | Field liquid production rate |
| FOPR | Field oil production rate |
| FOPT | Field oil production total |
| FWCT | Field watercut |
| FWIR | Field water injection rate |
| FWPR | Field water production rate |
| FWPT | Field water production total |
| IMPES | Implicit pressure explicit saturation method |
| IMPSAT | Implicit pressure and saturation method |
| K-L | Karhunen-Loeve |
| KPCA | Kernel principal component analysis |
| NLP | Nonlinear programming |
| NPV | Net present value |
| PCA | Principal Component Analysis |
| PCE | Polynomial chaos expansion |
| PCM | Probabilistic collocation method |
| SQP | Sequential quadratic programming |
| WBHP | Well bottom hole pressure |
| WHP | Well head pressure |
| WOPR | Well oil production rate |
| WWCT | Well watercut |

# References


1. ExxonMobil Corporation, Energy Outlook to 2030, technical report from www.exxonmobil.com/energyoutlook, 2004.
2. ExxonMobil Corporation, Long-term Energy Outlook – An ExxonMobil Analysis, presented at the *ASPO Third International Workshop on Oil and Gas Depletion,* Berlin, Germany, 2004.
3. Rossi, D., Carney, M., Kontchou, J.N., Lancaster, D., McIntyre, S., Reservoir and Production Optimization, white paper from www.slb.com, 2002.
4. Golder Associates, Circle Ridge Fractured Reservoir Project, technical report from www.fracturedreservoirs.com, Redmond, WA, 2000.
5. Brouwer, D.R., *Dynamic Water Flood Optimization with Smart Wells using Optimal Control Theory*, PhD Thesis, Delft University of Technology, Delft, Netherlands, 2004.
6. Yeten, B., *Optimum Deployment of Nonconventional Wells*, PhD Thesis, Stanford University, Stanford, CA, 2003.
7. Konopczynski, M., Design and Modelling of Intelligent Well Downhole Valves for Adjustable Flow Control Using Nodal Analysis, presented at the *SPE ATW Modeling and Optimization of Smart Wells,* Huntington Beach, CA, 2005.
8. Jansen, J.D., Brouwer, D.R., Naevdal, G., van Kruijsdiik, C.P.J.W, Closed-loop Reservoir Management, *First Break*, 23, 43-48, 2005.
9. Brouwer, D.R., Naevdal, G., Jansen, J.D., Vefring, E.H., van Kruijsdiik, C.P.J.W, Improved Reservoir Management Through Optimal Control and Continuous Model Updating, SPE paper 90149 presented at the *SPE Annual Technical Conference and Exhibition*, Houston, TX, 2004.

10. Sarma, P., Aziz, K., Durlofsky, L.J., Implementation of Adjoint Solution for Optimal Control of Smart Wells, SPE paper 92864 presented at the *SPE Reservoir Simulation Symposium*, Houston, TX, 2005.

11. Cao, H., *Development of Techniques for General Purpose Simulators*, PhD Thesis, Stanford University, Stanford, CA 2002.

12. Sarma, P., Durlofsky, L.J., Aziz, K., Chen, W.H., Efficient Real-time Reservoir Management Using Adjoint-based Optimal Control and Model Updating, *Computational Geosciences*, 10, 3-36, 2006.

13. Tarantola, A., *Inverse Problem Theory*, Society of Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005.

14. Loeve, M., *Probability Theory I and II*, 4th Edition, Springer-Verlag, New York, NY, 1977.

15. Gavalas, G.R., Shah, P.C., Seinfeld, J.H., Reservoir History Matching by Bayesian Estimation, *SPE Journal*, 16, 337-350, 1976.

16. Sarma, P., Durlofsky L.J., Aziz, K., Efficient Closed-loop Production Optimization under Uncertainty, SPE paper 94241 presented at the *SPE Europec/EAGE Annual Conference*, Madrid, Spain, 2005.

17. Xiu, D., Karniadakis, G.E., Modeling Uncertainty in Flow Simulations via Generalized Polynomial Chaos, *Journal of Computational Physics,* 187, 137-167, 2003.

18. Isukapalli, S.S., *Uncertainty Analysis of Transport-Transformation Models*, PhD Thesis, Graduate School-New Brunswick, The State University of New Jersey, New Brunswick, NJ, 1999.

19. Tatang, M.A., *Direct Incorporation of Uncertainty in Chemical and Environmental Engineering Systems*, PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1995.

20. Bryson, A.E., Ho, Y.C., *Applied Optimal Control: Optimization, Estimation, & Control*, Taylor & Francis, New York, NY, 1975.

