THE UNIVERSITY OF TULSA
THE GRADUATE SCHOOL

DEVELOPMENT OF A PROPOSAL DISTRIBUTION FOR AN EFFICIENT METROPOLIS-HASTING MCMC ALGORITHM FOR THE CHARACTERIZATION OF UNCERTAINTY IN RESERVOIR DESCRIPTION AND PRODUCTION FORECASTS

by
Xin Li

A thesis submitted in partial fulfillment of
the requirements for the degree of Doctor of Philosophy
in the Discipline of Petroleum Engineering

The Graduate School
The University of Tulsa

2017

T H E U N I V E R S I T Y O F T U L S A

THE GRADUATE SCHOOL

DEVELOPMENT OF A PROPOSAL DISTRIBUTION FOR AN EFFICIENT METROPOLIS-HASTING MCMC ALGORITHM FOR THE CHARACTERIZATION OF UNCERTAINTY IN RESERVOIR DESCRIPTION AND PRODUCTION FORECASTS

by
Xin Li

A THESIS

APPROVED FOR THE DISCIPLINE OF

PETROLEUM ENGINEERING

By Thesis Committee

Albert C. Reynolds, Chair
Mustafa Onur
William Coberly
Randy Hazlett

COPYRIGHT STATEMENT

ABSTRACT

Xin Li (Doctor of Philosophy in Petroleum Engineering)

Development of a Proposal Distribution for an Efficient Metropolis-Hasting MCMC Algorithm for the Characterization of Uncertainty in Reservoir Description and Production Forecasts

Directed by Albert C. Reynolds

142 pp., Chapter 6: Conclusions

(396 words)

Generating an estimate of uncertainty in production forecasts has become almost standard in the oil industry but is often done with procedures that yield at best a highly approximate uncertainty quantification. Formally, the uncertainty quantification of a production forecast can be achieved by generating a correct characterization of the posterior probability density function (pdf) of reservoir model parameters conditional to dynamic data and then sampling this pdf correctly. While Markov chain Monte Carlo (MCMC) provides a theoretically rigorous method for sampling any target pdf that is known up to a normalizing constant, in reservoir engineering applications, researchers have found that it may require extraordinarily long chains containing millions to hundreds of millions of states to obtain a correct characterization of the target pdf. When the target pdf has a single mode or has multiple modes concentrated in a small region, it may be possible to implement a proposal distribution based essentially on random walk so that the resulting Markov chain Monte Carlo algorithm based on the Metropolis-Hastings acceptance probability can yield a good characterization of the posterior pdf with a computationally feasible chain length. However, for a high-dimensional complex non-Gaussian pdf, for example, a multimodal pdf with modes separated by large regions of low or zero probability, characterizing the pdf

with MCMC based on random walk is not computationally feasible. While methods such as population MCMC exist for characterizing a multimodal pdf, their computational cost generally makes the application of these algorithm far too costly for field application. In this paper, we design a new proposal distribution based on a Gaussian mixture pdf for use in MCMC where the posterior pdf can be multimodal with the modes spread far apart. Simply put, the method generates modes using a gradient-based optimization method and constructs a Gaussian mixture model to use as the basic proposal distribution. Tests on three simple problems are presented to establish the validity of the method. The performance of the new MCMC algorithm is compared with random walk MCMC and is also compared with population MCMC for a target pdf which is multimodal. In addition, we compared the performance of several MCMC methods including the new developed two-level MCMC method in terms of the quality of uncertainty characterization and computational cost. We use a small but highly nonlinear reservoir model to compare the posterior pdf sampled using different MCMC methods with multiple Markov chains.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my advisor, Dr. Albert C. Reynolds for his guidance, support and patience during my Ph.D. study. I would also like to extend my appreciation to my committee members: Dr. Mustafa Onur, Dr. William Coberly and Dr. Randy Hazlett for their advices and comments on my research.

I am very grateful to all the member companies of the TUPREP, Graduate School and the McDougall School of Petroleum Engineering for all kinds of support provided during my Ph.D. study.

I deeply appreciate all the help, support and encouragement from my friends and colleges.

I must express my gratitude to my husband, Xin Liu, for his continuous love, support and help. I am truly thankful for having him in my life.

This dissertation is dedicated to my parents, for their unconditional love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Numerical reservoir simulation is an important tool for the overall field management and production forecasting. It uses a model of the geological and rock and fluid properties as input into a numerical model to predict the flow of fluids through porous media. In order to find a reservoir model or models which can be used for analysis and prediction, one generates models which minimize the difference between the observed data and the reservoir simulation outputs, a process known as history matching. However, neither static nor dynamic data have sufficient information and accuracy to resolve the features and properties of a complex, heterogeneous reservoir by integrating observed data by history matching, i.e., geological uncertainty in the reservoir model is unavoidable. Therefore, it is essential to quantify the uncertainty in the reservoir description and production forecasts in order to support field development decision making and manage risk.

Bayesian statistics provides a formal tool to treat the uncertainty quantification problem from a rigorous probabilistic point of view. Specifically, given a prior probability density function (pdf) of the reservoir model parameters, the posterior pdf conditional to the observed data can be obtained up to a normalizing constant by multiplying the prior pdf by the function that defines the likelihood of model parameters given the vector of observed data. Thus, quantifying the uncertainty in model parameters is reduced to sampling the posterior pdf [79]. Given a set of samples of this posterior pdf (realizations of the reservoir model), the uncertainty quantification of a production forecast can be generated by simply running the reservoir simulator with each realization.

The randomized maximum likelihood (RML) is one of the earliest methods designed to sample a posterior pdf of reservoir model parameters conditional to observed data. However,

theoretically, RML can only sample correctly only if the the prior model is Gaussian and the predicted data are a linear function of the model parameters. Therefore, RML cannot provide an accurate characterization of the posterior pdf. Recently, ensemble-based methods such as ensemble Kalman filter (EnKF) [26, 9, 48] and ensemble smoother (ES) [97] have become prominent in history-matching. These methods are very attractive because they are computationally efficient and easy to implement. However, none of these methods can sample the posterior pdf accurately when the predicted data are nonlinearly related to the model parameters. Unlike RML and ensemble-based methods, Markov chain Monte Carlo (MCMC) methods are able to sample the posterior pdf correctly as the number of states in the Markov chain goes to infinity [43]. For our history matching problems, to obtain each state in the Markov chain a run of the reservoir simulator is requited which makes MCMC methods very computationally expensive. In addition to the high computational cost problem, MCMC can encounter a local trapping problem, when the target pdf is multi-model.

1.1 Literature Review

1.1.1 Randomized Maximum Likelihood

Oliver et al. [77] presented one of the earliest methods for generating an approximate sampling of the posterior pdf for reservoir model parameters conditional to dynamic data. Their method, which is now known as randomized maximum likelihood (RML), approximates the prior pdf for the vector of reservoir model parameters by a Gaussian and assumes that measurement errors are Gaussian. When the prior pdf is Gaussian, measurement errors are Gaussian and predicted data are a linear function of the model parameters, RML provides a theoretically correct sampling of the posterior pdf [79], and two different proofs have been presented by Reynolds et al. [84] and Oliver [76]. For the nonlinear case, RML is not guaranteed to correctly sample the posterior pdf. More importantly, however, RML is designed to generate samples around the modes of a multimodal pdf which is a highly-desirable characteristic of RML or any other sampling algorithm. Even though RML was motivated by the

work of [96] on Markov chain Monte Carlo, unlike MCMC, there is no guarantee that RML provides a highly accurate characterization of the posterior pdf conditional to observed data and it is possible to design examples where it does not. On the other hand, for multimodal distributions, standard MCMC algorithms can be trapped in a region near a single mode for a long time and generating a correct sampling of the posterior pdf may require chains which contain tens of millions of states even for reasonably simple history-matching problems. Liu and Oliver [59] considered a small 1D single phase reservoir with 20 gridblocks. They use MCMC as the base case and compare five different algorithms to sample the posterior pdf for porosity and permeability conditional to pressure data. For MCMC, they generate a extremely long chain with 320 million states and their results suggest that the mixing of the chain is very slow and samples are correlated over one million iterations. Compared with the results obtained from MCMC, only RML gives a similar uncertainty characterization but with much less computational cost.

RML requires the optimization process to minimize the objective function, for large-scale reservoirs only the gradient-based methods are computationally feasible, such as Gauss-Newton [54] and Levenberg-Marquardt [101, 54]. However, the adjoint gradient which can efficiently give the gradient information is not always available in most of the commercial reservoir simulators. Even if the gradient information is given, the optimization process in RML may require many iterations to converge, thus the computational cost for RML may be high.

1.1.2 Ensemble-based Methods

Recently, it has become popular to use ensemble-based methods for assisted history-matching because they are computationally efficient and easy to adapt to different types of model parameters and to couple with any reservoir simulator. The ensemble Kalman filter (EnKF) [26, 9, 48] was the first such method introduced into the petroleum engineering literature [61, 73, 74]. In [61], EnKF is used to improve the predictions of the pressure behavior of two-phase flow in wellbore. Nævdal et al. [73] use EnKF on near-well reservoir

monitoring, updating permeability fields to forecast the future production. The review paper of Aanonsen et al. [1] gives an overview and a discussion of EnKF's successful application on reservoir problems. It is important to note that, EnKF has been applied successfully to field assisted-history matching problems [21, 28, 5, 44].

EnKF is designed as a parameter-state estimation algorithm but again can only be shown to sample correctly as the size of the ensemble goes to infinity under Gaussian and linear assumptions [94]. The linear-Gaussian assumptions, where the prior model parameters follow a Gaussian distribution and the relationship between model parameters and data is linear, are sufficient to ensure that the ensemble of updated (analyzed) states are statistically consistent with the ensemble of updated parameters (updated vectors of reservoir model parameters), i.e., the ensemble of updated states is statistically consistent with the ensemble of updated states obtained by running the forward model (reservoir simulator) from time zero with each updated vector of parameters. Unfortunately, there exist cases where this statistical consistency is not a reasonable assumption, e.g., when the random vector of reservoir model parameters includes depths of the initial fluid contacts [98] or reservoir structural features [90].

Largely because of this potential inconsistency between updated states and parameters in EnKF, many people have advocated using the ensemble smoother (ES) [97] instead of EnKF. The formulation of ES is similar with EnKF, only it simulates all the available data simultaneously in one step without recursively updating in time. Compared to EnKF, ES is faster, easier to implement (does not require restarts of the reservoir simulator) when applied to reservoir history matching problems. However, the data match obtained with ES is usually inferior to the one generated with EnKF [27]. Emerick and Reynolds [23] also show ES can give worse data match than EnKF on a small highly non-linear 1D water flooding case.

Significant effort has been investigated on "iterative" ES algorithms designed to improve the quality of the data match obtained with ES and its effectiveness in uncertainty quantification. Motivated by the equivalence between single and multiple data assimilation

for linear-Gaussian cases, Emerick and Reynolds [22, 24] proposed the ensemble smoother with multiple data assimilation (ES-MDA). In ES-MDA, one assimilates the same data multiple times with an inflated covariance matrix for measurement error. It has been successfully applied to Brugge field case to history match production data [24], results suggest that the performance of the standard ES is worse than the performance of EnKF and ES-MDA. Moreover, ES-MDA can provide a better data match than EnKF with comparable computational costs. Similar results were obtained by Emerick and Reynolds [25] who presented a field case application of EnKF and ES-MDA for history matching production and seismic data. ES-MDA gives significantly better data matches than EnKF with only 4% higher computational cost. Despite the low computational cost and better data match obtained with ES-MDA, until recently, there was no clear guidance on how to choose the optimal inflation factors for ES-MDA. Le et al. [52] proposed adaptive ES-MDA in which the inflation factors can be chosen by two automatic procedures. The application of the adaptive ES-MDA on a complicated synthetic case suggests that the adaptive ES-MDA methods are superior to the original ES-MDA algorithm in that they can prevent overcorrection of the initial guesses and result in good data matches. The results of Le et al. [52] also suggest that adaptive ES-MDA performs as well or better than the Levenberg-Marquardt form of an iterative ES proposed by Chen and Oliver [15]. Recently, Rafiee and Reynolds [82] developed a method to choose a priori the inflation factors. Their method allows the user to specify the number N_a of inflation factors a priori. As the number of reservoir simulation runs required to condition the N_e realizations in the ensemble of reservoir models to dynamic data by history matching, the total number of reservoir simulation runs required to provide N_e history-matched models using ES-MDA is $N_e N_a$. Thus, specifying N_a a priori allows the reservoir engineer to determine the computational resources that will be necessary to apply ES-MDA. Once N_a is specified, the inflation for the first data assimilation step is determined from the discrepancy principle and the other inflation factors follow a geometric sequence where the geometric ratio is determined such that the sum of the inverse of the N_a inflation factors sum to unity, a condition which is required to ensure that ES-MDA samples correctly in the linear Gaussian

case, see Emerick and Reynolds [24, 22]. The results of Rafiee and Reynolds [82] suggest that ES-MDA based on geometric inflation factors is more robust and reliable than the methods of Le et al. [52] in terms of obtaining a good history match, avoiding large corrections in the prior ensemble of models which produce unrealistically rough maps of rock property field, and avoiding an unrealistic reduction in uncertainty. Moreover, ES-MDA implementation is generally more computationally efficient than the Le et al. [52] method based on using a discrepancy principle or the Iglesias iterative ensemble-based method which is also based on using a inflation factors derived from the discrepancy principle. In one particular example, Rafiee and Reynolds [82] found that the Iglesias algorithm fails to converge in 200 iterations, i.e., a $200 \times N_e$ reservoir simulation runs, where $N_e = 400$. The main competitor to ES-MDA appears to be the Levenberg-Marquardt (LM) form of the iterative ES proposed by Chen and Oliver [15]. As shown by Le et al. [52], however, achieving a good performance with this method is strongly depend on the choice of the initial value of the LM parameters, an improper choice can result in a poor history match and guideline for choosing the initial value does not always result in a good performance; see Le et al. [52]. Most importantly, none of the methods discussed above is theoretically guaranteed to sample the posterior pdf for the case where the relation between predicted data and model parameters are non-linear.

Emerick and Reynolds [23] consider a small highly non-linear 1D water flooding case with 31 gridblocks. They compare the performance of nine different ensemble-based methods in terms of the quality of the data matches, quantification of uncertainty and computational cost. The reference posterior distribution is generated using MCMC with random walk proposal, the length of the Markov chain is 20 million. The results suggest that only RML can give similar data matches as good as those obtained with MCMC. Other non-iterative methods like EnKF and ES did not give acceptable history matches. ES-MDA has the best performance of all the tested iterative ensemble-based methods in terms of the approximation of posterior pdf of reservoir model parameters, data matches and uncertainty quantification. In addition, it can provide a quantification of uncertainty which is fairly similar to those generated with RML and MCMC.

1.1.3 MCMC

MCMC methods are used to sample a pdf by constructing a Markov chain which is a sequence of random variables (states), and the probability of generating a new state in the chain only depends on the current state. Theoretically, Markov chain Monte Carlo (MCMC) methods are able to sample asymptotically the target pdf [29, 95]. Many MCMC methods are developed based on the general framework of the Metropolis-Hastings algorithm [68, 43]. The efficiency of the Metropolis-Hastings algorithm depends on the choice of the proposal distribution; there are two popular choices for the proposal distribution. One of them is called an independent sampler, where the proposal distribution is independent of the current state. The rejection sampling is an independent sampler, Mengersen and Tweedie [67] showed that the choice of proposal distribution in rejection sampling should resemble the target distribution and have longer tails than the target distribution. To have the best performance, the proposal distribution must be a good approximation of the target distribution. The other choice of the proposal distribution is a Gaussian distribution, the mean of the Gaussian distribution is the current state and the covariance matrix (prior covariance matrix for our history matching problem) is fixed. Metropolis-Hastings algorithm with this proposal distribution is often called random walk. A difficult choice for the random walk proposal is the scaling factor for the covariance matrix. Roberts and Tweedie [88] and Roberts and Rosenthal [85] suggest that the scaling factor should be chosen such that the acceptance rate is between 20% and 50%. Gelman et al. [36] indicate that the acceptance rate of 0.234 is optimal as the dimension of the problem approaches infinity, but he also suggests $2.38^2/d$ (d is the dimension of the problem) as the optimal value for the scaling factor.

A special case of Metropolis-Hastings algorithm is the Gibbs sampler [37, 35]. This algorithm generates a component of a new state at one update using the conditional distribution of one component of the state given the rest of the components. One can show that the new state generated from the conditional distribution in the Gibbs sampler has an acceptance probability equal to 1. Thus, the Gibbs sampler is a very efficient sampling method

and it simplifies complex high-dimensional problem by breaking it down to low-dimensional problems. However, for some problems the conditional distribution of one component is very hard to derive. If the conditional distribution of some components cannot be easily obtained, Muller [71, 72] suggests that at any step that Gibbs sampler has difficulty, a Metropolis-Hastings proposal can be used instead. This method is called Metropolis-within-Gibbs (or variable-at-a-time Metroolis-Hastings).

Direct application of MCMC to realistic reservoir engineering history-matching problems is generally very computationally expensive, since it generally requires a minimum of one new run of the reservoir simulation to calculate the acceptance probability of transition from the current state to the proposed new state in the Markov chain. Thus, if one needs to generate millions of states in the burn-in period of a Markov chain before reaching a point where the states in the chain start to represent the target distribution [59, 70, 23], then MCMC becomes implausible for large-scale reservoir models. Emerick and Reynolds [23] applied random walk proposal on a 1D highly non-linear, water flooding, heterogeneous 31 gridblocks reservoir problem, they generated a very long Markov chain with 20 million states to obtain a relatively stable posterior distribution. One way to attempt to improve the computational efficiency of MCMC is to replace the reservoir simulator by a proxy or surrogate model [46, 45, 12] but to the best of our knowledge, there is no rigorous investigation of how much error is introduced in uncertainty quantification when the "true" forward model is replaced by a proxy.

A good MCMC algorithm needs to allow the Markov chain to explore the whole sample space rapidly, i.e., the chain should mix fast. In addition, the Markov chain needs to converge fast, i.e., that the chain should begin sampling from the stationary distribution in a computationally feasible number of iterations. In order to improve the mixing and convergence of the Markov chain, several schemes based on the Metropolis-Hastings algorithm have been proposed. The Hit-and-Run Metropolis algorithm [91, 92, 14] separates the process of generating a new state in the Metropolis-Hastings algorithm into two subprocesses: (i) randomly generate a direction on the unit sphere; (ii) generate a signed distance along

the selected direction then add it to the current state. Berger [4] showed that this algorithm is very useful for problems with a sharply constrained parameter space. Gradient of the target distribution can also be used to accelerate the convergence of the Metropolis-Hastings algorithm. Roberts and Tweedie [88] proposed the Metropolis-Adjusted Langevin Algorithm (or Langevin Monte Carlo) based on the Langevin diffusion process. In this algorithm, new states are proposed using the gradient information of the target pdf, then the new states are accepted or rejected based on the Metropolis-Hastings rule. The gradient information can drive the Markov chain towards the high probability regions, which is the reason why this algorithm has better mixing and converges faster than the Metropolis-Hastings algorithm with random walk proposal. Because computing the gradient of the target pdf can be very expensive, Dostert et al. [18] applied an inexpensive approximation of the gradient based on a coarse scale model to improve the computational efficiency. In addition, Martin et al. [66] proposed a stochastic Newton MCMC method in which the Metropolis-Hastings algorithm is accelerated by proposing a new states from a local Gaussian approximation based on local gradient and Hessian of the objective function evaluated at the current state. To reduce the computational cost of calculating the Hessian, they use a low-rank approximation of the Hessian and apply this method to a problem with 1025 model parameters, their results show that the stochastic Newton MCMC converges faster than the Langevin Monte Carlo. Algorithms that use gradient or Hessian information can generally find a high probability region much faster than the traditional Metropolis-Hastings algorithm, however, for a multi-model problem they could be easily trapped at a local mode, and in this case may converge but only sample part of the target distribution.

For most of the problems, gradient information is not available. Liu et al. [58] proposed a multiple-try Metropolis algorithm which use Monte Carlo samples to approximate the gradient of the target distribution. Instead of generating one new state in the Metropolis-Hastings algorithm, the multiple-try algorithm generates a set of proposals and then selects a good one based on the Metropolis-Hastings rule. The advantage of this algorithm is that it has higher acceptance rate. However, it needs to evaluate the objective function several

times to generate one state in the Markov chain, which adds computational expense. In addition, the reason why this method has a higher acceptance rate and a better mixing of Markov chain could because it utilizes more computations at each iteration.

The Metropolis-Hastings algorithm requires a choice of the proposal distribution, and the efficiency of the Markov chain can be improved by tuning the proposal distribution. Adaptive Metropolis algorithm, which was first proposed by Haario et al. [40], extends the random walk algorithm by adapting the covariance matrix based on all the existing states in the Markov chain. If the covariance matrix is adapted appropriately, the adaptive Metropolis algorithm will converge faster than random walk, because the adaptation process can gather more information about the target pdf. However, the adaptive Metropolis algorithm also suffers from the local trapping problem, i.e., if the target pdf is multi-model, the adaptation can be trapped for a long period at one mode. Roberts and Rosenthal [86] show that the adaptive Metropolis algorithm can converge to the stationary distribution under some assumptions, including that the adaptation should diminish as the number of states in the Markov chain increases. In order to make sure the Markov chain converges to the stationary distribution, Haario et al. [40] suggests that the gain factor of the covariance matrix calculated using the previous states in the chain should be equal to $O(1/n)$, here n is the number of states in the Markov chain. Similar to random walk, it is difficult to tune the scaling factor of the covariance matrix with adaptive Metropolis algorithm. Andrieu and Thoms [3] showed that if the scaling factor is too large or too small, the adaptive Metropolis algorithm will either have a very small or very large acceptance rate, and that this could result in a slow learning process of the covariance matrix. They improved the adaptive Metropolis algorithm by also adapting the scaling factor. In their work, an adaptation process for the scaling factor is added to make the acceptance rate of the Markov chain approach a user defined value. Fossum and Mannseth [30, 31] implemented a different version of the adaptive Metropolis algorithm, in which the proposal distribution is a Gaussian mixture model (GMM) with two different covariance matrices, one is the adapted covariance matrix and the other one is the prior covariance matrix.