21. Sarma, P., Chen, W.H., Durlofsky L.J., Aziz, K., Production Optimization with Adjoint Models under Nonlinear Control-State Path Inequality Constraints, SPE paper 99959 presented at the *SPE Intelligent Energy Conference and Exhibition*, Amsterdam, Netherlands, 2006.

22. Scholkopf, B., Smola, A., Muller, K., Nonlinear Component Analysis as a Kernel Eigenvalue Problem, *Neural Computation,* 10, 1299–1319, 1998.

23. Sarma, P., Durlofsky, L.J., Aziz, K., Kernel Principal Component Analysis for an Efficient, Differentiable Parameterization of Multi-point Geostatistics, submitted to *Mathematical Geology*, 2006.

24. Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional, New York, NY, 1989.

25. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, UK, 1992.

26. Gill, P.E., Murray, W., Wright, M.H., *Practical Optimization*, Academic Press, New York, NY, 1982.

27. Faithi, Z., Ramirez, W.F., Optimization of an Enhanced Oil Recovery Process with Boundary Controls - a Large Scale Nonlinear Maximization, presented at the *4th IFAC Symposium on Control of Distributed Parameter Systems*, Los Angeles, CA, 1985.

28. Mehos, G., Ramirez, W.F., Use of Optimal Control Theory to Optimize Carbon Dioxide Miscible Flooding Enhanced Oil Recovery, *Journal of Petroleum Science and Engineering*, 2, 247-260, 1989.

29. Liu, W., Ramirez, W.F., Optimal Control of Three-Dimensional Steam Flooding Process, *Journal of Petroleum Science and Engineering,* 11, 137-154, 1994.

30. Zakirov, I.S., Aanonsen, S.I., Zakirov, I.S., Palatnik, B.M., Optimization of Reservoir Performance by Automatic Allocation of Well Rates, presented at the

*5th European Conference on the Mathematics of Oil Recovery*, Leoben, Austria, 1996.

31. Asheim, H., Maximization of Water Sweep Efficiency by Controlling Production and Injection Rates, paper number SPE paper 18365 presented at the *SPE European Petroleum Conference*, London, UK, 1988.
32. Vironovsky, G.A., Waterflooding Strategy Design Using Optimal Control Theory, presented at the *6th European IOR Symposium*, Stavanger, Norway, 1991.
33. Sudaryanto, B., Yortsos, Y.C., Optimization of Displacements in Porous Media Using Rate Control, SPE paper 71509 presented at the *SPE Annual Technical Conference and Exhibition*, New Orleans, LA, 2001.
34. Brouwer, D.R., Jansen, J.D., Dynamic Optimization of Water Flooding with Smart Wells using Optimal Control Theory, SPE paper 78278 presented at the *SPE European Petroleum Conference*, Aberdeen, UK, 2002.
35. Li, R., Reynolds, A.C., Oliver, D.S., History Matching of Three-phase Flow Production Data, *SPE Journal*, 8, 328-340, 2003.
36. Aziz, K., Settari, A., *Fundamentals of Reservoir Simulati*on, Elsevier Applied Science Publishers, New York, NY, 1986.
37. Stengel, R.F., *Optimal Control and Estimation*, Dover Books on Advanced Mathematics, New York, NY, 1985.
38. Luenberger, D.G., *Optimization by Vector Space Methods*, John Wiley and Sons Inc., New York, NY, 1969.
39. Giering, R., Kaminski, T., Recipes for Adjoint Code Construction, *ACM Transactions on Mathematical Software*, 24, 437-474, 1998.
40. Aziz, K., Durlofsky, L., *Fundamentals of Reservoir Simulation*, course notes for PE223, Stanford University, Stanford, CA, 2001.
41. Mathworks, *Opimization Toolbox: For Use with Matlab*, The Mathworks Inc., Natick, MA, 2003.

42. Christie, M.A., Blunt, M.J., Tenth SPE Comparative Solution Project: A Comparison of Upscaling Techniques, *SPE Reservoir Evaluation and Engineering*, 4, 308-317, 2001.

43. Naevdal, G., Johnsen, L.M., Aanonsen, S.I., Vefring, E.H., Reservoir Monitoring and Continuous Model Updating Using Ensemble Kalman Filter, SPE paper 84372 presented at the *SPE Annual Technical Conference and Exhibition*, Denver, CO, 2003.

44. Wen, X.H., Chen, W.H., Real Time Reservoir Model Updating Using Ensemble Kalman Filter, SPE paper 92991 presented at the *SPE Reservoir Simulation Symposium*, Houston TX, 2005.