All the previous MCMC methods assume the problem is discretized into a finite number of dimensions, so the Markov chain explores a finite dimensional space. The convergence and mixing of the Markov chain generated using all the previous MCMC algorithms are heavily depend on the dimension of the problem, so that they are not computationally feasible for very large scale problems. For a problem in which the unknowns are a continuous functions, all the previous MCMC methods are not useful. Cotter et al. [16] designed a preconditioned Crank-Nicolson MCMC (pCN-MCMC) algorithm which has a convergence property invariant to the dimension of the model parameters when applied to discrete problems, so that pCN-MCMC is potentially very useful for large-dimensional problems. However, like random walk, the pCN-MCMC also has a scaling factor, which can control the performance of this algorithm. If it is too large, the acceptance rate is very low. Otherwise, the chain does not have sufficient mixing. Similar to random walk, the scaling factor can be determined by trial with short chains in order to control the acceptance rate of the Markov chain. Iglesias et al. [50] applied the pCN-MCMC on a reservoir model which has 3600 gridblocks with a total of 110 Markov chains, where each chain contains 500 thousand states. The total computational cost for this case is 55 million forward simulation runs which is very high.

Various authors have tried to improve the computational efficiency of MCMC. As in [78], the idea is that if one applies the Metropolis-Hastings algorithm with a proposal distribution that is close to the target distribution, it is expected that the chain will converge quickly so that we can obtain a correct representation of the target distribution with relatively short chains and thus significantly enhance computational efficiency. Oliver et al. [78] explored different MCMC algorithms to sample the posterior pdf of the permeability field conditional to transient pressure data for a small 2-D single-phase flow reservoir with 225 gridblocks. The proposal distribution is the prior Gaussian pdf or the Gaussian pdf centered at the maximum a posteriori (MAP) estimate with covariance matrix equal to the inverse Hessian matrix. If the proposal distribution given by the MAP estimate is a good approximation of the posterior pdf, then the acceptance probability for a new state proposed from this Gaussian should be higher than the acceptance probability for a new state proposed

from the prior Gaussian distribution. For the case considered in [78], the higher acceptance rate is achieved by incorporating the sensitivity information into the sampling distribution, however, for the most nonlinear case with prior variance of each gridblock log-permeability equal to 1.0, the number of independent states for both local and global perturbations is quite small (14 independent states out of 50,000 proposed transitions and 8 independent states out of 50,000 proposed transitions, respectively) in the Markov chain .

Ma et al. [64, 65] proposed a two-stage MCMC method to sample the posterior pdf of the permeability field conditional to water cut or gas-oil ratio. They reduce the computational cost of random walk by calculating the objective function on a reservoir model with coarser grid. The objective of the method is to sample the posterior pdf on a fine-grid model, for a new proposed permeability field on the fine-grid model, the data match is first calculated on a coarse-grid approximate model using an upscaled permeability field, then use a linear proxy or a nonparametric regression-based statistical function to approximate the data match on the fine-grid. If there is an improvement on the data match compared with the current state in the Markov chain, the fine-grid reservoir simulation is run to calculate the true data match, and the Metropolis-Hastings acceptance criteria is applied to determine whether to accept the new proposed state. Although they improved the computational efficiency, the two-stage MCMC method is indeed random walk, and it can be trapped at a local mode when the posterior pdf has multiple modes. Efendiev et al. [19] proposed a similar approach using streamline simulation to accelerate the random walk process.

Emerick and Reynolds [20] proposed a EnKF-MCMC which combines the EnKF and MCMC methodologies to improve the data matches and obtain a more reliable sampling of the posterior pdf. The proposed algorithm approximates the posterior pdf by a Gaussian distribution with the mean equal to the ensemble mean and the covariance matrix equal to the approximate covariance matrix based on the final ensemble obtained from EnKF. Thus, sampling the posterior distribution of the model parameters is the same as sampling the Gaussian approximation which is easy to sample. Application of the EnKF-MCMC

on a synthetic reservoir model suggests that the EnKF-MCMC gives better uncertainty characterization and data matches than EnKF. However, the computational cost of this method is high especially for a large scale case, because the calculation of the acceptance rate of a new state requires a simulation run to evaluate the objective function. In addition, because the new states are generated from the square root of the approximate covariance matrix, it is possible that the samples in the chain do not represent the posterior pdf correctly. To alleviate those problems, based on the work of Emerick and Reynolds [20], Emerick and Reynolds [22] proposed another relatively efficient EnKF-MCMC algorithm, in which they use a proposed combined parameter-state vector to approximate the likelihood part of the objective function instead of directly run using the reservoir simulator. They also make extra effort to improve the accuracy of the sampling by applying EnKF multiple times from different initial ensembles of state vectors, generating a number of chains from each final ensemble starting from different initial state, and resampling based on the actual values (obtained from reservoir simulation run) of the normalized objective function. The modified EnKF-MCMC algorithm was applied to history match and predict production data on a 3D two-phase case. They found that the posterior pdfs for commutative oil and water production obtained using EnKF-MCMC are very similar to those obtained using a random walk MCMC method with 2 million states in the Markov chain. For the simple 1D reservoir water flooding case considered in [25], the EnKF-MCMC outperforms ES and EnKF, and gives acceptable data matches and uncertainty quantification compared with the results obtained using random walk MCMC with a chain of length 20 million.

In addition to the high computational cost involved in the application of a MCMC algorithm, MCMC can encounter a local trapping problem, where for an exceeding large number of iterations, all states in the chain remain near a local mode and the probability of proposing and accepting a state on a different mode is extremely low. This situation is expected when the posterior distribution has multiple modes which are far from each other and are separated by regions of very low or zero probability. Although a hybrid MCMC algorithm [75] has been developed that conceptually allows the chain to move between high

probability regions widely separated by regions where the value of the target pdf is extremely small or zero, this algorithm can be extremely computationally expensive [6].

Population-based MCMC methods are proposed to alleviate the local-trapping problem, in which several Markov chains are run in parallel. Each of the parallel chains has different but related target pdfs. Parallel tempering is one of the population-based MCMC methods proposed by Geyer [38] and is also known as exchange Monte Carlo [49]. In parallel tempering, each of the parallel chains has a target pdf with a different temperature, the chain sampling the target pdf has the lowest temperature . The idea is that increasing the temperature can flatten the target distribution, by exchanging states in different chains, the chain with the lowest temperature (target pdf) can explore the sampling space more efficiently and converge faster. In practice, the temperature needs to be chosen very carefully in order to have a reasonable acceptance rate of exchanging states in different chains. Motivated by the genetic algorithm [47, 39], Liang and Wong [55] proposed the evolutionary Monte Carlo algorithm (EMC) which combines features in the genetic algorithm with MCMC. Compared to the parallel tempering algorithm, EMC adds a crossover operator which can improve the convergence of the Markov chains. Mohamed et al. [70] modified the population-based MCMC method developed by Liang and Wong [55] and applied it on a challenging problem known as the IC fault model [10] which has a multimodal posterior pdf.

Inspired by information sharing among different modes, Gao et al. [34] developed a distributed parallelized Gauss-Newton (DNG) optimization method to find multiple local minimums of an objective function. Starting from a large number of different base cases, a local quadratic model around each base case is constructed using a Gauss-Newton formulation. Then, based on the minimization of the local quadratic model within a small trust region, a new base case and trust region size are generated. By updating the local base case and local quadratic models iteratively, different local minima are located. The DNG method does not require the derivative of the objective funcion, and compared with the traditional derivative-free optimization algorithms, the DNG method is more efficient. Moreover, it also inherits the robustness from the traditional Gauss-Newton methods. Based on the DNG

optimization method, Gao et al. [33] proposed an approach with low computational cost to generate approximate conditional realizations using the multiple local Gaussian approximation technique. The posterior pdf is approximated using a Gaussian mixture model (GMM) with different weight, which is very easy to sample. The means in the GMM are the local minimums obtained from the DNG, the inverse of the covariance matrices are approximated by the Hessian obtained from the Gauss-Newton approximation along with the prior inverse covariance matrix. Although this sampling method is very efficient, the means and covariance matrices in GMM are only approximations which means the samples from the GMM may not represent the posterior distribution correctly. In addition, for a posterior pdf with multiple modes which are not fully separated, the weight associated with each Gaussian distribution may not be accurate. So that, directly sampling from the GMM may not give a correct representation of the posterior pdf, a resampling process is recommended after sampling the GMM.

Luengo and Martino [63] proposed a fully adaptive Gaussian mixture Metropolis-Hastings algorithm for sampling a non-Gaussian target pdf. In the algorithm of [63], one can ostensibly start with a completely random Gaussian mixture model (GMM) and use the states generated in the Markov chain to continuously adapt the means and covariances of the Gaussians as well as the weight of each Gaussian in the GMM. The objective of their algorithm is similar to the objective of the two-level MCMC algorithm we propose, namely, to find a proposal distribution that is close to the target pdf. The Luengo and Martino [63] algorithm starts with a GMM with arbitrary means whereas our algorithm starts with the mean of each Gaussian equal to a mode of the true target pdf so our initial Gaussian mixture model includes Gaussians with appropriate means which correspond to modes of the target distribution. Thus in the algorithm developed in this dissertation, we start with an initial GMM that is far better calibrated with the target pdf we wish to sample than is the initial GMM in the Luengo and Martino [63] algorithm. For this reason, our two-level MCMC should require shorter chains to obtain an appropriate sampling of the target pdf and thus increase computational efficiency for high-dimensional problems of ultimate

interest in petroleum reservoir history matching and uncertainty characterization. In fact, [63] state that one should not even begin adaptation of the GMM until we have generated at least $100d$ states in the chain where d is the dimension of the problem, which for history matching corresponds to the number of history-matching parameters; thus, according to their $100d$ criteria, their method would not be computationally feasible for a large scale history-matching problem. Note the authors apply their method to only three exceedingly trivial problems containing a maximum of two parameters and Gaussian mixture models consisting of at most six Gaussians yet generate Markov chains of length 5,000 in order to obtain good approximations to the posterior mean; they never demonstrate that they sample the true pdf correctly and, in the examples where the target pdf in a GMM, they always use the same number of Gaussians in the initial GMM used for the proposal distribution that are in the target distribution, whereas, in reality one would not know a priori how many Gaussians to include in the Gaussian mixture model. Finally, the Luengo and Martino [63] procedure requires that the length of the chain be specified a prior whereas in the algorithm presented here, we generate parallel Markov chains and utilize a procedure to monitor the convergence of the chains to the target distribution. Finally, an approximation of one basic algorithmic presented which yields a significant additional increase in computational efficiency.

1.2 Research Objectives

The Metropolis-Hastings algorithm is one of the MCMC methods which can asymptotically sample the target pdf. When the target pdf has a single mode or has multiple modes concentrated in a small region, using Metropolis-Hastings algorithms such as random walk can yield a good characterization of the target pdf. However, such a method may still require on the order of millions of states in the chain to converge to the target pdf. For a multimodal target pdf with modes separated by large regions of low or zero probability, standard Metropolis-Hastings algorithms are not feasible. While methods such as stochastic approximation Monte Carlo (SAMC) [56] and parallel tempering [38] have been proposed for such problems, our test results suggest that these methods either fail or are too expensive

because for our problems to get a state in the Markov chain requires one run of a reservoir simulation.

To solve the aforementioned problems, we intend to develop an efficient MCMC method which can sample a multi-modal target pdf with a plausible low computational cost. More specifically, the objectives of my research were as follows: (1) to develop a computationally efficient MCMC method to generate realizations of reservoir model parameters that can accurately characterize the posterior pdf conditional to observed data. If we can define a proposal distribution which is very similar to the target pdf, and use the Metropolis-Hastings algorithm with that proposal distribution to sample the target pdf, then we intuitively believe that we can sample a multimodal target pdf efficiently; (2) to investigate the performance of several MCMC methods for history matching and uncertainty characterization.

1.3 Dissertation Organization

There are 6 chapters in this dissertation. Chapter 1 gives a brief introduction of history matching and uncertainty characterization, literature review on RML, ensemble-based methods and various MCMC methods, and research objectives. In Chapter 2, the history matching problem in Bayesian framework is discussed, including the construction of the MAP estimate and RML, the theoretical background of MCMC and different MCMC algorithms. Chapter 3 presents the comparison in the quality of data matches, uncertainty characterization and computational cost using several MCMC methods on a small but highly nonlinear 1D water flooding case. In Chapter 4, we present the first major contribution of our work, a two-level MCMC method which generates modes using a gradient-based optimization method and uses the modes and the inverse Hessians at these modes to construct a Gaussian mixture model (GMM) to use as the basic proposal distribution. Via example problems we show that the constructed proposal distribution leads to a Metropolis-Hastings implementation that is far more efficient than traditional MCMC algorithms. As our second contribution, Chapter 5 presents a more computationally efficient approximation of our two-level MCMC method and its application on these four examples considered in Chapter 4.

The final chapter presents conclusions.

CHAPTER 2

THEORETICAL BACKGROUND

2.1 Bayesian Framework of the History Matching Problem

For problems of interest in assessing the uncertainty in reservoir description and performance, Bayes' theorem can be applied to obtain the conditional pdf for the N_m-dimensional vector of model parameters, m, given the vector of N_d-dimensional vector of observed data, d_{obs}. Let $\pi(m)$ or $f(m|d_{\rm obs})$ denote the posterior pdf for m conditional to d_{obs}, Bayes' theorem gives,

$$\pi(m) \equiv f(m|d_{\rm obs}) = \frac{f(d_{\rm obs}|m)f(m)}{f(d_{\rm obs})} = aL(m|d_{\rm obs})f(m). \tag{2.1}$$

Here, $f(m)$ is the prior pdf of the model parameters, $f(d_{obs})$ is the pdf of the observed data, $f(d_{\rm obs}|m)$ is the conditional pdf of d_{obs} given m, $L(m|d_{\rm obs})$ is the likelihood function, and a is a normalizing constant. If we approximate the prior pdf as a Gaussian with mean m_{pr} and covariance matrix C_M, and assume that the measurement error is Gaussian with mean zero and covariance matrix C_D, then following Tarantola [93] or Oliver et al. [79] the posterior pdf, $f(m|d_{\rm obs})$ is given by

$$
\begin{aligned}
&\pi(m) \\
&\equiv f(m|d_{\text{obs}}) \\
&= a \exp\left\{-\frac{1}{2}\left(m-m_{\text{pr}}\right)^T C_{\text{M}}^{-1}\left(m-m_{\text{pr}}\right)^T\right\} \times \exp\left\{-\frac{1}{2}\left(g(m)-d_{\text{obs}}\right)^T C_{\text{D}}^{-1}\left(g(m)-d_{\text{obs}}\right)^T\right\} \\
&= a \exp\left\{-\frac{1}{2}\left(m-m_{\text{pr}}\right)^T C_{\text{M}}^{-1}\left(m-m_{\text{pr}}\right)^T - \frac{1}{2}\left(g(m)-d_{\text{obs}}\right)^T C_{\text{D}}^{-1}\left(g(m)-d_{\text{obs}}\right)^T\right\} \\
&= a \exp\{-O(m)\}
\end{aligned}
\tag{2.2}
$$

[79] where $O(m)$ is the objective function defined by

$$
O(m) = \frac{1}{2}(m-m_{pr})^T C_M^{-1}(m-m_{pr}) + \frac{1}{2}(g(m)-d_{obs})^T C_D^{-1}(g(m)-d_{obs}). \tag{2.3}
$$

If we define the model mismatch part as

$$
O_m(m) = \frac{1}{2}(m-m_{pr})^T C_M^{-1}(m-m_{pr}), \tag{2.4}
$$

and the data mismatch part as

$$
O_d(m) = \frac{1}{2}(g(m)-d_{obs})^T C_D^{-1}(g(m)-d_{obs}), \tag{2.5}
$$

then

$$
O(m) = O_m(m) + O_d(m). \tag{2.6}
$$

Here $g(m)$ is the data predicted from the forward model (reservoir simulator in our applications) given the vector of model parameters m. If $g(m)$ involves no model error, e.g., represents exact physics, the difference between $g(m)$ and d_{obs} is equal to the measurement error.

2.2 The MAP Estimate

The maximum a posteriori (MAP) estimate is the model m_{MAP} that maximizes the pdf of Eq. 2.2, i.e.,

$$m_{\text{MAP}} = \arg\min_{m} O(m). \tag{2.7}$$

Because the forward model $g(m)$ is highly nonlinear, the posterior pdf $f(m|d_{\text{obs}})$ is not a Gaussian. Hence, there is no guarantee that Eq. 2.3 has a unique minimum, it may have multiple local or global minima, i.e., the pdf $\pi(m)$ may have multiple modes.

If the relation between the model parameters and the predicted data is linear, i.e.,

$$g(m) = Gm, \tag{2.8}$$

the posterior pdf given by Eq. 2.2 is Gaussian where G is the $N_d \times N_m$ sensitivity matrix, then the MAP estimate can be found by solving the following equation:

$$0 = \nabla_m O(m) = \left(C_{\text{M}}^{-1} + G^T C_{\text{D}}^{-1} G\right)(m - m_{\text{pr}}) + G^T C_{\text{D}}^{-1}\left(G m_{\text{pr}} - d_{\text{obs}}\right). \tag{2.9}$$

Solving Eq. 2.9 for m, the MAP estimate is given by

$$m_{\text{MAP}} = m_{\text{pr}} - \left(C_{\text{M}}^{-1} + G^T C_{\text{D}}^{-1} G\right)^{-1} G^T C_{\text{D}}^{-1}\left(G m_{\text{pr}} - d_{\text{obs}}\right). \tag{2.10}$$

If we expand $O(m)$ around m_{MAP} usng a Taylor series, we can rewrite $O(m)$ as:

$$O(m) = O(m_{\text{MAP}}) + \left(\nabla_m O(m_{\text{MAP}})\right)^T (m - m_{\text{MAP}}) + \frac{1}{2}(m - m_{\text{MAP}})^T H (m - m_{\text{MAP}}), \tag{2.11}$$

where H is the Hessian matrix given by

$$H = C_M^{-1} + G^T C_D^{-1} G \tag{2.12}$$

If the the relation between the model parameters and the predicted data is linear (Eq. 2.8),

then using $\nabla_m O(m_{\rm MAP}) = 0$, Eq. 2.11 can be further written as

$$O(m) = O(m_{\rm MAP}) + \frac{1}{2}\left(m - m_{\rm MAP}\right)^T H \left(m - m_{\rm MAP}\right). \tag{2.13}$$

Substituting Eq. 2.13 to Eq. 2.2, we can write the posterior pdf as

$$\begin{aligned} f(m|d_{\rm obs}) &= a \exp\left\{-O(m_{\rm MAP})\right\} \exp\left\{-\frac{1}{2}\left(m - m_{\rm MAP}\right)^T H \left(m - m_{\rm MAP}\right)\right\} \\ &= \widehat{a} \exp\left\{-\frac{1}{2}\left(m - m_{\rm MAP}\right)^T H \left(m - m_{\rm MAP}\right)\right\}. \end{aligned} \tag{2.14}$$

Here, $\widehat{a}$ is a normalizing constant. Thus, for linear case (Eq. 2.8), $f(m|d_{\rm obs})$ is a Gaussian with mean given by $m_{\rm MAP}$ and covariance matrix given by

$$\begin{aligned} C_{\rm MAP} &= H^{-1} \\ &= \left(C_{\rm M}^{-1} + G^T C_{\rm D}^{-1} G\right)^{-1} \\ &= C_{\rm M} - C_{\rm M} G^T \left(G C_{\rm M} G^T + C_{\rm D}\right)^{-1} G C_{\rm M}, \end{aligned} \tag{2.15}$$

If the relation between the model parameters and the predicted data is linear (Eq. 2.8), then the minimum of $2O(m)$ has a χ^2 distribution with N_d degrees of freedom [79], i.e., the mean is N_d and the variance is $2N_d$. It is reasonable to assume the result is approximately correct for nonlinear case, so that $O(m_{\rm MAP})$ should satisfy

$$N_d - 5\sqrt{2N_d} \leq O(m_{\rm MAP}) \leq N_d + 5\sqrt{2N_d}. \tag{2.16}$$

2.3 The Randomized Maximum Likelihood (RML) Estimation

For uncertainty analysis, it is desirable to have multiple realizations of the reservoir model. One of the earliest method for generation of an approximate sampling of the posterior pdf represented by Eq. 2.2 was proposed by Oliver et al. [77]. This method is now known as randomized maximum likelihood (RML) which can efficiently generate an approximation sampling of the posterior pdf. However, RML is guaranteed to generate to generate theoret-

ically correct samples of the posterior pdf only when the prior pdf is Gaussian, measurement errors are Gaussian and predicted data are linearly related to the model parameters [79]. Although for nonlinear case RML is not guaranteed to sample correctly, there are works [77, 23, 84] that shows RML can give an approximate sampling in some cases. In addition, the most important characteristic of RML is that it is designed to sample from different modes of the posterior distribution. Suppose N_e is the number of samples, RML generates an approximate sampling of the posterior pdf of Eq. 2.2 as follows:

RML Algorithm

Do $j = 1, 2, \cdots, N_e$.

1. Generate realizations m_{uc} from the prior Gaussian $\mathcal{N}(m_{pr}, C_M)$ using

$$m_{\text{uc,j}} = m_{\text{pr}} + C_M^{1/2} Z_M^j, \tag{2.17}$$

here Z_M^j is a column vector of independent standard random normal deviates with N_M (number of model parameters) dimensions. $C_M^{1/2}$ is the square root of C_M. If the Cholesky decomposition of C_M is $C_M = LL^T$, then $C_M^{1/2} = L$.

2. Generate realizations d_{uc} from $\mathcal{N}(d_{obs}, C_D)$ using

$$d_{\text{uc,j}} = d_{\text{obs}} + C_D^{1/2} Z_D^j, \tag{2.18}$$

here Z_D^j is a column vector of independent standard random normal deviates with N_D (number of observed data) dimensions. $C_D^{1/2}$ is the square root of C_D. If C_D is a diagonal matrix, then the square root $C_D^{1/2}$ can be found by simply replacing each diagonal entry of C_D by its square root. More generally, we let $C_D^{1/2}$ denote the symmetric square root of C_D which can be found from its spectral decomposition.

3. Minimize the following objective function to obtained a realization $m_{uc,j}$.

$$O_j(m) = \frac{1}{2}\left(m - m_{\mathrm{uc},j}\right)^T C_{\mathrm{M}}^{-1}\left(m - m_{\mathrm{uc},j}\right) + \frac{1}{2}\left(g(m) - d_{\mathrm{uc},j}\right)^T C_{\mathrm{D}}^{-1}\left(g(m) - d_{\mathrm{uc},j}\right). \quad (2.19)$$

With two quite different approaches, Oliver et al. [79] and Reynolds et al. [84] prove this procedure can sample the posterior pdf correctly in the linear Gaussian case. The result of Oliver et al. [79] is more general as it allows for uncertainty in the prior mean. Follow the RML algorithm, a sample of the posterior pdf is given by

$$m_c = \arg\min_m O_r(m), \quad (2.20)$$

where the $O_r(m)$ is defined as

$$O_r(m) = \frac{1}{2}\left(m - m_{\mathrm{uc}}\right)^T C_{\mathrm{M}}^{-1}\left(m - m_{\mathrm{uc}}\right) + \frac{1}{2}\left(g(m) - d_{\mathrm{uc}}\right)^T C_{\mathrm{D}}^{-1}\left(g(m) - d_{\mathrm{uc}}\right), \quad (2.21)$$

here m_{uc} and d_{uc} are realizations from $\mathcal{N}(m_{pr}, C_M)$ and $\mathcal{N}(d_{obs}, C_D)$, respectively. Follow Eq. 2.16, the objective function $O(m_c)$ should satisfy [79]

$$N_d - 5\sqrt{2N_d} \leq O(m_c) \leq N_d + 5\sqrt{2N_d} \quad (2.22)$$

or equivalently,

$$1 - 5\sqrt{\frac{2}{N_d}} \leq \frac{O(m_c)}{N_d} \leq 1 + 5\sqrt{\frac{2}{N_d}}. \quad (2.23)$$

2.4 Markov Chain Monte Carlo (MCMC)

RML and ensemble-based history matching methods such as ES, EnKF are computationally efficient but cannot generally characterize the uncertainty accurately when the data is not linearly related to the model parameters. MCMC is a procedure for generating samples of a target pdf, the most important characteristic for MCMC method is that under reasonably assumptions, it is able theoretically to asymptotically sample the posterior pdf; however, MCMC methods are normally very computationally expensive.

A Markov chain is a sequence of random vectors $\{m_n\}_{n\geq 0}$ where the probability of the next state depends only on the current state, i.e.,

$$\mathrm{P}(m_n|m_0, m_1, m_2, ..., m_{n-1}) = \mathrm{P}(m_n|m_{n-1}) \tag{2.24}$$

2.4.1 Metropolis-Hastings Algorithm

A properly defined Markov Chain Monte Carlo (MCMC) algorithm can sample a target probability density function (pdf) by constructing Markov chains. Throughout $\pi(m)$ denotes the target pdf which is the probability distribution for the random vector m. In cases of interest to us, $\pi(m)$ represents the posterior distribution of the model m (vector of model parameter) conditional to a vector of observed data d_{obs}. In most of the MCMC algorithms, the proposed new state in the chain depends on the current state, and the proposed state will be accepted or rejected as the next state in the chain based on an acceptance criteria. Throughout, $q(m, \widetilde{m})$ is the proposal distribution (pdf), i.e., "the probability" of proposing a transition from the state m in the chain to the state $\widetilde{m}$. Note this means the proposed state, $\widetilde{m}$, is obtained by sampling $q(m, \widetilde{m})$. Throughout, $\alpha(m, \widetilde{m})$ denoted the acceptance probability which is the probability of accepting the new proposed state, $\widetilde{m}$; i is the index of the chain, i.e., m_i denotes the ith state in the chain. The most popular MCMC algorithm is the Metropolis-Hastings algorithm [68, 43], which is given below.

Algorithm 2.1: Metropolis-Hastings Algorithm

1. Set $i = 0$ and choose the initial state m_0.

2. Propose a new state $\widetilde{m}_{i+1}$ by sampling the proposal distribution $q(m_i, \widetilde{m}_{i+1})$.

3. Evaluate the acceptance probability (pdf) at $\widetilde{m}_{i+1}$ by the Metropolis-Hastings condition given by

$$\alpha(m_i, \widetilde{m}_{i+1}) = \min\Big\{1, \frac{\pi(\widetilde{m}_{i+1})q(\widetilde{m}_{i+1}, m_i)}{\pi(m_i)q(m_i, \widetilde{m}_{i+1})}\Big\}. \tag{2.25}$$

4. Generate a random number u from the uniform distribution $U(0, 1)$.

5. If $u \leq \alpha(m_i, \widetilde{m}_{i+1})$, the new proposed state is accepted and we set $m_{i+1} = \widetilde{m}_{i+1}$. Otherwise, repeat the current state in the chain, i.e., set $m_{i+1} = m_i$.

6. Set $i = i + 1$ and return to step 2 until the chain has converged and we have obtained the number of samples desired.

It is important to note that the acceptance probability requires calculating the ratio $\pi(\widetilde{m}_{i+1})/\pi(m_i)$, and this ratio can be calculated without knowledge of the normalizing constant which is an important advantage of the Metropolis-Hastings algorithm. In our history matching problems, after using Bayes' theorem, the target pdf $\pi(m)$ has the form of $\pi(m) = f(m|d_{obs}) = a\exp(-O(m))$ where a is the normalizing constant and $O(m)$ is the objective function which typically involves the sum of a data mismatch term and a model mismatch term, see Eq. 2.11.

From Eq. 2.24, we define the transition probability $p_{i,j}$=P$(m_j \,|\, m_i)$. A state m in a Markov chain is said to have period k where k is a positive integer if given $m_n = m$ at any state n, the probability of returning to state m at state $n+i$ is zero when $i < k$ and nonzero for $i = k$. If $k = 1$ for all states in the chain, then the chain is said to be aperiodic. The chain is said to be irreducible if starting from any state, it is possible to reach any other state in a finite number of transitions. If a Markov chain is aperiodic, irreducible and satisfies

$$\sum_i \pi(m_i) p_{i,j} = \pi(m_j), \tag{2.26}$$

the chain will converge to a unique stationary (invariant) distribution π starting from any initial state as the length of chain goes to infinity. Metropolis et al. [68] introduced a stronger condition, known as the detailed balance which is given by

$$\pi(m_i) p_{i,j} = \pi(m_j) p_{j,i}. \tag{2.27}$$

Summing the detailed balance equation over i and using $\sum_i p_{ji} = 1$, it is easy to see that if the detailed balance holds, then Eq. 2.26 holds. Metropolis et al. [68] also suggested use

$p_{i,j} = \alpha(m_i, m_j)q(m_i, m_j)$, while Hastings [43] proved that the chain will converge to the stationary distribution π if we define

$$\alpha(m_i, m_j) = \min\left(1, \frac{\pi(m_j)q(m_j, m_i)}{\pi(m_i)q(m_i, m_j)}\right),$$

and the chain is aperiodic and irreducible. Note the preceding equation is Eq. 2.29 in the Metropolis-Hastings algorithm (Algorithm 2.1).

One of the simplest proposal distributions is the random walk [62], in which the new state in the Markov chain is sampled from a Gaussian distribution centered at the current state m_k, with a scaled covariance matrix $\sigma^2 C_M$, i.e., $q(m_k, \widetilde{m}_{k+1}) = N(m_k, \sigma^2 C_M)$, where C_M is the prior covariance matrix, and σ is a scaling factor with $\sigma \leq 1$. Note that σ controls the size of perturbation, i.e., the magnitude of $\widetilde{m}_{k+1} - m_k$ and the performance of the Markov chain [89]. Generally, if the value of σ is too high, the acceptance rate will be very low and it will require an extremely long chain to generate a sufficient number of samples to represent the target pdf. As the value of σ decreases, the acceptance rate tends to increase, but if σ is too small, the chain may not be well mixed and may require excessively long chain to characterize the posterior pdf. The optimal choice of σ is case-dependent, but in general an optimal value of σ should lead to a asymptotically optimal acceptance rate equal to 0.234 under quite general conditions [36, 87]. Brooks et al. [8] suggests that for problems with dimension 1, the optimal acceptance rate is approximately 0.44. For our test problems, we chose the value of σ based on some experiments with short chains in order to attempt to find a value of σ that gives an acceptance rate roughly equal to 0.234. It is also known from Roberts et al. [87] and Roberts and Rosenthal [85] that $\sigma = \frac{(2.38)^2}{d}$ is optimal for large dimensional problems, here d is the dimension of the problem.

In random walk, the proposal distribution is symmetric, i.e., $q(m_k, \widetilde{m}_{k+1}) = q(\widetilde{m}_{k+1}, m_k)$. Therefore, the acceptance probability of Eq. 2.29 simplifies to

$$\alpha(m_k, \widetilde{m}_{k+1}) = \min\left\{1, \frac{\pi(\widetilde{m}_{k+1})}{\pi(m_k)}\right\}. \tag{2.28}$$

While the Metropolis-Hastings algorithm using random walk will theoretically converge, it may require chains which contain millions of states to obtain a reasonable approximation to the target pdf [59, 23]. Thus, as noted earlier, our primary objective is to develop an MCMC algorithm that can generate a correct sampling using chains of length on the order of 10,000 or less as opposed to algorithms that require chain lengths to be on the order of millions or tens of millions to obtain a reasonable approximation of the target pdf [59, 23].

Algorithm 2.2: Random Walk Metropolis-Hastings Algorithm

1. Set $i = 0$ and choose the initial state m_0.

2. Propose a new state $\widetilde{m}_{i+1}$ by sampling the proposal distribution $N(m_i, \sigma^2 C_M)$.

3. Evaluate the acceptance probability (pdf) at $\widetilde{m}_{i+1}$ by the Metropolis-Hastings condition given by

$$\alpha(m_i, \widetilde{m}_{i+1}) = \min\left\{1, \frac{\pi(\widetilde{m}_{i+1})}{\pi(m_i)}\right\}. \tag{2.29}$$

4. Generate a random number u from the uniform distribution $U(0, 1)$.

5. If $u \leq \alpha(m_i, \widetilde{m}_{i+1})$, the new proposed state is accepted and we set $m_{i+1} = \widetilde{m}_{i+1}$. Otherwise, repeat the current state in the chain, i.e., set $m_{i+1} = m_i$.

6. Set $i = i + 1$ and return to step 2 until the chain has converged and we have obtained the number of samples desired.

2.4.2 Gibbs Sampler

The Metropolis-Hastings algorithm requires a user defined proposal distribution, which can affect the efficiency of the sampling. Gibbs sampling is a method that does not need to specify the proposal distribution; all the samples are proposed from conditional distributions with the acceptance probability equal to 1. Suppose the model parameter m has N_M dimensions, and $m = \{m_1, m_2, \cdots, m_{N_M}\}$. Let $f_k(m_k \mid m_1, \cdots, m_{k-1}, m_{k+1}, \cdots, m_{N_M})$ denote the full conditional distribution of m_k given $\{m_1, \cdots, m_{k-1}, m_{k+1}, \cdots, m_{N_M}\}$

Algorithm 2.3: Gibbs Sampler

1. Set $i = 0$ and choose the initial state $m_0 = \{m_{0,1}, m_{0,2}, \cdots, m_{0,N_M}\}$.

2. Set $i = i + 1$, generate $m_{i,1} \sim f_1(m_1 \mid m_{i-1,2}, \cdots, m_{i-1,N_M})$.

3. Generate $m_{i,2} \sim f_2(m_2 \mid m_{i-1,1}, m_{i-1,3}, \cdots, m_{i-1,N_M})$.

$\vdots$

N_M. Generate $m_{i,N_M} \sim f_{N_M}(m_{N_M} \mid m_{i-1,1}, m_{i-1,2}, \cdots, m_{i-1,N_M-1})$.

N_{M+1}. Return to step 2 until we have obtained the number of samples desired.

Although Gibbs sampler does not need to specify a proposal distribution, Gibbs sampler requires a method to sample from the conditional pdf. In addition, if the model parameters are highly correlated, it is very difficult to change the value of one parameter without changing the others.

2.4.3 Adaptive MCMC

The Metropolis-Hastings algorithm requires an input of the proposal pdf, and the efficiency of the chain depends on the choice of the proposal pdf. If the proposal pdf is the target distribution itself, then every new state generated from the proposal pdf will be accepted. However, the point of MCMC is that we cannot directly sample from the target pdf. Haario et al. [40] proposed an adaptive MCMC algorithm in which the proposal pdf is estimated using all the available samples in the Markov chain. One of the adaptive MCMC algorithm is given below. The proposal distribution in this algorithm has the same form as the random walk proposal, but the covariance matrix in the proposal distribution and the mean which is used to calculate the covariance matrix adapt as the chain evolves. In this algorithm, C_{Mi} denotes the covariance matrix at the ith iteration (ith state in the chain), initially it is the prior covariance matrix for our history matching problem. μ_i denotes the mean which is used to calculate the updated covariance matrix, it is set to the initial state in the chain at the beginning of the algorithm.

Algorithm 2.4: Adaptive Metropolis Algorithm

1. Set $i = 0$ and choose the initial state m_0.

2. Propose a new state $\widetilde{m}_{i+1}$ by sampling the proposal distribution $N(m_i, \sigma^2 C_{Mi})$.

3. Evaluate the acceptance probability (pdf) at $\widetilde{m}_{i+1}$ by the Metropolis-Hastings condition given by

$$\alpha(m_i, \widetilde{m}_{i+1}) = \min\left\{1, \frac{\pi(\widetilde{m}_{i+1})q(\widetilde{m}_{i+1}, m_i)}{\pi(m_i)q(m_i, \widetilde{m}_{i+1})}\right\}. \tag{2.30}$$

4. Generate a random number u from the uniform distribution $U(0, 1)$.

5. If $u \leq \alpha(m_i, \widetilde{m}_{i+1})$, the new proposed state is accepted and we set $m_{i+1} = \widetilde{m}_{i+1}$. Otherwise, repeat the current state in the chain, i.e., set $m_{i+1} = m_i$.

6. Update the mean and covariance matrix using

$$\mu_{i+1} = \mu_i + \beta_{i+1}(m_{i+1} - \mu_i) \tag{2.31}$$

$$C_{Mi+1} = C_{Mi} + \beta_{i+1}((m_{i+1} - \mu_{i+1})(m_{i+1} - \mu_{i+1})^T - C_{Mi}). \tag{2.32}$$

7. Set $i = i + 1$ and return to step 2 until the chain has converged and we have obtained the number of samples desired.

The term β_i is a gain factor sequence, it is assumed that β_i satisfies the following condition:

$$\sum_{i=1}^{\infty} \beta_i = \infty \quad \text{and} \quad \sum_{i=1}^{\infty} \beta_i^{1+\delta} < \infty, \tag{2.33}$$

for some $\delta \in (0, 1]$ in order to show that the Markov chain converges to its stationary pdf [40]. Haario et al. [40] suggests setting $\beta_i = O(\frac{1}{i})$.

The adaptive Metropolis algorithm can be improved in various ways. Similar to the random walk, the efficiency of the chain could be affected by the choice of the scaling factor σ. If σ is too large or too small, the algorithm will either have a very small or a very large acceptance rate which could affect the efficiency of the chain. Because the covariance matrix in the proposal distribution is changing as the chain evolve, using trial runs with short chains can only estimate a scaling factor for the initial covariance matrix. Andrieu and Thoms [3]

proposed an improved adaptive Metropolis algorithm in which the scaling factor and the covariance matrix are adapted simultaneously in order to make the acceptance rate of the chain reach a desired value. At the ith iteration, the scaling factor σ_i is updated by

$$\log(\sigma_{i+1}) = \log(\sigma_i) + \beta_{i+1}(\alpha_i - \alpha^*). \tag{2.34}$$

Here, α^* is the desired acceptance rate, α_i is the acceptance probability given by Eq. 2.30 at the ith iteration, β_{i+1} is the gain factor sequence. In the early stage of the chain, this adaptation of the scaling factor can be very useful. As the chain evolves, the gain factor β_{i+1} can be very small so that the scaling factor does not change.

2.4.4 Newton MCMC Method

In Section 2.2, we showed that if the relation between the model parameters and the predicted data is linear (Eq. 2.8), then the posterior pdf $f(m|d_{\text{obs}})$ is a Gaussian distribution. However, for the nonlinear case, the posterior pdf is no longer a Gaussian. At a given point m_i, the quadratic approximation of the objective function is given by

$$\begin{aligned}
Q_i(m) &\approx Q_i(m_i) + g_i^T(m - m_i) + \frac{1}{2}(m - m_i)^T H_i(m - m_i) \\
&= Q_i(m_i) + \frac{1}{2}g_i^T(m - m_i) + \frac{1}{2}(m - m_i)^T g_i + \frac{1}{2}(m - m_i)^T H_i(m - m_i) + g_i^T H^{-T} g_i - g_i^T H^{-T} g_i \\
&= Q_i(m_i) + \frac{1}{2}g_i^T H_i^{-T} H_i(m - m_i) + \frac{1}{2}(m - m_i)^T H_i H_i^{-1} g_i \\
&\qquad + \frac{1}{2}(m - m_i)^T H_i(m - m_i) + g_i^T H_i^{-T} g_i - g_i^T H_i^{-T} g_i \\
&= Q_i(m) + \frac{1}{2}(m - (m_i - H_i^{-1}g_i))^T H_i(m - (m_i - H_i^{-1}g_i)) - g_i^T H_i^{-T} g_i \\
&= Q_i(m) + \frac{1}{2}(m - (m_i - H_i^{-1}g_i))^T H_i(m - (m_i - H_i^{-1}g_i)) + const
\end{aligned} \tag{2.35}$$

where $g_i = g(m_i) = \nabla O(m_i)$ and $H_i = \nabla^2 O(m_i)$. Eq. 2.35 shows the minimizer of $Q_i(m)$ is given by $m_{i+1} = m_i - H_i^{-1}g_i$. Inspired by the local quadratic approximation of the objective

function, in Newton MCMC method, the proposal distribution is given by

$$q(\widetilde{m}_{i+1}|m_i) = \mathcal{N}(m_i - \sigma_1 H_i^{-1} g_i, \sigma^2 H_i^{-1}), \tag{2.36}$$

The Newton MCMC algorithm is given below, unlike random walk, the proposal distribution in the Newton MCMC algorithm is not symmetric, so that the acceptance probability is calculated using Eq. 2.38. Note that design a chain that has the desired acceptance rate, i.e., between 20% and 50%, we add two scaling factors σ and σ_1 in Eq. 2.37. Both of the scaling factors can be tuned based on the acceptance rate of the chain. To estimate an approximate value of σ and σ_1 that roughly has acceptance rate 0.2, we run several experimental chains with different values of the scaling factors, each chain has 2000 states, then we use the value of scaling factors associate with the chain that has the desired acceptance rate.

Algorithm 2.5: Newton MCMC Algorithm

1. Set $i = 0$ and choose the initial state m_0.
2. Propose a new state $\widetilde{m}_{i+1}$ using

$$\widetilde{m}_{i+1} = m_i - \sigma_1 H_i^{-1} g_i + \sigma\xi, \tag{2.37}$$

where $\xi \sim \mathcal{N}(0, H_i^{-1})$.

3. Evaluate the acceptance probability at $\widetilde{m}_{i+1}$ by the Metropolis-Hastings condition given by

$$\alpha(m_i, \widetilde{m}_{i+1}) = \min\left\{1, \frac{\pi(\widetilde{m}_{i+1})q(\widetilde{m}_{i+1}, m_i)}{\pi(m_i)q(m_i, \widetilde{m}_{i+1})}\right\}. \tag{2.38}$$

4. Generate a random number u from the uniform distribution $U(0, 1)$.
5. If $u \leq \alpha(m_i, \widetilde{m}_{i+1})$, the new proposed state is accepted and we set $m_{i+1} = \widetilde{m}_{i+1}$. Otherwise, repeat the current state in the chain, i.e., set $m_{i+1} = m_i$.
6. Set $i = i + 1$ and return to step 2 until the chain has converged and we have

obtained the number of samples desired.

This method can improve the convergence of the chain by sampling new states from a local Gaussian approximation, however, this method also can be trapped at a local mode when the posterior pdf is multi-model.

2.4.5 Population MCMC Method

When the target pdf is complex and multimodal, the standard Metropolis-Hastings algorithms can be trapped for a very long time near a single mode, and thus have difficulty sampling the whole distribution. In order to sample a target pdf which has complex response surface and multiple modes, Liang and Wong [55] proposed an algorithm named population Monte Carlo (or evolutionary Monte Carlo). This algorithm is derived from the conceptual idea inherent in the genetic algorithm and simulated annealing, and it works by running several separate parallel Markov chains with different temperatures in parallel. Chains with higher temperature allow the state to move more freely in the sampling space, and different chains interact by exchanging states, so the chain which samples the target pdf can escape from a local mode more easily. Mohamed et al. [70] implemented the population MCMC method, and applied it on the IC-fault model.