45. Aitokhuehi, I., Durlofsky, L.J., Optimizing the Performance of Smart Wells in Complex Reservoirs Using Continuously Updated Geological Models, *Journal of Petroleum Science and Engineering*, 48, 254-264, 2005.

46. Caers, J., The Probability Perturbation Method: An Alternative to Traditional Bayesian Approaches for Solving Inverse Problems, presented at the 9th *European Conference on the Mathematics of Oil Recovery*, Cannes, France, 2004.

47. Chen, W.H., Gavalas, G.R., Seinfeld, J.H., Wasserman, M.L., A New Algorithm for Automatic History Matching, *SPE Journal* 14, 593–608, 1974.

48. Chavent, G.M., Dupuy, M., Lemonnier, P., History Matching by Use of Optimal Control Theory, *SPE Journal* 15, 74–86, 1975.

49. Wasserman, M.L., Emanuel, A.S., Seinfeld, J.H., Practical Applications of Optimal Control Theory to History-Matching Multiphase Simulator Models, SPE paper 5020 presented at the *SPE-AIME Annual Fall Meeting*, Houston, TX, 1974.

50. Watson, A.T., Seinfeld, J.H., Gavalas, G.R., Woo, P.T., History Matching in Two-Phase Petroleum Reservoirs, SPE paper 8250 presented at the *SPE-AIME Annual Fall Technical Conference and Exhibition*, Las Vegas, NV, 1979.

51. Wu, Z., Reynolds, A.C., Oliver, D.S., Conditioning Geostatistical Models to Two-Phase Production Data, *SPE Journal*, 4, 142-155, 1999.

52. Wu, Z., Datta-Gupta, A., Rapid History Matching Using a Generalized Travel Time Inversion Method, SPE paper 66352 presented at the *SPE Reservoir Simulation Symposium*, Houston, TX, 2001.

53. Zhang, F., Skjervheim, J.A., Reynolds, A.C., Oliver, D.S., Automatic History Matching in a Bayesian Framework, Example Applications, SPE paper 84461 presented at the *SPE Annual Technical Conference and Exhibition*, Denver, CO, 2003.

54. Oliver, D.S., Multiple Realizations of the Permeability Field From Well Test Data, *SPE Journal*, 1, 145-154, 1996.

55. Reynolds, A.C., He, N., Chu, L., Oliver, D.S., Reparameterization Techniques for Generating Reservoir Descriptions Conditioned to Variograms and Well-Test Pressure Data, *SPE Journal*, 1, 413-426, 1996.

56. Huang, S.P., Quek, S.T., Phoon, K.K., Convergence Study of the Truncated Karhunen-Loeve Expansion for Simulation of Stochastic Processes, *International Journal of Numerical Methods in Engineering,* 52, 1029–1043, 2001.

57. Strebelle, S., Conditional Simulation of Complex Geological Structures using Multiple-point Statistics, *Mathematical Geology*, 34, 1-22, 2002.

58. Lucor, D., Xiu, D., Su, C.H., Karniadakis, G.E., Predictability and Uncertainty in CFD, *International Journal for Numerical Methods in Fluids*, 43, 483-505, 2003.

59. Jardak, M., Su, C.H., Karniadakis, G.E., Spectral Polynomial Chaos Solutions of the Stochastic Advection Equation, *Journal of Scientific Computing*, 17, 319-338, 2002.

*60.* Zhang, D., Lu, Z., An Efficient High-Order Perturbation Approach for Flow in Random Porous Media via Karhunen-Loeve and Polynomial Expansions, *Journal of Computational Physics*, 194, 773-794, 2004.

61. Lu, Z., Zhang, D., A Comparative Study on Uncertainty Quantification for Flow in Randomly Heterogeneous Media using Monte Carlo Simulations, the Conventional

and KL-Based Moment-Equation Approaches, *SIAM Journal of Scientific Computing*, 26, 558-577, 2005.

62. Webster, M., Tatang, M.A., Application of the Probabilistic Collocation Method for an Uncertainty Analysis of a Simple Ocean Model, technical report from the MIT Joint Program on the Science and Policy of Global Change, Cambridge, MA, 1996.

63. Deutsch, C., Journel, A., *GSLIB: Geostatistical Software Library and User's Guide*, second edition, Oxford University Press, New York, NY, 1998.

64. Mehra, R.K., Davis, R.E., A Generalized Gradient Method for Optimal Control Problems with Inequality Constraints and Singular Arcs, *IEEE Transactions on Automatic Control*, 17-1, 1972.