Within the Bayesian framework, the target pdf that we want to sample is the posterior distribution given by the equation

$$f(m|d_{obs}) = CL(m|d_{obs})f(m), \tag{2.39}$$

where m is the model parameter, $L(m|d_{obs})$ is the likelihood of m given observed data, C is the normalising constant and $f(m)$ is the prior pdf form. For the IC fault model considered in this paper, the objective function does not have the model mismatch part, which is similar to assuming the prior distribution is uniform.

For population MCMC, in order to sample a target distribution $f(m)$, one sample an

augmented system with the distribution

$$f(m(1), m(2), \ldots, m(N)) = \prod_{k=1}^{N} f_k(m(k)), \tag{2.40}$$

where $\{m(1), m(2), \ldots, m(N)\}$ is a population, N is the population size which is also the total number of parallel chains each with a different temperature. Note here one of the f_k is the target pdf. Define $t = (t_1, \ldots, t_N)$ with $t_1 = 0 < \cdots < t_N = 1$ is the temperature ladder associated with parallel chains, i.e., t_k is associated with chain k. The target pdf for the sequence of chains is given by the equation

$$f_k(m(k)|d_{obs}) = C_k(t_k)(L(m(k)|d_{obs}))^{t_k} f(m(k)), \tag{2.41}$$

where $m(k)$ is the state (model parameters) of the kth chain at temperature t_k, and C_k is the corresponding normalizing constant. Note that the temperature ladder is inversely proportional to the temperature defined in the original population MCMC algorithm [55]. The target pdf of the chain with $t_N = 1$ is actually the posterior pdf and it has the lowest temperature. The chain with the highest temperature has $t_1 = 0$ and its target pdf is the prior distribution. In population MCMC, a state in the Markov chain is not a state in a single chain, it is the population M which is consist of N states, i.e., $M = \{m(1), \ldots, m(N)\}$, here, $m(k)$ is a single state from the kth chain. Thus, the target pdf is the joint pdf given by

$$f(M|d_{obs}) = C_t(t) \prod_{k=1}^{N} (L(m(k)|d_{obs}))^{t_k} f(m(k)), \tag{2.42}$$

where $t = (t_1, \ldots, t_N)$, $C_t(t) = \prod_{k=1}^{N} C_k(t_k)$.

We first present the population MCMC algorithm below, and then we give a more detailed discussion of the crossover, mutation and exchange operator, for those not familiar with the genetic algorithm.

Algorithm 2.5: Population MCMC

1. Randomly generate the initial state for each parallel chain, form the initial population $M_0 = \{m_0(1), \ldots, m_0(N)\}$ and assign a temperature for each of the parallel chain from the temperature ladder $t = \{t_1, \ldots, t_N\}$.

2. One iteration of the Markov chain includes the following steps.

a. Similar to the genetic algorithm, apply a mutation or crossover operator to each parallel chain to get a new population with mutation rate q_m and crossover rate $1 - q_m$. q_m can be chosen to achieve a trade-off between the exploration (mutation) and convergence of algorithm (crossover). For a small population size, q_m is usually set to a large value to explore the sample space more.

b. Apply an exchange operator. Exchange $m(k)$ ($k \in \{1, 2, \cdots, N\}$) and its adjacent state $m(j)$ in the current population for N pairs (k, j), where k is sampled uniformly in $\{1, \ldots, N\}$ and $j = k \pm 1$. The probability of exchange $m(j)$ and $m(k)$ is $p_e(m(j), m(k))$, where $p_e(m(k+1), m(k)) = p_e(m(k-1), m(k)) = 0.5$ and $p_e(m(2), m(1)) = p_e(m(N-1), m(N)) = 1$.

3. Repeat step 2 until chains converge.

On the IC fault model, we follow the same implementation as [70], and the mutation rate is $q_m = 1$, which means we want to have more opportunities to explore the sample space. In the step b, $p_e(m(j), m(k))$ is the probability of exchanging states $m(j)$ and $m(k)$. Because $j = k \pm 1$, if $k = 1$, the only choice for j is $j = 0$. Similarly, if $k = N$, the only choice for j is $j = N - 1$. Thus, $p_e(m(2), m(1)) = p_e(m(N-1), m(N)) = 1$. For other values of k, as $j = k \pm 1$, the probability of $j = k+1$ is 0.5 and the probability of $j = k-1$ is also 0.5. It means $p_e(m(k+1), m(k)) = p_e(m(k-1), m(k)) = 0.5$. For $k = 1$, then we automatically set $j = 2$; for $1 < k < N$, we generate a random number between $[0, 1]$, if the random number is less than 0.5, we set $j = k - 1$. Otherwise, we set $j = k + 1$; for $k = N$, we set $j = N - 1$ automatically. The operators used in our implementation are defined below.

Mutation

For the state $m(k)$ randomly selected from the current population $M = \{m(1), \ldots, m(k), \ldots, m(N)\}$, we generate a new state $m(k)'$ as the next sate in the kth chain, then a new population is formed as $M' = \{m(1), \ldots, m(k)', \ldots, m(N)\}$. The next state $m(k)'$ is chosen based on the sampling method used. The acceptance probability of the new population M' is,

$$\alpha_i = \min\left(1, \frac{f(M'|d_{obs})T(M|M')}{f(M|d_{obs})T(M'|M)}\right), \tag{2.43}$$

Substitute Eq. 2.42 to Eq. 2.43,

$$\alpha_i = \min\left(1, \frac{L(m(k)'|d_{obs})^{t_k} f(m(k)')}{L(m(k)|d_{obs})^{t_k} f(m(k))} \frac{T(M|M')}{T(M'|M)}\right), \tag{2.44}$$

where $T(M|M')$ defines the transition probability from population M to population M'. When the transition is symmetric, then the transition probability in Eq. 2.44 is canceled.

Exchange

This operation is the same as defined in parallel tempering [38]. Suppose M is the current population, and t is the temperature ladder, and let

$$(M, t) = [(m(1), t_1), \ldots, (m(k), t_k), (m(j), t_j), \ldots, (m(N), t_N)], \tag{2.45}$$

where t_k is the temperature associated with state m_k. Suppose we wish to exchange two adjacent states $m(k)$ and $m(j)$ (assume $j = k - 1$) without changing the temperature, i.e., change from Eq. 2.45 to

$$(M', t) = [(m(1), t_1), \ldots, (m(j), t_k), (m(k), t_j), \ldots, (m(N), t_N)]. \tag{2.46}$$

The acceptance probability of the new population (M', t) is $\min(1, r_e)$, with

$$r_e = \frac{f(M'|d_{obs})T(M|M')}{f(M|d_{obs})T(M'|M)} = \frac{L(m(j)|d_{obs})^{t_k} L(m(k)|d_{obs})^{t_j}}{L(m(k)|d_{obs})^{t_k} L(m(j)|d_{obs})^{t_j}} \frac{T(M|M')}{T(M'|M)} \tag{2.47}$$

where $T(M|M')$ defines the transition probability from population M to population M'. It is shown by [55] that, $T(M|M') = T(M'|M)$, so the transition term in the acceptance probability is canceled, and

$$r_e = \frac{L(m(j)|d_{obs})^{t_k} L(m(k)|d_{obs})^{t_j}}{L(m(k)|d_{obs})^{t_k} L(m(j)|d_{obs})^{t_j}}. \tag{2.48}$$

2.4.6 Preconditioned Crank-Nicolson MCMC

All the MCMC methods introduced in the previous sections assume the sampling space is a finite dimensional space, i.e., the vector of model parameters m is finite dimensional. If m is a continuous function of position, all the MCMC methods we mentioned previously are not applicable. Moreover, the efficiency of the standard MCMC methods are highly dependent on the dimension of the problem, so that the normal MCMC algorithms like random walk and adaptive MCMC can be computationally prohibitive for large-scale problems. Derived from the Crank-Nicolson discretization of stochastic partial differential equations, Cotter et al. [16] proposed the preconditioned Crank-Nicolson MCMC (pCN MCMC) method, in which the convergence property is invariant with respect to the dimension of the model parameters when applied to discrete problems. The proposal distribution pCN MCMC algorithm is very similar to the proposal distribution in the random walk algorithm. For random walk proposal, the new state $\widetilde{m}_{i+1}$ is generated by

$$\widetilde{m}_{i+1} = m_i + \sigma\xi, \tag{2.49}$$

where $\xi \sim \mathcal{N}(0, C_M)$ and σ is the scaling factor. Whereas, in the pCN-MCMC method, the new state $\widetilde{m}_{i+1}$ is generated by

$$\widetilde{m}_{i+1} = \sqrt{1-\sigma^2} m_i + \sigma\xi, \tag{2.50}$$

here, ξ is generated in the same way as random walk. The proposal distribution is derived from the discretization of stochastic partial differential equations which are invariant for

eigher the reference or the target measure [16]. Although the pCN-MCMC method is independent of the dimension of the problem, the efficiency of the chain still depends on the value of the scaling factor, which can be chosen the same way as random walk.

2.4.7 Monitoring Convergence of a Markov Chain

Although Markov chains are known to converge to the stationary distribution (target distribution) under reasonable assumptions [53], unless proposed states in the Markov chain are obtained by sampling the target distribution itself, states m_i, $i = 0, 1, 2, \cdots I$ generally do not represent a correct characterization of $\pi(m)$, where, unfortunately, I may be an extremely large integer. These early states represent the burn-in part of the chain and are discarded because they do not provide a correct sampling of $\pi(m)$. It is useful to estimate a value of I that ensures that the states m_i, $i = I + 1, I + 2, \cdots$ represents a correct sampling of the posterior pdf and thus can be used to characterize this pdf. In order to determine the burn-in period, the convergence of the Markov chain needs to be monitored. In this paper, we use the multivariate potential scale reduction factor (MPSRF) [7] to monitor the convergence of Markov chains. For a target pdf which is multimodal and the modes are far apart, a chain may appear to converge when in actuality, it in trapped near a single mode, so it is important to initiate multiple parallel chains from different initial states to check convergence [7].

Starting from different initial states generated randomly from the prior distribution of model parameters, we run s parallel chains, but denote the current length of each chain by i, i.e., the current iteration number for each chain is i. The chain convergence is determined by comparing the within chain variance with the variance of the samples from all the different chains. If the two variances are sufficiently close, the chains have converged and all states obtained subsequent to convergence give a representative characterization of the target pdf.

The between sequence covariance at iteration i is defined as

$$B_i = \frac{1}{s-1} \sum_{j=1}^{s} (\overline{m}_j^i - \overline{m}^i)(\overline{m}_j^i - \overline{m}^i)^T, \tag{2.51}$$

where $\overline{m}_j^i$ is the mean of the samples in the jth Markov chain, and $\overline{m}^i$ is the mean of all the chains, which is defined as

$$\overline{m}^i = \frac{1}{s}\sum_{j=1}^{s}\overline{m}_j^i. \tag{2.52}$$

The within-sequence variance W is defined as

$$W = \frac{1}{s(i-1)}\sum_{j=1}^{s}\sum_{t=1}^{i}(m_j^t - \overline{m}_j^i)(m_j^t - \overline{m}_j^i)^T, \tag{2.53}$$

where m_j^t is the state of index t from the jth chain. The posterior covariance matrix is defined as

$$V = \frac{i-1}{i}W + \frac{s+1}{s}B_i. \tag{2.54}$$

By measuring the "closeness" of the posterior covariance matrix V and the within-sequence variance W, the convergence of multiple Markov chains can be monitored. Under the reasonable assumption that the average covariance matrix of the chains, W, is nonsingular, closeness could be defined as

$$\| R \|_F \leq \epsilon, \tag{2.55}$$

where

$$R = W^{-1}V = \frac{i-1}{i} + \frac{s+1}{s}W^{-1}B_i; \tag{2.56}$$

the subscript F denotes the Frobenious norm and ϵ is the error tolerance, Enforcing Eq. 2.55 would require knowledge of how to properly define ϵ. Thus, we instead monitor convergence using the scalar measure of the distance between the matrices V and W developed by [7]. Here as in [7], the scalar measure is defined by

$$\widehat{R} = \frac{i-1}{i} + \frac{s+1}{s}\lambda_1, \tag{2.57}$$

where λ_1 is the largest eigenvalue of the matrix $W^{-1}B_i$. $\widehat{R}$ is a scalar approximation of the MPSRF convergence factor for multivariate problems. When chains converge, the value of $\widehat{R}$ should approach 1 [7], but for multimodal distributions, the value of $\widehat{R}$ is always greater

than 1 [51]. (Note $\widehat{R} \to 1$ as $i \to \infty$ if $\lambda_1 \to$ goes to zero.) For all the example problems presented, we calculate and plot the value of $\widehat{R}$ from the beginning of multiple chains. When the value of $\widehat{R}$ is close to 1 and tends to be stable for a sufficiently large number of states, we discard the burn-in samples of each chain obtained prior to convergence and then regard the rest of the samples as correct samples from the target pdf.

CHAPTER 3

SAMPLING PERFORMANCE OF SEVERAL MCMC METHODS

In this chapter, we compare the performance of six MCMC methods on the quality of data matches, uncertainty characterization and chain convergence. Because the high computational cost of MCMC algorithms, the example we used for comparison is a small 1D highly nonlinear two-phase reservoir model. The reference posterior distribution we use to compare with other MCMC methods is generated using random walk with 5 Markov chains, each chain has 2.3 billion iterations. For the other MCMC methods, we initialize five Markov chains, the largest size of each chain is 1 million and we use them to check the convergence and generate the posterior distribution. The MCMC methods we considered including the modified adaptive MCMC [31], the original adaptive MCMC proposed by Haario et al. [40], preconditioned Crank-Nicolson MCMC [16], Newton MCMC [66] and two-level MCMC (chapter 4).

3.1 Case Description

This test case is the two-phase 1-D horizontal waterflooding example considered by Emerick and Reynolds [23]. As shown in Fig. 3.1, the number of gridblocks is 31, and each gridblock is 50 ft$\times$ 50 ft$\times$ 50 ft. The porosity is uniform and equal to 0.25; the oil viscosity is 2 cp; the water viscosity is 1 cp. The compressibility of water, oil and rock, respectively, is given by 10^{-6} psi^{-1}, 10^{-5} psi^{-1} and 10^{-6} psi^{-1}, respectively. The water injector in the first gridblock is operated at a constant bottomhole pressure of 4,000 psi. There is a producer in the last gridblock, which is operated at a constant bottomhole pressure of 3,000 psi. In the center of the gridblock, there is a monitor well. The initial reservoir pressure is 3500 psi. The model parameters are log-permeability of each gridblock, the prior mean for $\ln(k)$ is 5.0

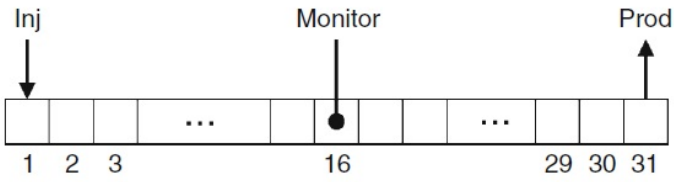
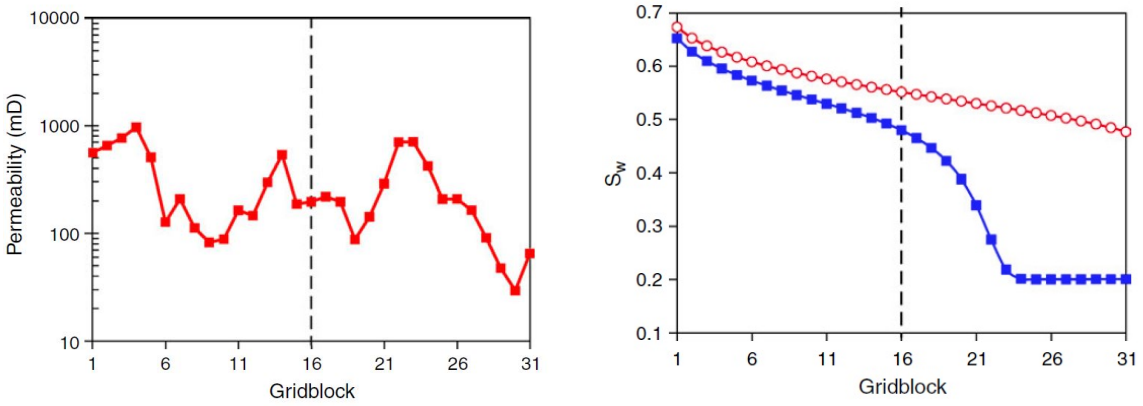


Figure 3.1: Gridblocks and well locations.



(a) Permeability of the true model. **(b) Water saturation distribution.**

Figure 3.2: Example 2: (a) the permeability distribution of the true model; (b) the water saturation distribution at the end of history matching period (blue) and at the end of the forecast period (red). In both plots, the vertical dashed line is the location of the monitor well.

and the prior variance is 1.0. The true permeability field was generated using an exponential covariance function with a practical range equal to 500 ft, which is equal to ten times the width of a gridblock. Fig. 3.2(a) shows the true permeability field. Fig. 3.2(b) presents the water saturation distribution at the end of historical period (blue squares) and at the end of the forecast period (red circles). The historical period is 360 days, the total reservoir life is 750 days. The observation data is the pressure at the monitor well, which is measured every 30 days throughout the 360 days historical period. Random Gaussian measurement error is added to the pressure data generated by the true model with mean equal to zero and standard deviation equal to 1 psi. Note that the results of Fig. 3.2(b) indicate that no water has broken through at the right-end production well at the end of the 360 day historical period.

As the prior pdf is Gaussian and the measurement errors are Gaussian, the posterior

pdf is given by Eq. 2.2, where the objective function $O(m)$ is given by Eq. 2.3.

3.2 Random Walk

The reference posterior distribution for comparison is generated using random walk which was presented in Chapter 2 (Algorithm 2.2). In the random walk algorithm, the value of the scaling factor σ is 0.005. This value was chosen based on several experiments in which we generated chains of length 2000 with different values of σ and chose the value which gave an acceptance rate closest to 0.234 as recommended by Gelman et al. [36], Roberts et al. [87]. Specifically, we ran five experimental chains with 2000 states. The value of σ for each chain is 0.05, 0.01, 0.007, 0.005, 0.001, respectively, and the acceptance rate for each chain respectively is 0.003, 0.08, 0.15, 0.25, 0.74. Thus we choose $\sigma = 0.005$. For random walk, we initialize five parallel Markov chains where the initial states are generated randomly from the prior Gaussian distribution, the length of each chain is 23 million, the convergence rate is plotted in Fig. 3.3. It indicates that when the random walk proposal distribution is used, very long chains much be generated before the value of MPSRF converges to a value close to one. In fact, the results of Fig. 3.3 suggest that the MPSRF value is still slightly decreasing as the chain index increases even after generating chains of length 23 million. Metropolis-Hastings algorithm with a random walk proposal distribution converges very slowly because each new state proposed is obtained from a relatively small neighborhood around the current state in order to obtain acceptable acceptance rates. This proposal distribution results in a chain where the states are highly correlated as a function of the chain index [78, 6, 59], and, in this case, it requires the generation of very long chains in order to generate a reasonable characterization of the posterior pdf. Fig. 3.3 indicates that the chains converge very slowly, but the value of MPSRF has almost stabilized at a value of roughly 2 from states 19 million to 23 million. Thus, we combine the last 500 thousand samples from each chain to form the posterior distribution. Fig. 3.4(a) presents the posterior distribution of the permeability field obtained using random walk, this figure shows a higher uncertainty for the gridblocks to the right of the water front (at the end of the history matching period) which is the

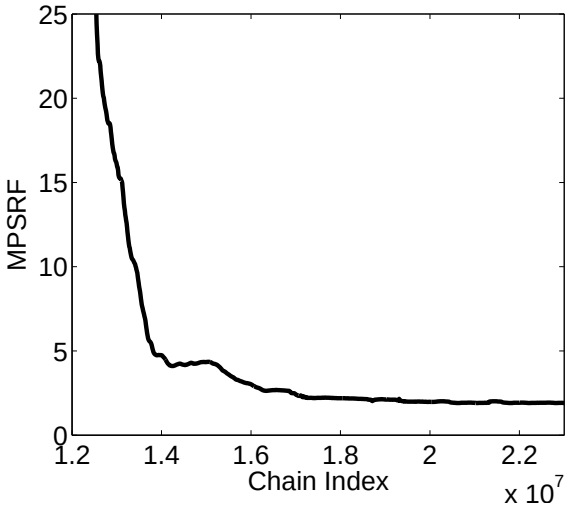
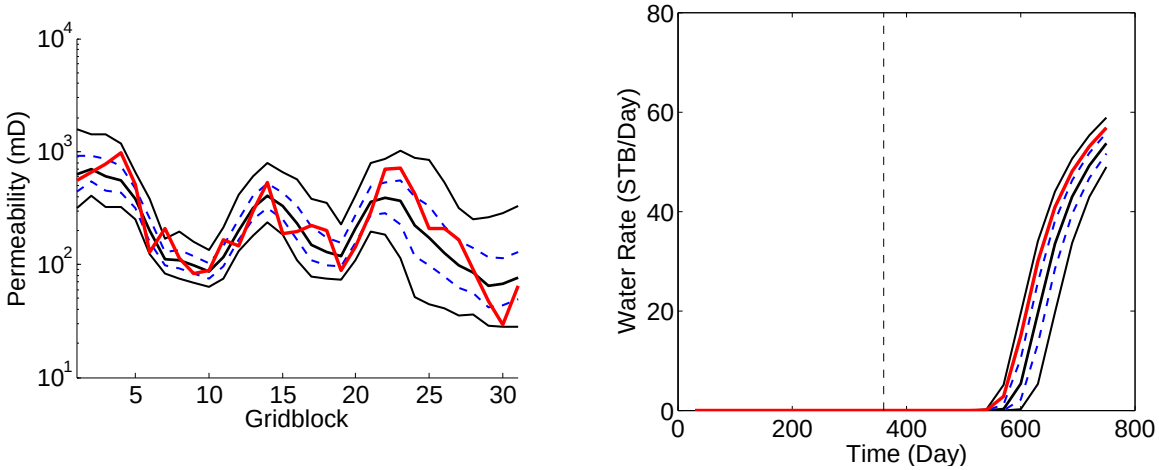


Figure 3.3: Convergence rate calculated using 5 Markov chains of random walk. Only values of MPSRF less than 25 are plotted.

22nd gridblock as shown in fig. 3.2(b). Fig. 3.4(b) presents the posterior distribution of the prediction of the water production rate, it indicates that the true water production rate is located slightly above the P75 percentile. Fig. 3.5 shows the normalized objective function values that correspond to the states of the Markov chains. It indicates that the states in the chains are highly correlated which means the chains are not well mixed.

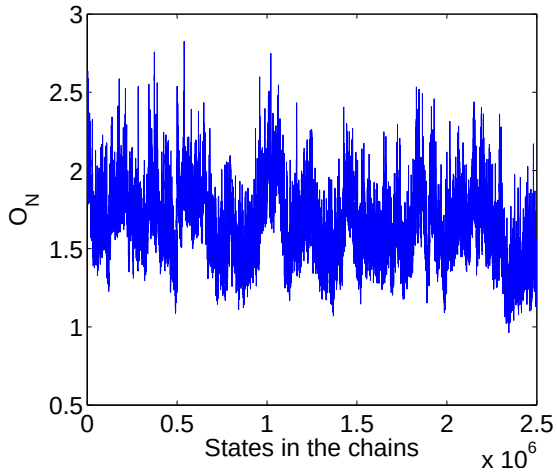
3.3 The Preconditioned Crank-Nicolson (pCN) MCMC

For MCMC algorithms such as random walk and adaptive MCMC, the convergence rate heavily depends on the dimension of the problem. For large scale problems, it is computationally prohibitive to apply these MCMC methods. Proposed by Cotter et al. [16], the Preconditioned Crank-Nicolson (pCN) MCMC is a method such that the number of iterations required for convergence is almost invariant of the dimension of the vector of model parameters when applied to discrete problems. This unique property makes it very useful on large-dimensional problems. The pCN MCMC algorithm we applied here is the one implemented by Iglesias et al. [50], they applied this algorithm on a reservoir model with 3600 gridblocks with a total of 110 Markov chains. The algorithm of the pCN MCMC is very similar to the algorithm of random walk. The difference is that in the algorithm of pCN



(a) Permeability distribution using random walk. (b) Water production rate using random walk.

Figure 3.4: The posterior distribution of permeability (a) and water production rate prediction (b) obtained using iterations from 22.5 to 23 million when applying random walk. In both of the figures, the red curve represents the true case, the solid black curves from bottom to top are P2, median and P98, the blue dashed curves from bottom to top are P25 and P75. In plot (b), the vertical dashed line separates the historical and prediction periods.



(a) Mixing using the random walk.

Figure 3.5: Normalized objective function value of every 1000th states from all the 5 parallel Markov chains using random walk.

MCMC, the proposal distribution at the current state m is given by

$$q(m, \widetilde{m}) = \mathcal{N}(\sqrt{1-\sigma^2}m + (1-\sqrt{1-\sigma^2})m_{\text{pr}}, \sigma^2 C_M), \tag{3.1}$$

where σ is a scalar and $0 < \sigma < 1$, $\widetilde{m}$ is the new proposed state, m_{pr} is the prior mean and C_M is the prior covariance matrix. Similar to random walk, the proposal distribution is symmetric, thus the acceptance probability only depends on the value of the target pdf. In addition, the efficiency of the chain still depends on the value of σ, which can be chosen the same way as random walk, i.e., run several short chains with different values of σ, the one gives the acceptance rate close to 0.23 will be used. In this case, we set $\sigma = 0.005$; the corresponding acceptance rate is 0.25.

For pCN MCMC, we initialize five parallel chains with the same initial states as used in the random walk, the length of each chain is one million. Although the value of MPSRF shown in Fig. 3.6 decreases much faster than is the case for the random walk result (Fig. 3.3). The value of the MPSRF reduces to 5 at the end of one million iterations where we stopped the chains. We use the last 200 thousand states in each of the chains to form the posterior distribution. The posterior distribution of permeability field is shown in Fig. 3.7(a), it indicates that spread of the permeability is very narrow and very different from the results obtained using random walk, the permeability of the true case on gridblocks 23 to 31 are not covered in the spread. This is due to the fact that the chains did not converge. Fig. 3.7(b) presents the posterior distribution of the water production rate, and we note that the uncertainty obtained using the pCN MCMC is lower than the uncertainty obtained using random walk. However, the water production rate generated from the true case is located slightly above the 75 percentile, which is similar to the corresponding random walk result. Fig. 3.8 shows the mixing of the chain, i.e., the normalized objective function values that correspond to the states in the Markov chains. As shown in Fig. 3.8, the range of the normalized objective function using the pCN MCMC is less than the range obtained using random walk (Fig. 3.5), and the states in the chains are highly correlated because

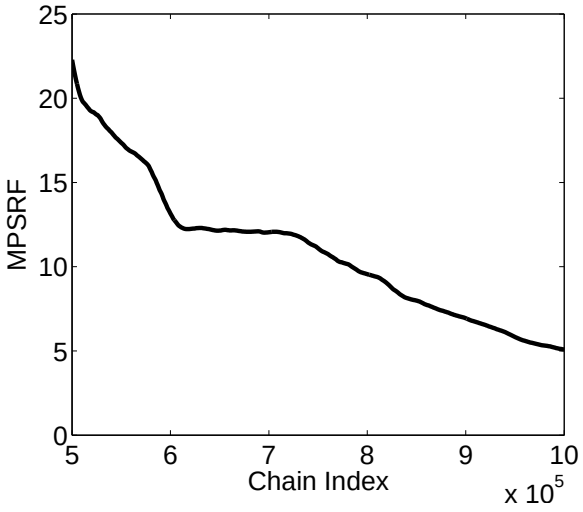


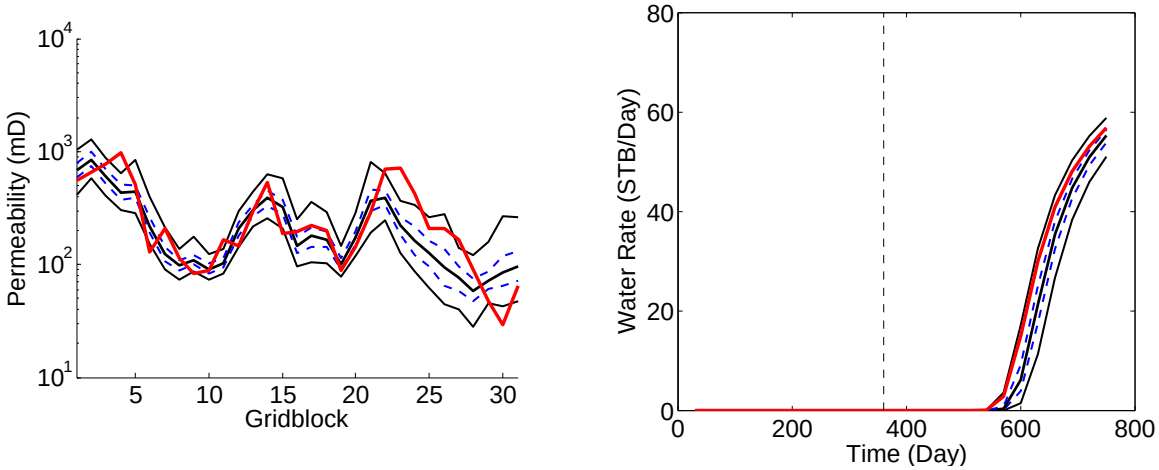
Figure 3.6: Convergence rate calculated using 5 Markov chains of pCN MCMC. Only values of MPSRF less than 25 are plotted.

using $\sigma = 0.005$ in Eq. 3.1 results in highly correlated states.

3.4 The Original Adaptive MCMC

In the original adaptive MCMC algorithm proposed by Haario et al. [40], the proposal pdf is adapted using all the available samples in the chain. The adaptive MCMC algorithm is the same as the Metropolis Hastings algorithm, except the covariance matrix in the proposal pdf and the mean which is used to calculate the covariance matrix keep adapting as the chain evolves. If the covariance matrix in the proposal pdf is adapted appropriately, the adaptive MCMC can converge faster than random walk, as the adaptation process can gather information on the distribution of the target pdf. Because the covariance matrix in the proposal pdf keeps changing, the chain does not necessarily converge to the stationary pdf. Roberts and Rosenthal [86] show that the adaptive MCMC will converge to the stationary distribution under some assumptions. The most important assumption is that the adaptation should diminish as the number of the states in the chain increases. The algorithms we applied on this case is the one proposed by Haario et al. [40], where the complete algorithm is given in Chapter 2 as Algorithm 2.4.

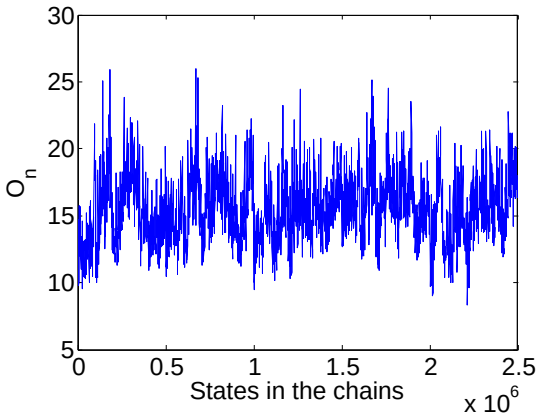
An approximation value of the scaling factor in the proposal distribution can be deter-



(a) Permeability distribution using pCN MCMC.

(b) Water production rate using pCN MCMC.

Figure 3.7: The posterior distribution of permeability (a) and water production rate prediction (b) obtained using iterations from 800 thousand to 1 million when applying pCN MCMC. In both of the figures, the red curve represents the true case, the solid black curves from bottom to top are P2, median and P98, the blue dashed curves from bottom to top are P25 and P75. In plot (b), the vertical dashed line separates the historical and prediction periods.



(a) Mixing using the pCN MCMC.

Figure 3.8: Normalized objective function value of every 1000th states from all the 5 parallel Markov chains using pCN MCMC.

mined using the same procedure as for random walk, i.e., we run several short experimental chains with different values of the scaling factor, and then use the value of the scaling factor that has an acceptance rate around 0.23. In this method, the covariance matrix in the proposal distribution keeps changing, the overall acceptance rate in the chain may change as the chain evolves. For this case, we set the value of the scaling factor equal to 0.07. The acceptance rate at the end of the chain is 0.28. We run 5 chains start from the same initial states as used in random walk to check the convergence where the length of each chain is one million. We plot the convergence rate in Fig. 3.9. The results of Fig. 3.9 suggests that the chains converge much faster using adaptive MCMC than using random walk. Moreover, the value of the MPSRF reduce to 2 at the end of one million iterations as opposed to 5 for pCN MCMC. We use the last 200 thousand states from each of the chains to form the posterior distribution. Fig. 3.10(a) and Fig. 3.10(b) show the posterior distribution of permeability field and water production rate using the adaptive MCMC. The posterior distributions shown in Fig. 3.10(a) and Fig. 3.10(b) are very similar to the result obtained using random walk. For the posterior distribution presented in Fig. 3.10(b), the water production rate generated using the true case is slightly higher than the 75 percentile, which is the same as results obtained using random walk. Fig. 3.11 shows the normalized objective function values that corresponding to the states in the Markov chains generated using the adaptive MCMC. These results indicate less correlation between states in the chain than is evident from the random walk or pCN MCMC.

3.5 Modified Adaptive MCMC [31]

The adaptive MCMC algorithms are a group of algorithms in which the covariance matrix in the proposal pdf is adapted using all the available states in the chain. The modified adaptive MCMC algorithm is a variation of the original adaptive MCMC algorithm and was applied to a two-phase reservoir flow problem by Fossum and Mannseth [30]. Compared to the original adaptive MCMC algorithm, the proposal distribution in the modified adaptive MCMC algorithm is a Gaussian mixture model (GMM) with two different covariance

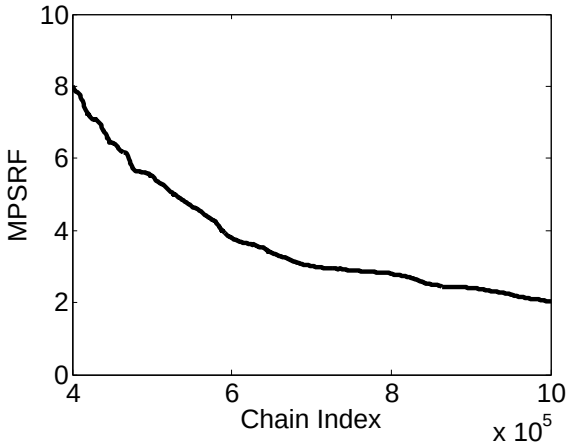
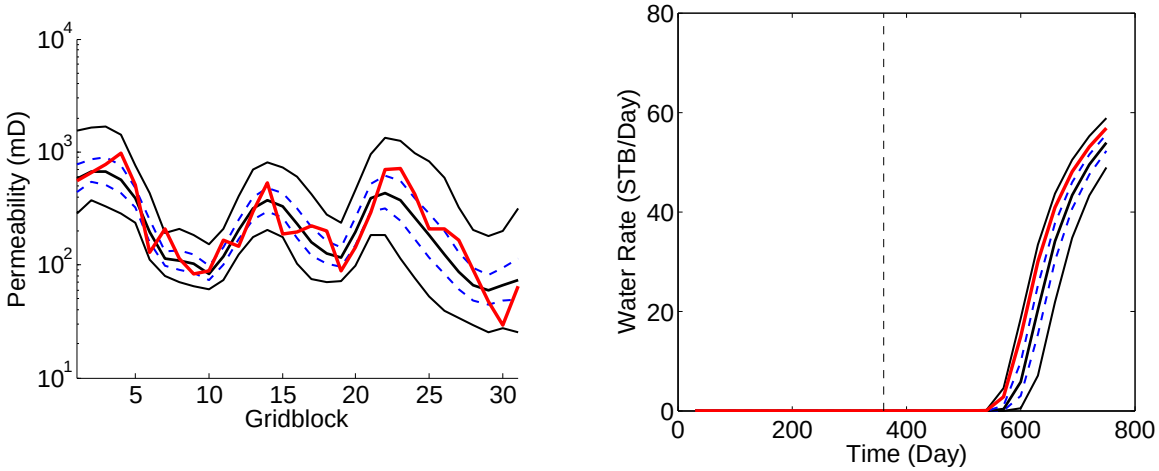


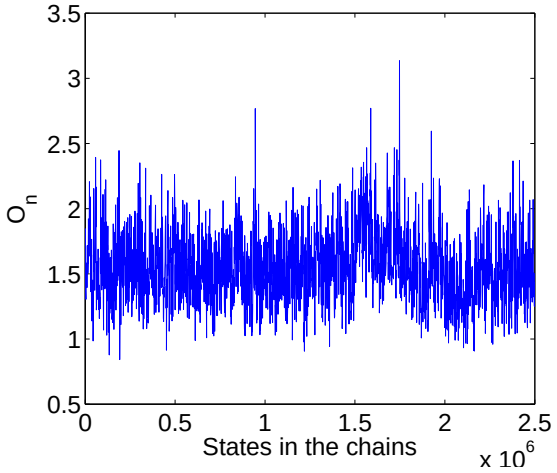
Figure 3.9: Convergence rate calculated using 5 Markov chains of the original adaptive MCMC. Only values of MPSRF less than 8 are plotted.



(a) Permeability distribution using the adaptive MCMC.

(b) Water production rate using the adaptive MCMC.

Figure 3.10: The posterior distribution of permeability (a) and water production rate prediction (b) obtained using iterations from 80 thousand to 1 million when applying the original adaptive MCMC. In both of the figures, the red curve represents the true case, the solid black curves from bottom to top are P2, median and P98, the blue dashed curves from bottom to top are P25 and P75. In plot (b), the vertical dashed line separates the historical and prediction periods.



(a) Mixing using the adaptive MCMC.

Figure 3.11: Normalized objective function value of every 1,000 state from iteration 500 thousand to 1 million in all the 5 parallel Markov chains using the original adaptive MCMC.

matrices (Eq. 3.2), i.e., the prior covariance matrix and the adapted covariance matrix. Although the proposal distribution is a GMM, it is symmetric and the acceptance probability is calculated using the value of the target pdf as shown in Eq. 3.3 below.

Algorithm 3.1: Modified Adaptive MCMC Algorithm

1. Set $i = 0$ and $\beta = 1$, choose the initial state m_0 and set $C_{M0} = C_{M\,\text{prior}}$.

2. Propose a new state $\widetilde{m}_{i+1}$ by sampling the proposal distribution which is given by GMM.

$$\widetilde{m}_{i+1} \sim (1-\beta)\mathcal{N}\left(m_i, \frac{2.38^2}{N_m} C_{Mi}\right) + \beta\mathcal{N}\left(m_i, \frac{2.38^2}{N_m} C_{M\,prior}\right) \tag{3.2}$$

3. Evaluate the acceptance probability (pdf) of the proposed state $\widetilde{m}_{i+1}$ by the Metropolis-Hastings condition given by

$$\alpha(m_i, \widetilde{m}_{i+1}) = \min\left\{1, \frac{\pi(\widetilde{m}_{i+1})}{\pi(m_i)}\right\}. \tag{3.3}$$

4. Generate a random number u from the uniform distribution $U(0,1)$.

5. If $u \leq \alpha(m_i, \widetilde{m}_{i+1})$, the new proposed state is accepted and we set $m_{i+1} = \widetilde{m}_{i+1}$. Otherwise, repeat the current state in the chain, i.e., set $m_{i+1} = m_i$.

6. Update the mean and covariance matrix using

$$\mu_{i+1} = \mu_i + \frac{1}{i+1}(m_{i+1} - \mu_i), \tag{3.4}$$

$$C_{Mi+1} = \frac{i-1}{i} C_{Mi} + \frac{1}{i}((m_{i+1} - \mu_{i+1})(m_{i+1} - \mu_{i+1})^T). \tag{3.5}$$

7. Update β using

$$\beta = \begin{cases} 1 & \text{if } O(m_{i+1}) \leq 50 \\ 0.1 & \text{if } O(m_{i+1}) > 50 \end{cases}$$

8. Set $i = i + 1$ and return to step 2 until the chain has converged and we have obtained the number of samples desired.

Here, the gain factor is set to $\frac{1}{i}$. To check the convergence, we run five parallel Markov chains using the same initial states as used in random walk, the length of each chain is one million. Fig. 3.12 presents the convergence rate, it indicates that using the modified adaptive MCMC algorithm, the chains converge faster than does random walk, however, the value of the MPSRF is still greater than 10 after 1 million iterations in the chain. It converges slower than the original adaptive MCMC in which the proposal distribution is a Gaussian with only the adapted covariance matrix. We combine the last 200,000 states from each chain to form the posterior distribution shown in Fig. 3.13. Fig. 3.13(a) shows the posterior distribution of the permeability field, it is very similar with results obtained using random walk proposal. Fig. 3.13(b) presents the posterior distribution of water production rate, it is also very similar with the posterior distribution obtained using random walk. For both the modified adaptive MCMC and the random walk, the water production rate from the true model is slightly higher than the 75 percentile. Fig. 3.14 shows the normalized objective function values that correspond to the states in the Markov chains. The range of the normalized objective function using the modified adaptive MCMC is very similar with the range based on random walk, however, the result of Fig. 3.14 shows less correlation than is evident in the Fig. 3.5 random walk result.

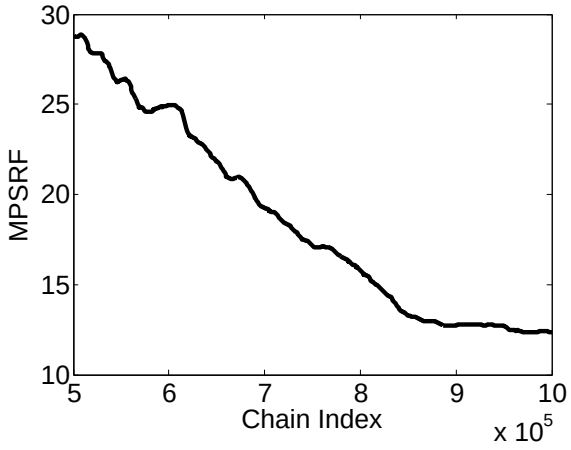
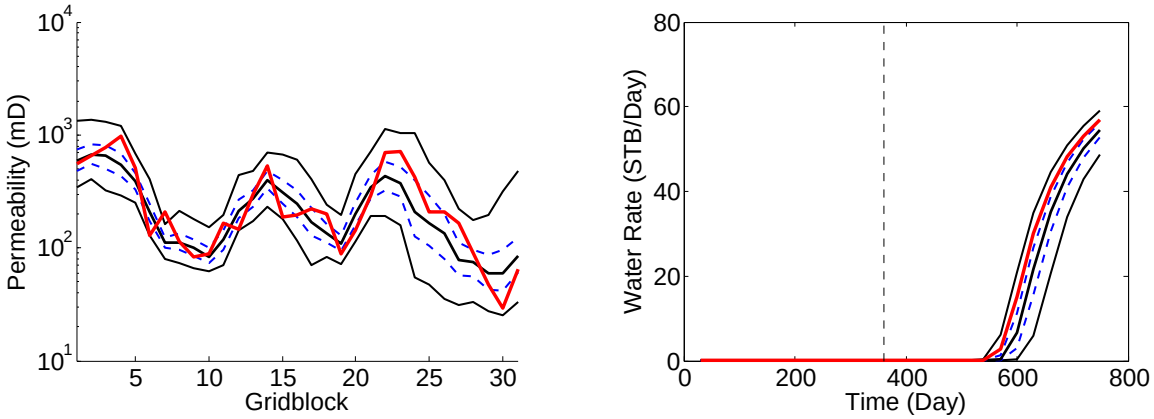


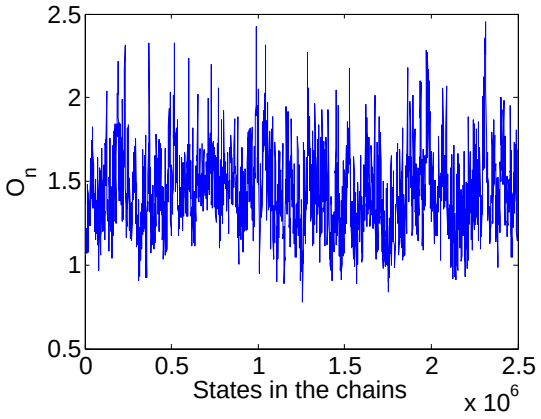
Figure 3.12: Convergence rate calculated using 5 Markov chains of the modified adaptive MCMC method. Only values of MPSRF less than 30 are plotted.