65. Feehery, W.F., *Dynamic Optimization with Path Constraints,* PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1998.

66. Fisher, M.E., Jennings, L.S., Discrete Time Optimal Control Problems with General Constraints, *ACM Transactions on Mathematical Software,* 18-4, 1992.

67. Bryson, A.E., Denham, W.F., Dreyfus, S.E., Optimal Programming Problems with Inequality Constraints I: Necessary Conditions for Extremal Solutions, *AIAA Journal*, 1-11, 1963.

68. Denham, W.F., Bryson, A.E., Optimal Programming Problems with Inequality Constraints II: Solution by Steepest Descent, *AIAA Journal*, 2-11, 1964.

69. Jacobson, D., Lele, M., A Transformation Technique for Optimal Control Problems with a State Variable Inequality Constraint, *IEEE Transactions on Automtic Control*, 5, 457-464, 1969.

70. Rao, S.S., *Optimization: Theory and Applications*, John Wiley and Sons Ltd, New York, NY, 1978.

71. Agkun, M., Haftka, R.T., Wu, K.C., Sensitivity of Lumped Constraints using the Adjoint Method, presented at the *40$^{th}$ AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, St. Louis, MO, 1999.

72. Liu, S.Q., Shi, J., Dong, J., Wang, S., A Modified Penalty Function Method for Inequality Constraints Minimization, from http://www.mmm.muroran-it.ac.jp/~shi/, 2004.

73. Kelley, C.T., *Iterative Methods for Optimization,* Society of Industrial and Applied Mathematics, Philadelphia, PA, 1999.

74. Sengul, M., Yeten, B., Kuchuk, F., The Completion Challenge – Modeling Potential Well Solutions, *Middle East and Asia Reservoir Review,* 5, 2004.

75. Schlumberger, *ECLIPSE 100 Technical Description*, version 2001A, Schlumberger SIS, 2001.

76. Strebelle, S., Journel, A., Reservoir Modeling Using Multiple-Point Statistics, SPE paper 71324 presented at the *SPE Annual Technical Conference and Exhibition*, New Orleans, LA, 2001.

77. Caers, J., Zhang, T., Multiple-Point Geostatistics: A Quantitative Vehicle for Integrating Geologic Analogs into Multiple Reservoir Models, *AAPG Memoir,* 80, 383-394, 2004.

78. Scholkopf, B., Smola, A.J., *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.

79. Scholkopf, B., Mika, S., Smola, A.J., Ratsch, G., Muller, K.R., Kernel PCA Pattern Reconstruction via Approximate Pre-Images, presented at the *8th International Conference on Artificial Neural Networks*, Skovde, Sweden, 1998.

80. Kwok, J.T., Tsang, I.W., The Pre-Image Problem with Kernel Methods, *IEEE Transactions in Neural Networks*, 15, 1517-1525, 2004.

81. Franc, V., Hlavac, V., *Statistical Pattern Recognition Toolbox for Matlab: User's guide,* Czech Technical University, Prague, 2004.

82. Golub, G.H., van Loan, C.F., *Matrix Computations*, The Johns Hopkins Press, Baltimore, MD, 1996.

83. Sakamoto, S., Ghanem, R., Polynomial Chaos Decomposition for the Simulation of Non-Gaussian Nonstationary Stochastic Processes, *Journal of Engineering Mechanics,* 128, 190-201, 2002.

84. Caers, J., Hoffman, T., The Probability Perturbation Method: A New Look at Bayesian Inverse Modeling, *Mathematical Geology*, 38, 81-100, 2006.

85. Hu, L.Y., Blanc, G., Noetinger, B., Gradual Deformation and Iterative Calibration of Sequential Stochastic Simulations, *Mathematical Geology*, 33, 475–489, 2001.

86. Sarma, P., Durlofsky L.J., Aziz, K., Computational Techniques for Closed-Loop Reservoir Modeling with Application to a Realistic Reservoir, to be presented at the *10th European Conference on the Mathematics of Oil Recovery,* Amsterdam, Netherlands, 2006.

87. Zhang, T., *Multiple Point Geostatistics: Filter-based Local Training Pattern Classification for Spatial Pattern Simulation*, PhD Thesis, Stanford University, Stanford, CA, 2006.

88. Coats, K.H., A Note on IMPES and Some IMPES Based Simulation Models, SPE paper 49774 presented at the *SPE Symposium on Reservoir Simulation*, Houston, TX, 1999.