(a) Permeability distribution using the modified adaptive MCMC.

(b) Water production rate using the modified adaptive MCMC.

Figure 3.13: The posterior distribution of permeability (a) and water production rate prediction (b) obtained using iterations from 80 to 100 thousand when applying the modified adaptive MCMC method. In both of the figures, the red curve represents the true case, the solid black curves from bottom to top are P2, median and P98, the blue dashed curves from bottom to top are P25 and P75. In plot (b), the vertical dashed lines separates the historical and prediction periods.



(a) Mixing using the modified adaptive MCMC.

Figure 3.14: Normalized objective function value of every 1,000 state from iteration 500 thousand to 1 million in all the 5 parallel Markov chains using the modified adaptive MCMC method.

3.6 Newton MCMC

A good MCMC algorithm should alow the Markov chain to explore the whole sample space and find the region with high probability rapidly, i.e., the chain should mix fast and converge to the stationary distribution in a computationally feasible number of iterations. The gradient of the target distribution can be used to accelerate the convergence and also improve the mixing of the Markov chain, because using a gradient can drive the Markov chain to high probability regions in fewer iterations. Proposed by Martin et al. [66], Newton MCMC is one of the methods that utilize the gradient information of the target pdf. For Newton MCMC, a new state is proposed from a the Gaussian approximation based on local gradient and Hessian of the objective function evaluated at the current state. The proposal pdf is given by

$$q(m, \widetilde{m}) = \mathcal{N}(m - \sigma_1 H(m)_i^{-1} g(m), \sigma^2 H(m)^{-1}), \tag{3.6}$$

where m is the current state, $\widetilde{m}$ is the new proposed state, $g(m) = \nabla O(m)$ and $H(m) = \nabla^2 O(m)$. The terms σ_1 and σ are two scaling factors which can be turned in the same way as random walk. Note this proposal pdf given by Eq. 3.6 is not symmetric, so the acceptance

probability should be calculated using Eq. 2.38. Details on the Newton MCMC algorithm are given in Chapter 2 Algorithm 2.5.

In this case we set $\sigma_1 = 0.0001$ and $\sigma = 0.25$, those values were chosen based on several experiments in which we generated chains of length 4000 with different values of σ_1 and σ, and then select the values that gave an acceptance rate close to 0.23. Because the covariance matrix in the proposal distribution is changing as the chain evolves, the final acceptance rate of the chain could be different from the acceptance rate of the short chain in the experiment. In this case, the acceptance rate of the whole chain is 0.45, which is between the suggested range [66], i.e., from 30% to 50%. To check the convergence rate, we initiate five parallel Markov chains with the same initial states as random walk, the size of each chain is one million. The convergence rate is plotted in Fig. 3.15. Compared with random walk, Newton MCMC converges much faster. The value of MPSRF reduces to 1.2 at the 300 thousand iteration, and it has almost stabilized at a value of roughly 1.1 after 600 thousand iterations. Thus, we combine states from 600 thousand to 800 thousand iterations in all five Markov chains to form the posterior distribution. Fig. 3.16(a) presents the posterior distribution of the permeability field using Newton MCMC, this figure shows a lower permeability on gridblocks on the right of the 22nd gridblock (location of water front at the end of the history matching period) than results obtained using random walk. The posterior distribution of water production rate from the producer is shown in Fig. 3.16(b) indicates bias; Even though the width of the spread generated using Newton MCMC is similar with the one obtained using random walk, the water production rate generated from the true case is not in the spread which is very different from the results obtained using random walk. Overall, although Newton MCMC converges faster and the value of the MPSRF is very close to unity, the posterior distribution shown in Fig. 3.16 is different from the posterior distribution obtained using random walk. There is no clear reason why this occurs possibly, the reason could be that all the chains converge to a high probability region of the target pdf, and they are all trapped in this region. Fig. 3.17 presents the mixing of the chains, i.e., the normalized objective function values that correspond to the states of the

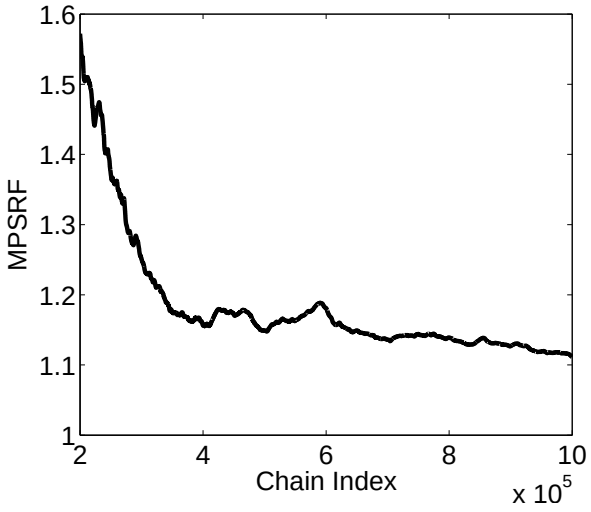
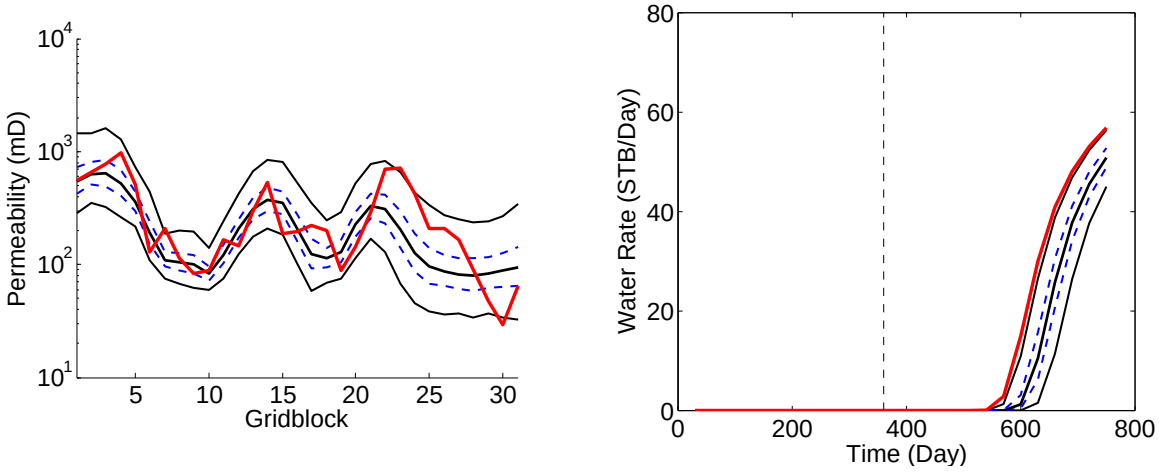


Figure 3.15: Convergence rate calculated using 5 Markov chains of the Newton MCMC. Only values of MPSRF less than 2 are plotted.

chains. As we mentioned previous, the mixing of states generated using Newton MCMC is better than the results obtained using random walk, i.e., the chains constructed with the Newton MCMC exhibits much less correlation than is evident in the random walk.

3.7 Two-level MCMC Method

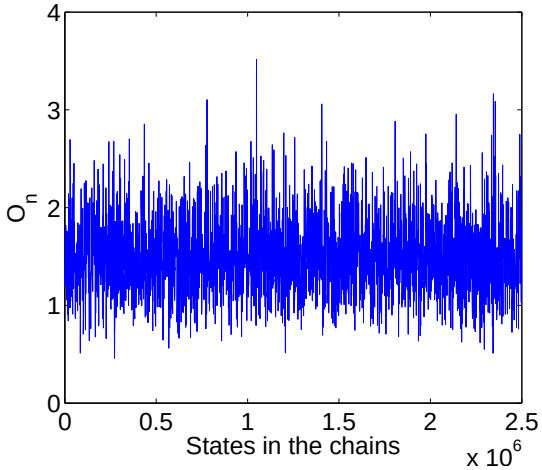
The two-level MCMC method is developed in this work, the detailed description of this method is discussed in the next chapter. Here, we only show the results obtained using the two-level MCMC method on this case and compare the results with the results obtained using other MCMC methods. The idea of the two-level MCMC method is that we design a proposal distribution which is very similar to the target pdf, and use the Metropolis-Hastings algorithm with that proposal distribution to sample the target pdf. In this way, we can sample a multimodal target pdf efficiently. For the two-level MCMC method, we run 5 parallel Markov chains with each chain starting from an sample randomly generated from the GMM constructed at the first level of the algorithm. The convergence rate of the 5 parallel chains is shown in Fig. 3.18, where it indicates that the chains converge after about 15 thousand iterations. States generated using the two-level MCMC method converge much faster than any of other MCMC methods discussed in this chapter. Based on the



(a) Permeability distribution using the Newton MCMC.

(b) Water production rate using the Newton MCMC.

Figure 3.16: The posterior distribution of permeability (a) and water production rate prediction (b) obtained using iterations from 15 to 25 thousand when applying the Newton method. In both of the figures, the red curve represents the true case, the solid black curves from bottom to top are P2, median and P98, the blue dashed curves from bottom to top are P25 and P75. In plot (b), the vertical dashed lines separates the historical and prediction periods.



(a) Mixing using the Newton MCMC.

Figure 3.17: Normalized objective function value of every 1,000 state from iteration 500 thousand to 1 million in all the 5 parallel Markov chains using the Newton MCMC.

convergence rate shown in Fig. 3.18, we use states from 15 thousand to 25 thousand in all chains to generate the posterior distribution. Fig. 3.19(a) presents the posterior distribution of the permeability field, it indicates that the marginal pdf's of gridblock permeabilities obtained using the two-level MCMC method is fairly similar to the results obtained using random walk. Both of them show higher uncertainty in permeability to the right of the water front (at the end of the historical period) than the uncertainty in permeability to the left of the water front. The posterior distribution of the water production rate is presented in Fig. 3.19(b), the uncertainty obtained using the two-level MCMC is slightly higher than the uncertainty obtained using random walk. For both methods, the water production rate of the true model is slightly above the 75 percentile. Fig. 3.20 shows the normalized objective function values that correspond to the states of the Markov chains. Comparing Fig. 3.20 with those of Fig. 3.5, random walk proposal exhibits far more correlation than is shown in the two-level MCMC results. This result arises because in random walk new states are generated from a relatively small neighborhood of the current state which can induce long correlations in the chain, whereas, for the two-level MCMC method, states are generated independently from the proposal distribution which tens to eliminate correlation and promote mixing. Overall, results obtained using two-level MCMC method are similar to results obtained using random walk; however, states generated using two-level MCMC converges much faster and the computational cost using two-level MCMC method is much lower than any other methods tested in this chapter.

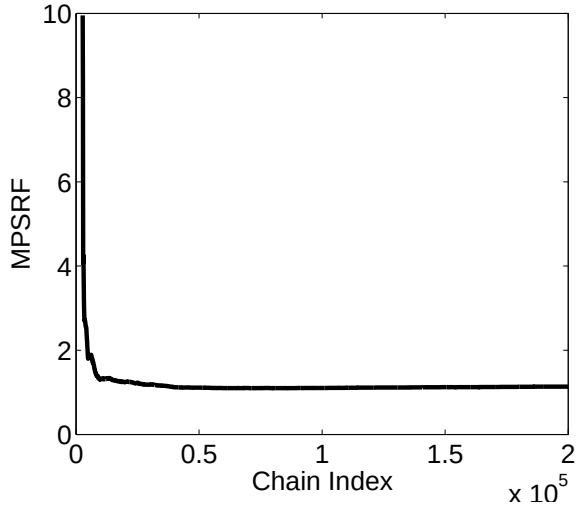
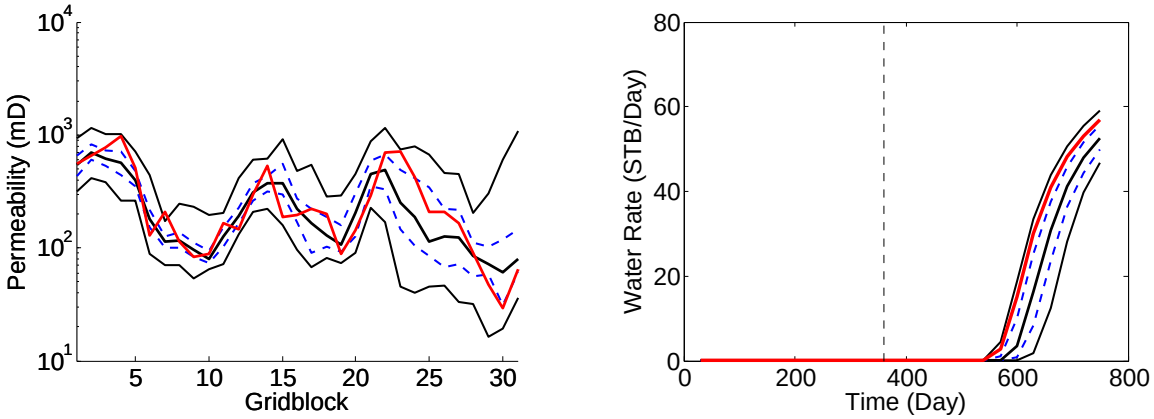


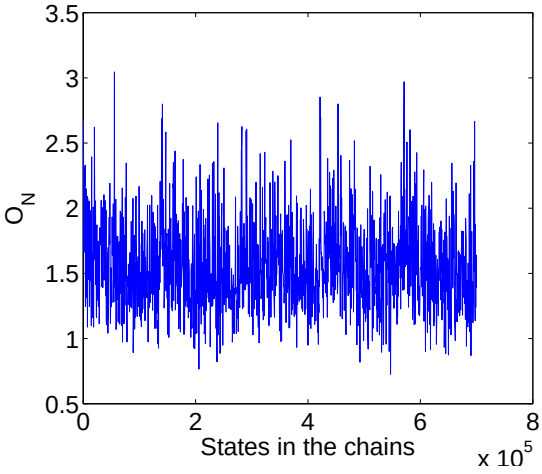
Figure 3.18: Convergence rate calculated using 5 Markov chains of the two-level MCMC method. Only values of MPSRF less than 8 are plotted.



(a) Permeability distribution using the two-level MCMC.

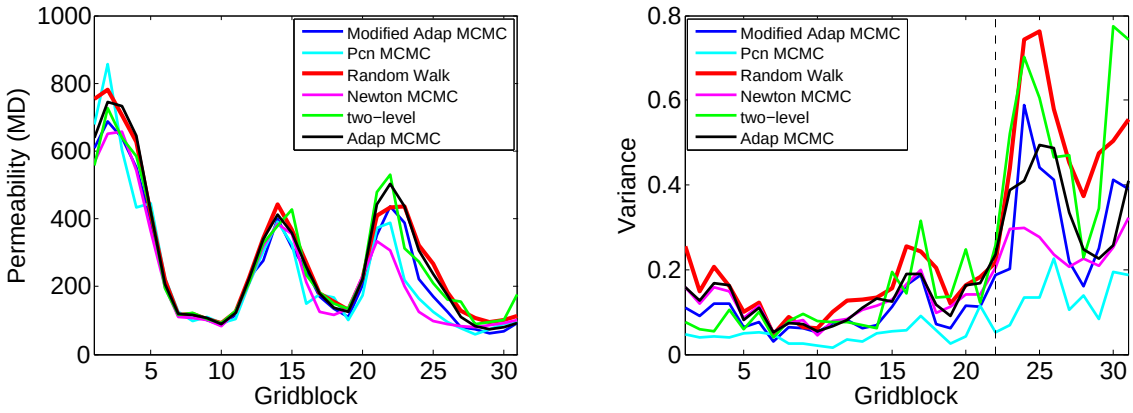
(b) Water production rate using the two-level MCMC.

Figure 3.19: The posterior distribution of permeability (a) and water production rate prediction (b) obtained using iterations from 15 to 25 thousand when applying the two-level MCMC method. In both of the figures, the red curve represents the true case, the solid black curves from bottom to top are P2, median and P98, the blue dashed curves from bottom to top are P25 and P75. In plot (b), the vertical dashed lines separates the historical and prediction periods.



(a) Mixing using the Newton MCMC.

Figure 3.20: (a) Normalized objective function value of states in all the 5 parallel Markov chains using the two-level MCMC method.



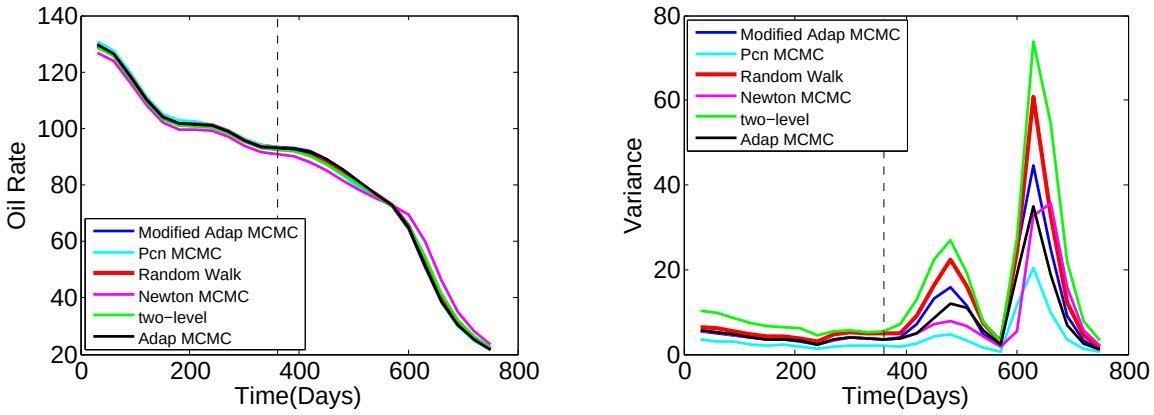
(a) Mean of the permeability field. **(b) Variance of the permeability field.**

Figure 3.21: Comparison of mean (a) and covariance (b) of the permeability field using different MCMC algorithms. The vertical dashed lines denotes the 22nd gridblock, which is the location of water front at the end of the history matching period.

3.8 Overall Comparison

Fig. 3.21 presents the values of the mean and variance (var[ln$k$]) of log-permeability for all the MCMC methods considered in this chapter. We assume the results generated from the five chains are the correct one, although there is no way to prove this is the case. According to the results shown in Fig. 3.21(a), the means are fairly similar except for gridblocks 1, 2, and gridblocks on the right of 22 (position of waterfront at the end of history matching period). Of all results those obtained using pCN MCMC and Newton MCMC are the most different from the results obtained using random walk. Fig. 3.21(b) indicates that, pCN MCMC and Newton MCMC result in the largest under estimation of log-permeability. For the other MCMC methods except pCN MCMC and Newton MCMC, the variances of gridblock log-permeabilities are fairly similar, and the uncertainty for gridblocks on the right of the 22nd gridblock are much higher than the gridblocks on the left of the 22nd gridblock.

Fig. 3.22 presents the values of the mean and variance of oil production rate for all the MCMC methods considered in this chapter. Fig. 3.22(a) indicates the mean of oil production rate obtained using all the MCMC methods are very similar. According to results shown in Fig. 3.22(b), pCN MCMC underestimated the var[q_0], two-level MCMC and modified



(a) Mean of oil production rate. **(b) Variance of oil production rate.**

Figure 3.22: Comparison of mean (a) and covariance (b) of oil production rate using different MCMC algorithms. The vertical dashed lines separates the historical and prediction periods.

adaptive MCMC result in var$[q_0]$ fairly close to the variance obtained from random walk, although a little overestimation (two-level MCMC) and underestimation (modified adaptive MCMC) are observed for the forecast period.

CHAPTER 4

TWO-LEVEL MCMC METHOD

To obtain a computationally efficiency MCMC method, we develop in this chapter a procedure to generate a proposal distribution that is a reasonable approximation of the target distribution. As in [78], the idea is that if one applies the Metropolis-Hastings algorithm with a proposal distribution that is close to the target distribution, it is expected that the chain will converge quickly so that we can obtain a correct representation of the target distribution with relatively short chains and thus significantly enhance computational efficiency. Computational efficiency is of paramount importance for problems of interest to us where m is a vector of reservoir model parameters, the target pdf, $\pi(m)$ is the posterior pdf for m conditional to observed data and the evaluation of the acceptance probability requires a run of a reservoir simulator. As discussed in detail below, our proposal distribution is a Gaussian mixture model which is constructed using modes of the target pdf where these modes are estimated by history matching the observed data starting with a set of initial guesses.

4.1 Posterior pdf and objective function

For problems of interest in assessing the uncertainty in reservoir description and performance, it is common to approximate the prior pdf for the N_m-dimensional vector of model parameters, m, as a Gaussian with mean m_{pr} and covariance matrix C_M; it is also customarily assumed that the measurement error is Gaussian with mean zero and covariance matrix C_D. Under the preceding two assumptions the posterior pdf for m given the vector of N_d-dimensional vector of observed data, d_{obs}, is given by

$$\pi(m) = f(m|d_{obs}) = a\exp(-O(m)), \tag{4.1}$$

[79] where a is the normalizing constant and $O(m)$ is the objective function defined by

$$O(m) = \frac{1}{2}(m - m_{pr})^T C_M^{-1}(m - m_{pr}) + \frac{1}{2}(g(m) - d_{obs})^T C_D^{-1}(g(m) - d_{obs}). \tag{4.2}$$

Here $g(m)$ is the data predicted with the forward model (reservoir simulator in our applications) using the model m. If m is the vector of model parameters and $g(m)$ involves no model error, e.g., represents exact physics, the difference between $g(m)$ and d_{obs} is equal to the measurement error.

The third example that we present in this work pertains to the IC-fault model which was considered previously by [70]. The authors used the posterior pdf of Eq. 4.1 as the target pdf with $O(m)$ defined by

$$O(m) = \frac{1}{N_d}(g(m) - d_{obs})^T C_D^{-1}(g(m) - d_{obs}), \tag{4.3}$$

as the target pdf. Thus, in order to compare the results on uncertainty quantification generated with our two-level MCMC algorithm to those of [70], which were obtained using a population MCMC algorithm, the objective function of Eq. 4.3 is used to define the posterior pdf. This corresponds to seeking maximum likelihood estimates when we build our Gaussian mixture model as described below, where we use the plural form to indicate that $O(m)$ may have multiple local minima.

4.2 Two-level MCMC Algorithm

In the first level of our MCMC algorithm, we approximate the target pdf, $\pi(m)$ by a Gaussian mixture model (GMM). At the second level, this GMM is used as the proposal distribution in the Metropolis-Hasting Algorithm, i.e., Algorithm 2.1.

4.2.1 Gaussian mixture model

In the two-level MCMC method, the proposal distribution is a Gaussian mixture model (GMM). GMM is a probability density function which is a weighted summation of

Gaussian probability densities, and it can be represented by

$$p(m) = \sum_{\ell=1}^{k} w_\ell \mathcal{N}(m_\ell^*, C_\ell) = \sum_{\ell=1}^{k} w_\ell G_\ell(m|m_\ell^*, C_\ell), \tag{4.4}$$

where $p(m)$ is constructed so that it gives a reasonable approximation of the true pdf for m. In Eq. 4.4, $G_\ell(m|m_\ell^*, C_\ell)$ is a Gaussian density function of the form

$$G_\ell(m|m_\ell^*, C_\ell) = \frac{1}{\sqrt{(2\pi)^{N_m}|C_\ell|}} \exp(-\frac{1}{2}(m - m_\ell^*)^T C_\ell^{-1}(m - m_\ell^*)), \tag{4.5}$$

with mean vector m_ℓ^* and covariance matrix C_ℓ. Here $|C_\ell|$ denotes the determinant of the covariance matrix C_ℓ. It is important to note that the weights w_ℓ's must satisfy $\sum_{\ell=1}^{k} w_\ell = 1$ is order for $p(m)$ to be a pdf.

To generate a Gaussian mixture model (GMM) which approximates the target pdf of Eq. 4.1 we apply the algorithm below.

Algorithm 4.1: Approximating the Target Distribution with a Gaussian Mixture Model.

1. For $\ell = 1, 2, \cdots n_1$, where n_1 is the number of initial guesses, generate an initial guess $\tilde{m}_\ell^0$ and minimize $O(m)$ to find a mode, m_ℓ^* of the pdf $\pi(m)$ given in Eq. 4.1. If the prior pdf is the Gaussian $\mathcal{N}(m_{pr}, C_M)$, the $\tilde{m}_\ell^0$s can be generated as independent samples of this prior Gaussian. Given an initial guess, the minimum of $O(m)$ is obtained by the BFGS quasi-Newton method [100, 32]. In our examples, we use an in-house trust-region BFGS code [13] which approximates the Hessian itself, not the inverse Hessian. At the end of each optimization process, we record the quasi-Newton estimate of the Hessian, $B_k(m_\ell^*)$ and the inverse Hessian $B_k^{-1}(m_\ell^*)$ is used as the covariance matrix for the Gaussian distribution with mean m_ℓ^* in the Gaussian mixture model.

2. As for large scale problems, it is possible to generate a large number of minima, where many of them are close to each other [79], we may wish to reduce the number of Gaussians in the Gaussian mixture model to k where $k << n_1$. A fundamental result

of [93], [32] and [79] suggests that at a minimizer, the normalized objective function defined by

$$O_N(m) = \frac{2O(m)}{N_d} \tag{4.6}$$

with $O(m)$ given by Eq. 4.2 should satisfy the following condition:

$$O_N \leq 1 + 5\sqrt{\frac{2}{N_d}}. \tag{4.7}$$

In our examples, to further reduce the number of Gaussians in the Gaussian mixture model we retain only minimizing models that satisfy $O(m_\ell^*) \leq 1.5$ which in the examples considered (except for the toy problem) is only slightly less than the upper bound of the Eq. 4.7. For simplicity, we denote the set of models retained at the end of this step by m_ℓ^*, $\ell = 1, 2, \cdots n$ and in the next step we further reduce this set by k means clustering.

3. For the user-chosen value of k, apply k-medoids clustering [60] to obtain clusters with the set of centroids given by $\{\tilde{m}_1^*, \cdots, \tilde{m}_k^*\}$ where k is the number of clusters. Then, for $\ell = 1, 2, \cdots k$, we replace each centroid $\tilde{m}_\ell^*$ by the minimizing model from step 2 that is closest to $\tilde{m}_\ell^*$ and in the ℓth cluster. We now simply denote the remaining minimizing models by m_ℓ^*, $\ell = 1, 2, \cdots k$. Then the GMM is given by

$$p(m) = \sum_{\ell=1}^{k} w_\ell \mathcal{N}(m_\ell^*, C_\ell), \tag{4.8}$$

where m_ℓ^* is the ℓth mode and $C_\ell = B_\ell^{-1}(m_\ell^*)$ where w_ℓ is set equal to the fraction of the minimizing models in the ℓth cluster divided by the total number of minimizing models retained at the end of step 2.

To sample the posterior pdf, $\pi(m)$, we use the Metropolis-Hastings algorithm, Algorithm 2.1 with the GMM constructed in Algorithm 4.1 as the proposal distribution, i.e., the

probability of proposing a transition from state m_i to the state $\widetilde{m}_{i+1}$ is given by

$$q(m_i, \widetilde{m}_{i+1}) = p(\widetilde{m}_{i+1}) = \sum_{\ell=1}^{k} w_\ell G_\ell(\widetilde{m}_{i+1}|m_\ell^*, C_\ell). \tag{4.9}$$

It is important to note that the proposal distribution of Eq. 4.9 is independent of m_i, which promotes mixing in the chain. Because the covariance matrix obtained from the quasi-Newton trust region optimization method is not the exact covariance matrix, we use proposed samples to update each Gaussian in the Gaussian mixture model. The updating procedure used is similar to the rank-μ-update in CMA (Covariance Matrix Adaptation) Evolution Strategy (ES) proposed by [41, 42].

Algorithm 4.2 - Two Level MCMC

1. Apply Algorithm 4.1 to obtain a Gaussian mixture model. This step represents Level 1 in its entirety and steps 2-6 represent Level 2.

2. Set $i = 0$ and choose $m_i = m_0$, the initial state in the Markov chain, which is generated randomly from the proposal pdf.

3. Propose a new state, $\widetilde{m}_{i+1}$, with probability $q(m_i, \widetilde{m}_{i+1})$, i.e., $\widetilde{m}_{i+1}$ is obtained by sampling the proposal pdf of Eq. 4.9.

4. Compute the probability of accepting the transition from m_i to $\widetilde{m}_{i+1}$, i.e., the probability of accepting $\widetilde{m}_{i+1}$ as the next state in the Markov chain as

$$\alpha(m_i, \widetilde{m}_{i+1}) = \min\Big\{1, \frac{\pi(\widetilde{m}_{i+1}))q(\widetilde{m}_{i+1}, m_i)}{\pi(m_i)q(m_i, \widetilde{m}_{i+1})}\Big\}. \tag{4.10}$$

5. Sample a random number u from a uniform distribution on $[0, 1]$. If $u \leq \alpha(m_i, \widetilde{m}_{i+1})$, then the proposed state $\widetilde{m}_{i+1}$ is accepted as the next state in the Markov chain, i.e. $m_{i+1} = \widetilde{m}_{i+1}$. If $u > \alpha(m_i, \widetilde{m}_{i+1})$, repeat the old state in the chain, i.e., set $m_{i+1} = m_i$. Find the closest Gaussian mode to m_{i+1}, i.e., find the Gaussian index in the GMM such that

$$j = \arg\min_k ||m_{i+1} - m_k^*||. \tag{4.11}$$

Update the covariance matrix associated with the jth Gaussian in the GMM follow [40] using

$$C_j^{(i+1)} = C_j^{(i)} + \beta_i[(m_{i+1} - m_j^*)(m_{i+1} - m_j^*)^T - C_j^{(i)}]. \tag{4.12}$$

Here, m_j^* is the mean associated with the jth Gaussian in the GMM and $C_j^{(i+1)}$ is the covariance matrix associated with the jth Gaussian in the $i+1$th iteration. The terms β_i are defined as a gain factor sequence. It is required that β_i satisfies the following conditions:

$$\sum_{i=1}^{\infty} \beta_i = \infty \quad \text{and} \quad \sum_{i=1}^{\infty} \beta_i^{1+\delta} < \infty, \tag{4.13}$$

for some $\delta \in (0, 1]$ in order to show that the Markov chain converges to its stationary pdf. [40]. Haario et al. [40] also suggested setting $\beta_i = O(\frac{1}{i})$. In this paper, for low dimensional cases (test problem one and three) we use $\beta_i = \frac{1}{i}$, for a higher dimensional case (test problem two), we use $\beta_i = \frac{C}{i}$, where the value C is chosen such that the overall acceptance rate of the Markov chain is between 15% and 50% [85]. We chose the value of C based on some experiments with short chains in order to attempt to find a value of C that gives an acceptance rate between 15% and 50% . The maximum value of β_i is 0.01.

6. Set $i = i + 1$ and go to step 3, until the value of the convergence rate (MPSRF) close to 1 (Chapter 2, section 2.4.7).

Note to sample the Gaussian mixture model of Eq. 4.9, we first define the following subintervals of the interval $[0, 1]$: $I_1 = [0, w_1]$ and $I_j = (w_{j-1}, w_j]$ for $j = 1, 2, \cdots k$. Then to sample the GMM of Eq. 14, we generate a sample z from the uniform distribution on $[0, 1]$ and find the I_ℓ that contains z. To obtain our sample of the GMM, we then sample the ℓth Gaussian, $\mathcal{N}(m_\ell^*, C_\ell) = G_\ell(m|m_\ell^*, C_\ell)$. A sample of this multivariate Gaussian can be done using the Cholesky decomposition of covariance matrix [2, 79] for reasonably sized problems. For very large problems, by sequential Gaussian simulation [17] or, preferably, with the FFT [83] a sample of $\mathcal{N}(m_\ell^*, C_\ell)$ can be generated.

4.3 Verification and Applications

We tested our two-level MCMC method on five example problems. The first one (Example (1a)) is a simple non-linear problem, which has only one model parameter and one observation data, but the posterior pdf has two modes. In this example, the exact posterior distribution is known, so it is used to demonstrate that the two-level MCMC method can sample the target pdf correctly and relatively efficiently. Our second example is a modification of Example 1a which is then used to illustrate the importance of using covariance matrix adaptation. The third example is a 1-D horizontal waterflooding case, and we compare the results obtained from the two-level MCMC method with the random walk method which can sample the target pdf reasonably accurately given a sufficiently long Markov chain. The fourth example is the IC fault Model. For this problem the posterior pdf is known to be multimodal [11, 10]. We compare our results with the results from the population MCMC method which is known to be able to sample a multimodal target pdf [55]. The last example is a 2D synthetic case. In all of these three examples, we run multiple chains and use the convergence factor MPSRF to monitor the convergence of the chains.

4.3.1 Example 1a

We apply the two-level MCMC method to a simple non-linear problem [99]. The forward model is given by

$$d(m) = g(m,t) = 1 - 4.5(m - 2\pi/3)^2 + (t-1)\sin(m), \tag{4.14}$$

where m is a 1-D model parameter which is a scalar and t is time. A single time step data is used with $t = 1$. The prior model is a normal distribution $\mathcal{N}(2.3, 0.04)$. The true data is generated using $t = 1$ and $m = 1.88$ in Eq. 4.14. We add random noise generated from the normal distribution $\mathcal{N}(0, 0.01)$ to the true data to generate the observation data d_{obs}. In this case, we have a Gaussian prior and Gaussian measurement error. So by Bayes theorem,

the target pdf is the posterior distribution given by

$$\pi(m) = f(m \mid d_{obs}) = a\exp(-O(m)), \tag{4.15}$$

where a is the normalizing constant and the objective function $O(m)$ is given by

$$O(m) = 0.5\left(\left(\frac{m-2.3}{0.2}\right)^2 + \left(\frac{g(m,1)-d_{obs}}{0.1}\right)^2\right) \tag{4.16}$$

We run the two-level MCMC algorithm on this simple case using 30 different initial guesses generated randomly from the prior pdf. After optimization, 25 of the initial guesses converge to 2.3077 with variance $C_k = 0.0025$ and for the other 5 initial guesses, the optimization algorithm converged to 1.9141 with variance $C_k = 0.0035$. Based on the optimization results, the GMM is given by

$$p(x) = \frac{25}{30}\mathcal{N}(2.3077, 0.0025) + \frac{5}{30}\mathcal{N}(1.9141, 0.0035). \tag{4.17}$$

In this problem Eq. 4.17 represents the proposal distribution for Algorithm 4.1. In this example, the gradient can be calculated explicitly and only 180 evaluations of the objective function are required to do all 30 optimization runs.

The blue curve in Fig. 4.1 represents the target pdf distribution, while the black curve in Fig. 4.1 is the GMM represented the proposal distribution of Eq. 4.17. For two-level MCMC, using this GMM distribution as the proposal pdf, we run 5 parallel Markov chains starting from different initial states which are generated randomly by sampling the initial GMM of Eq. 4.15. The length of each chain is 10 thousand, which we will show is much longer than is needed to obtain a reasonable sampling of the posterior pdf. In order to check the convergence of these 5 parallel Markov chains, we calculate the convergence rate from the beginning of each Markov chain. Fig. 4.2(a) shows that the MPSRF stabilizes at a value close to one by the 500th state in the Markov chain, which suggests the chains have converged to the target pdf. This suggests from state 500 onward, we are correctly sampling the target

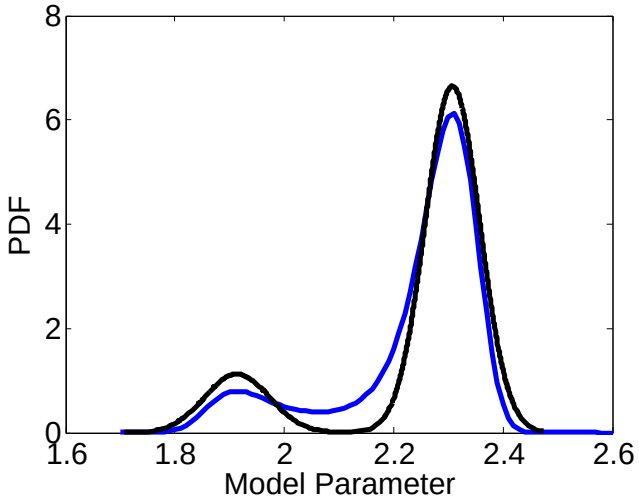
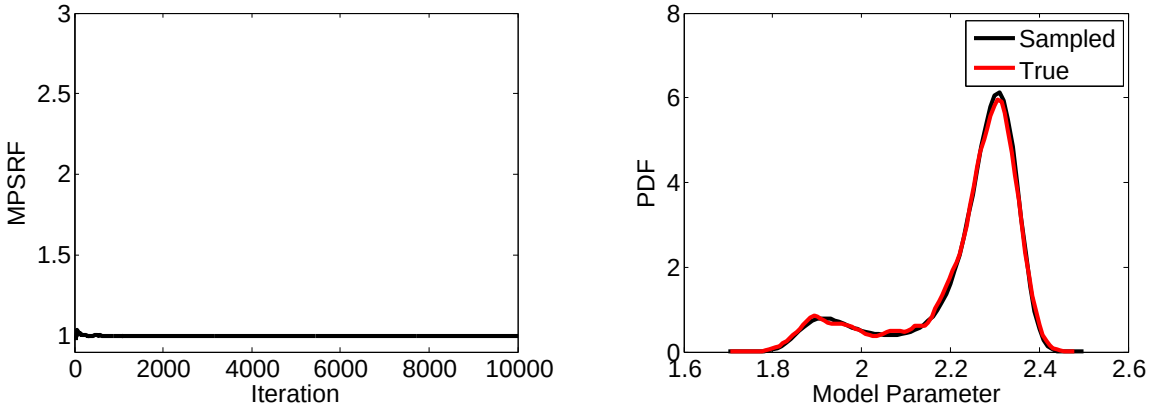


Figure 4.1: Comparison of target pdf in blue and proposal (GMM) pdf in black for example 1a.

pdf. For a one parameter problem, we believe 1,500 samples from each chain is sufficient to construct the pdf, by monitoring the MPSRF, we could stop the chains after constructing 2,000 states in each chain. However, as we wish to validate the utility of MPSRF, in all examples, we generate far more states that are necessary after the value of MPSRF indicates convergence of the chains. Thus, by combining the states corresponding to an index of 500 to 2000 in all chains, we should obtain a good representation of the posterior distribution, and the results of Fig. 4.2(b) indicates that we do. The overall acceptance rate in this case is 80%. In Fig. 4.3, we show that the value of the normalized objective function O_N obtained at accepted states in the 5 parallel chains. In Fig. 4.3, and all similar figures in this paper, we plot all states in the first chain followed by the second chain and so on, where the four red vertical lines demarcate the point where one chain ends and another chain begins. The range of the normalized objective function value changes from close to 0 to 5, and there is no obvious trend in the plot of Fig. 4.3, which indicates that the chain is well mixed.

4.3.2 Example 1b

To show the value of adapting the covariance matrix, we apply the two-level MCMC



(a) Convergence Rate. **(b) Posterior Distribution.**

Figure 4.2: Results of two-level MCMC, Example 1a: (a) MPSRF versus chain index calculated using 5 Markov chains. (b) Comparison of the posterior pdf with the distribution obtained using every 2nd samples from the 500th to the 2000th state from all of the 5 Markov chains.

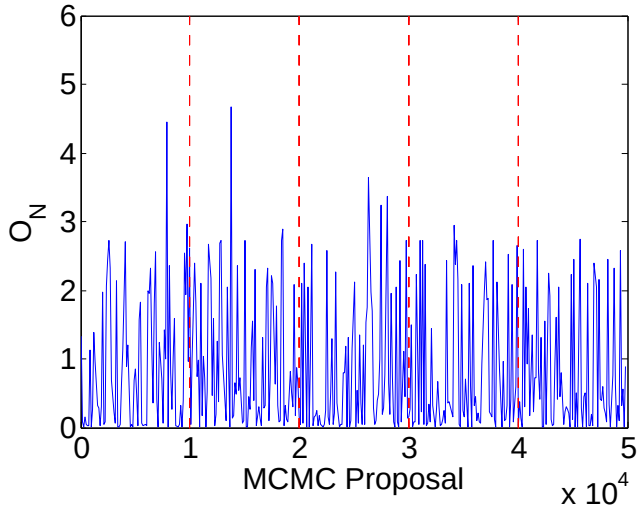
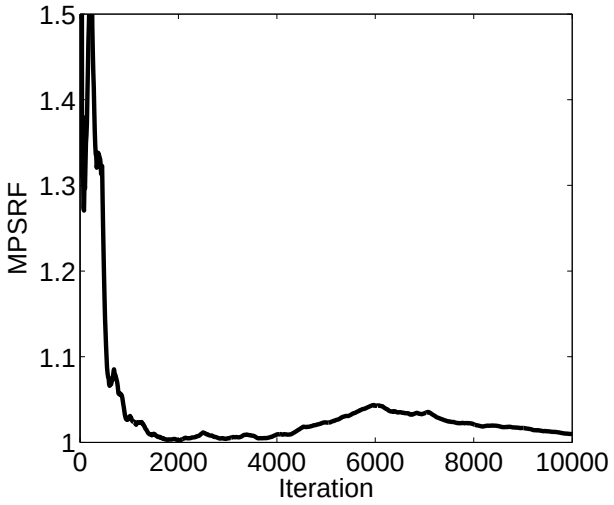


Figure 4.3: Value of normalized objective function of every 125th sample of each of the 5 Markov chains using two-level MCMC, Example 1a.

algorithm on a modification of Example 1a. For this new case, we change the prior model to a different normal distribution $\mathcal{N}(2.1, 0.04)$. The true data is generated using $t = 1$ and $m = 1.66$ in Eq. 4.14. We again add random noise generated from $\mathcal{N}(0, 0.01)$ to the true data to generate the observation data $d_{\text{obs}} = 0.0621$. Using 30 different initial guesses generated randomly from the prior pdf, we run the two-level MCMC algorithm. After the optimization process, 16 of the 30 initial guesses converge to 1.6448 with variance $C_k = 0.0006$ and for the other 14 initial guesses, the optimization algorithm converges to 2.5442 with variance $C_k = 0.0006$. To show the value of adapting the covariance matrix, we significantly reduce both of the variances to $C_k = 0.00005$, so that, our proposal pdf is given by

$$\begin{aligned} p(x) &= \frac{16}{30}\mathcal{N}(1.6448, 0.00005) + \frac{14}{30}\mathcal{N}(2.5442, 0.00005) \\ &= \frac{16}{30}G(m|1.6448, 0.00005) + \frac{14}{30}G(m|2.5442, 0.00005). \end{aligned} \tag{4.18}$$

The sampling results without the adaptation of the covariance matrix are considered first. To check the convergence of the chains, we run 5 Markov chains starting from five different initial guesses generated from Eq. 4.18. The behavior of the MPSRF shown in Fig. 4.4 is erratic and we cannot definitively conclude that we have obtained convergence of the 5 Markov chains. Here, we use states from the 2,000th iteration to the 8,000th iteration to generate the posterior pdf and compare it with the target pdf in Fig. 4.5(b). Note that we obtain a poor approximation (red curve in Fig. 4.5(b)) to the true posterior pdf shown by the black curve in Fig. 4.5(b). This poor approximation result from the GMM (Eq. 4.18) shown in Fig. 4.5(a) is very different from the target pdf. Specifically, the approximate pdf shown in blue does not adequately represent the tails of the two Gaussians of the target pdf. This occurs because the covariances of Gaussians which define the proposal distribution of Eq. 4.18 (Fig. 4.5(a)) are two small so that from a practical viewpoint the chain is not irreducible, i.e., we effectively cannot sample from the tails of the target distribution because we cannot propose states from the tails of the Gaussians making up the target pdf. Fig. 4.5(b) indicates that we did not sample the target pdf correctly when we used Eq. 4.18 as the proposal pdf

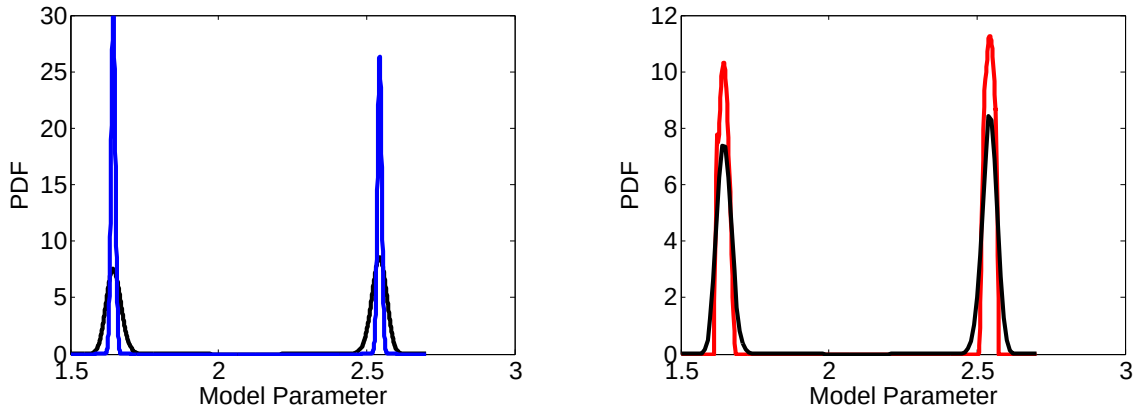


(a) Convergence Rate.

Figure 4.4: Example 1b, results with no covariance matrix adaptation: convergence rate of the Markov chain generated without covariance matrix adaptation.

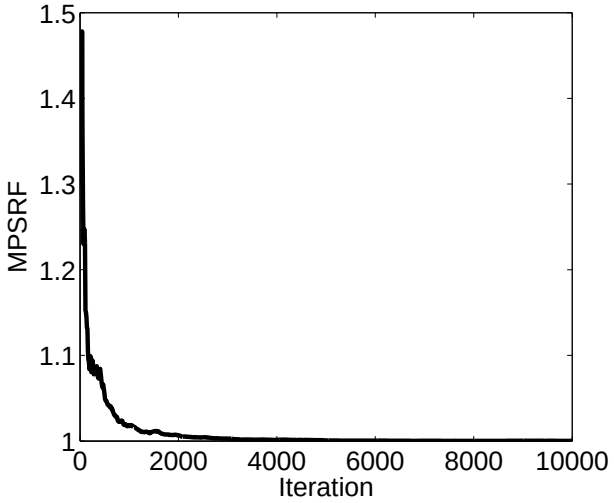
without covariance matrix adaptation. However, we are able to correct the initial GMM of Eq. 4.18 with covariance matrix adaptation so that as the iteration proceed, the proposal pdf becomes more similar to the target pdf. The distribution of the proposed GMM after adaptation is shown Fig. 4.7(a). Note that with covariance matrix adaptation, the final GMM proposal pdf (blue) in Fig. 4.7(a) is close to the target pdf shown in black, and because of this, by using covariance matrix adaptation, we were able to obtain a virtually exact representation of the posterior pdf as shown in Fig. 4.7(b), whereas with no covariance matrix adaptation the MCMC algorithm resulted in a relatively poor representation of the target pdf; see Fig. 4.5(b). In Fig. 4.7(b), we compare the posterior pdf using states from the 1000th iteration to the 3000th iteration with the target pdf. The results of Fig. 4.7(b) indicate that we can obtain a very good representation of the target pdf using two-level MCMC with covariance matrix adaptation. The overall acceptance rate in this case is 0.78.

For comparison, we also use random walk to approximate the posterior pdf discussion. The scaling factor σ in the random walk proposal is chosen such that the acceptance rate of the chain is around 0.234. To estimate an approximate value of σ that roughly achieves this acceptance ratio, we run five experimental chains each with 2000 states. The



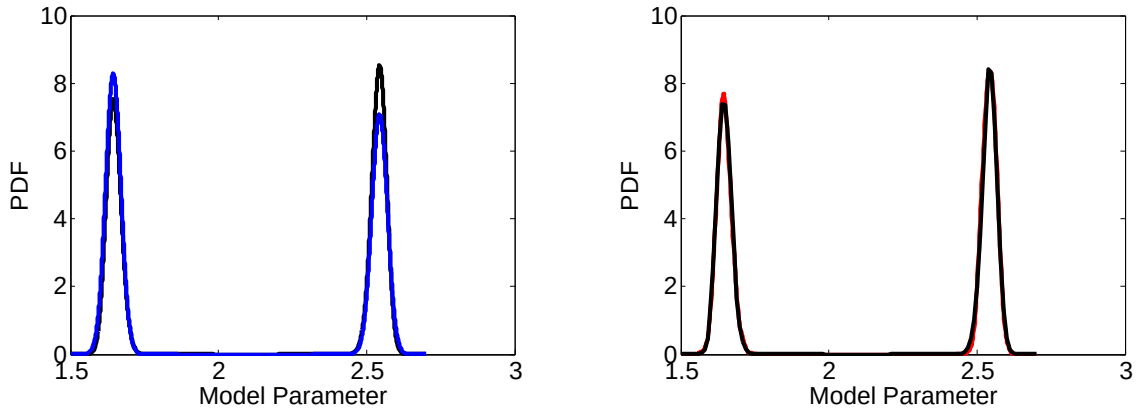
(a) Proposal Pdf (GMM). (b) Posterior Distribution.

Figure 4.5: Example 1b, results with no covariance matrix adaptation: (a) comparison of target pdf in black and the initial proposal (GMM) pdf in blue. (b) Comparison between the target pdf and the distribution of every second samples from the 2000th iteration to 8000th iteration using all five chains. In (b), the black curve represents the target pdf, the red curve represents the sampled pdf.



(a) Convergence Rate.

Figure 4.6: convergence rate (MPSRF) with covariance matrix adaptation.



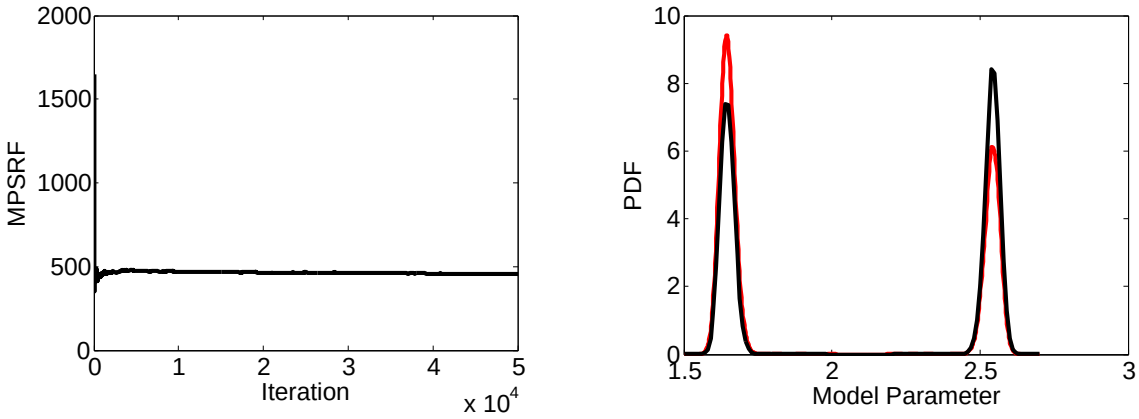
(a) GMM after Adaptation. **(b) Posterior Distribution.**

Figure 4.7: Example 1b, results using covariance matrix adaptation: comparison of target pdf in black and proposal (GMM) pdf at the end of adaptation in blue with covariance matrix adaptation. (b) is the comparison between the target pdf and the distribution of every 2nd samples from the 1000th iteration to 3000th iteration using all five chains. In (b), the black curve represents the target pdf, the red curve represents the sampled pdf.

value of σ for each chain is $0.5, 0.6, 0.7, 1.0$, and the acceptance rate for each chain is $0.285, 0.251, 0.221, 0.157$, respectively. Based on these results, we use $\sigma = 0.7$. In order to monitor convergence, we also initialize five parallel chains where the five initial guesses are generated randomly by sampling the prior distribution. The convergence rate is shown in Fig. 4.8(a). Fig. 4.8(a) indicate that for this case, random walk does not converge. Nevertheless, we use states from the 10,000th iteration to the 30,000th iteration to generate the posterior distribution shown in Fig. 4.8(b). Note that the posterior pdf generated using random walk is different from the target pdf, which indicates that random walk does not sample correctly. More specifically, each individual chain actually generates samples from only one of the Gaussians in the target pdf. This means if we generated only one chain, our approximation to the target pdf would have only a single mode.

4.3.3 Example 2

Reservoir model description: This test case is the two-phase 1-D horizontal waterflooding example considered by Emerick and Reynolds [23]. As shown in Fig. 4.9, the number



(a) Convergence Rate. **(b) Posterior Distribution.**

Figure 4.8: Example 1b: results using random walk proposal: (a) MPSRF versus chain index calculated using 5 Markov chains. (b) Comparison of the posterior pdf with the distribution obtained using samples from the 10,000th to the 30,000th state from all of the 5 Markov chains. In (b), the black curve represents the target pdf, the red curve represents the sampled pdf.

of gridblocks is 31, and each gridblock is 50 ft$\times$ 50 ft$\times$ 50 ft. The porosity is uniform and equal to 0.25; the oil viscosity is 2 cp; the water viscosity is 1 cp. The compressibility of water, oil and rock, respectively, is given by 10^{-6} psi^{-1}, 10^{-5} psi^{-1} and 10^{-6} psi^{-1}, respectively. The water injector in the first gridblock is operated at a constant bottomhole pressure of 4,000 psi. There is a producer in the last gridblock, which is operated at a constant bottomhole pressure of 3,000 psi. In the center of the gridblock, there is a monitor well. The initial reservoir pressure is 3500 psi. The model parameters are log-permeability of each gridblock, the prior mean for $\ln(k)$ is 5.0 and the prior variance is 1.0. The true permeability field was generated using an exponential covariance function with a practical range equal to 500 ft, which is equal to ten times the width of a gridblock. Fig. 4.10(a) shows the true permeability field, which is the same as the one used by Emerick and Reynolds [23]. Fig. 4.10(b) presents the water saturation distribution at the end of historical period (blue squares) and at the end of the forecast period (red circle). The historical period is 360 days, the total reservoir life is 750 days. The observation data is the pressure at the monitor well, which is measured every 30 days throughout the 360 days historical period (same as the observation data used

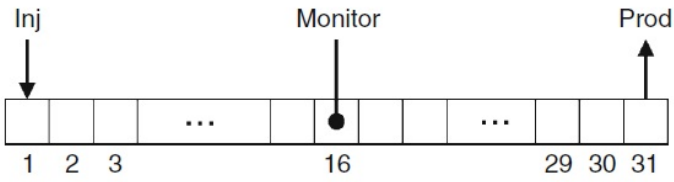
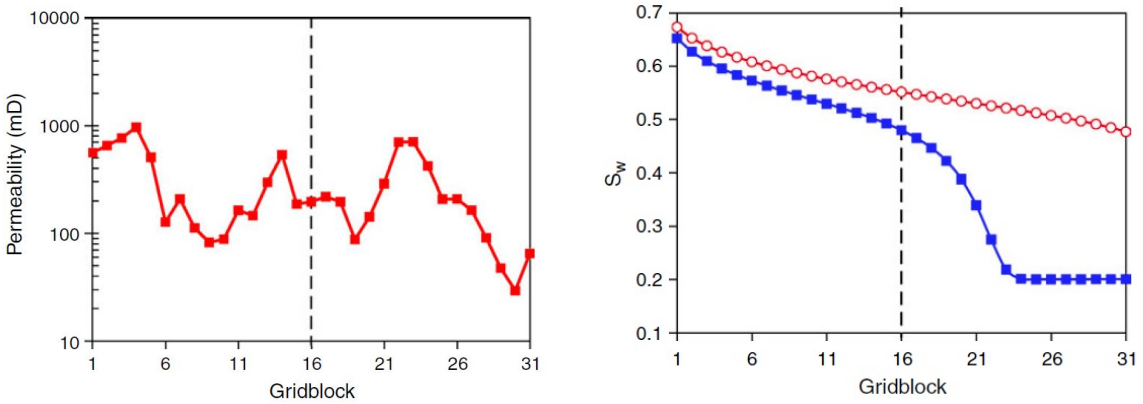


Figure 4.9: Gridblocks and well locations.



(a) Permeability of the true model. **(b) Water saturation distribution.**

Figure 4.10: Example 2: (a) the permeability distribution of the true model; (b) the water saturation distribution at the end of history matching period (blue) and at the end of the forecast period (red). In both plots, the vertical dashed line is the location of the monitor well.

in [23]). Random Gaussian measurement error is added to the pressure data generated by the true model with mean equal to zero and standard deviation equal to 1 psi. Note that the results of Fig. 4.10(b) indicate that no water has broken through at the right-end production well at the end of the 360 day historical period.

As the prior pdf is Gaussian and the measurement errors are Gaussian, the posterior pdf is given by Eq. 4.1, where the objective function $O(m)$ is given by Eq. 4.2.

Results: We run the two-level MCMC algorithm on this 1-D reservoir model using 200 initial guesses which are 200 samples generated from the prior distribution of the model parameters, $\mathcal{N}(m_{prior}, C_M)$. The computational time required to generate one adjoint solution for evaluation of the gradient in the BFGS optimization algorithm is approximately equal to 0.3 the time required for one forward reservoir simulation run. Based on this equivalence, the average number of equivalent reservoir simulation runs required to estimate each

mode of the posterior is 52 reservoir simulation runs. As we estimated 200 modes of the posterior pdf for this example in order to provide an exhaustive experiment, 10,400 reservoir simulation runs were used to estimate the initial Gaussian mixture model. In practice, we would likely have to settle for generating fewer modes to enhance computational efficiency. In this case, we use all minimizing models which result in a normalized objective function value less than 1.5 to generate 25 modes by k-medoids clustering, see Algorithm 4.1. During sampling in Algorithm 4.2, to increase the sampling efficiency, we restrict the covariance matrix to $0.1 \times C_\ell$ in the GMM. The scale factor 0.1 is chosen by running short experiment chains (Metropolis-Hastings algorithm without adaptation) with 200 iterations. We use scale factor 0.1 because for chains with a scale factor larger than 0.1 do not have samples with a normalized objective function value comparable to the objective function value at the mode is accepted as a state in the chain. The gain factor is defined as $\beta_i = \frac{100}{i}$, and the overall acceptance rate is around 0.22. In this case, we use $C = 100$ because it gives an acceptance rate between 15% and 50%. In order to check the convergence of the chain, we generate 5 parallel Markov chains with each chain starting from an independent sample randomly generated from the GMM constructed at the first stage of the algorithm. Fig. 4.11 shows the chain convergence rate, i.e., the value of MPSRF versus the chain index. The results of Fig. 4.11 indicate that the chains converge after about 15 thousand iterations. Thus, we use states from 15 thousand to 25 thousand in all chains to generate the posterior distribution. Fig. 4.12 presents the posterior distribution of the permeability field and predicted water production rate (percentiles P5, P25, P50, P75, P95). Fig. 4.12(a) indicates that the uncertainty in permeability ahead of water front (at the end of historical period) is higher than the uncertainty in permeability behind the water front. In Fig. 4.12(b), the water production rate of the true model is slightly above the P75 curve. The vertical dashed line in Fig. 4.12(b) and similar figures separates the historical period from the prediction period. Note that even though the water rate at the producer was zero during the history matching period and these zero rates were not used as observed data to be matched, the uncertainty bands obtained from the MCMC characterization of the posterior pdf of the permeability

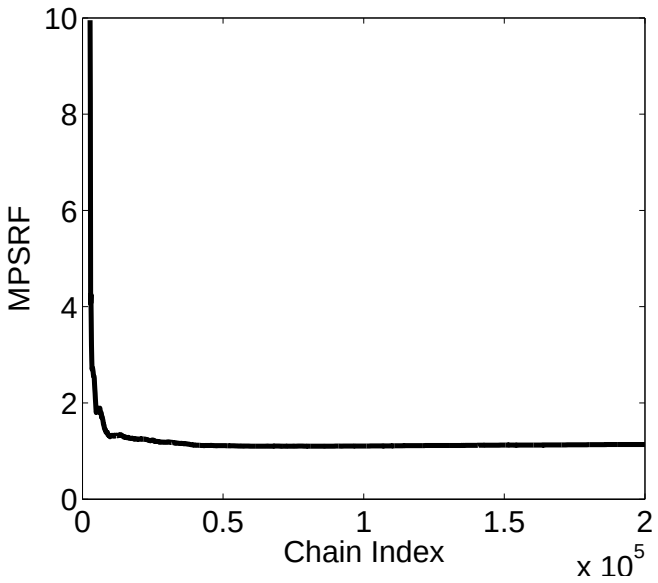
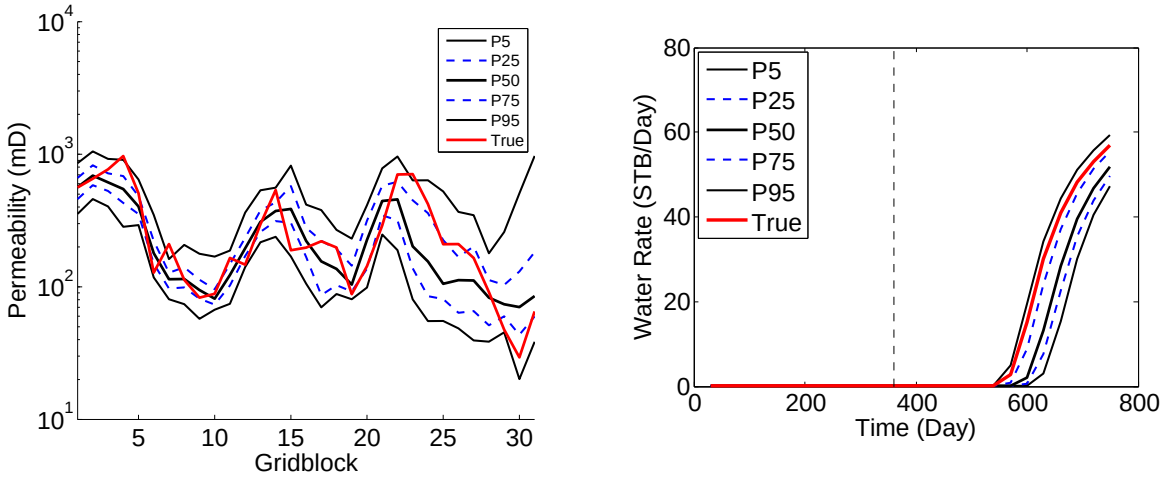


Figure 4.11: Convergence rate calculated using 5 Markov chains for example 2 using Algorithm 4.2. Only values of MPSRF less than 10 are plotted.

field encompass the prediction of the water rate from the true model.

In order to verify that we used enough samples to generate the posterior distribution, Fig. 4.13 compares results obtained using all states with indices from 15 thousand to 25 thousand with results obtained using all states with indices from 60 thousand to 200 thousand. Fig. 4.13 indicate that the posterior distribution of the water production rate obtained from the longer Markov chain and the shorter Markov chain are quite close, and that the posterior distribution of permeability obtained using samples in the longer Markov chain and the shorter Markov chain are very close. This suggests that samples (states) corresponding to indices from 15,000 to 25,000 are sufficient to characterize the posterior pdf.

The results using random walk are used to compare with the two-level MCMC method. For random walk, the value of σ is 0.005. This value was chosen based on several experiments in which we generated chains of length 2000 with different values of σ, $\sigma = 0.05, 0.01, 0.007, 0.005, 0.001$ and chose the value which gave an acceptance rate closest to 0.234. Specifically, we ran five experimental chains with 2000 states. The value of σ for each chain is 0.05, 0.01, 0.007, 0.005, 0.001, respectively, and the acceptance rate for each chain respectively is 0.003, 0.08, 0.15, 0.25, 0.74. For random walk, we also initialize five

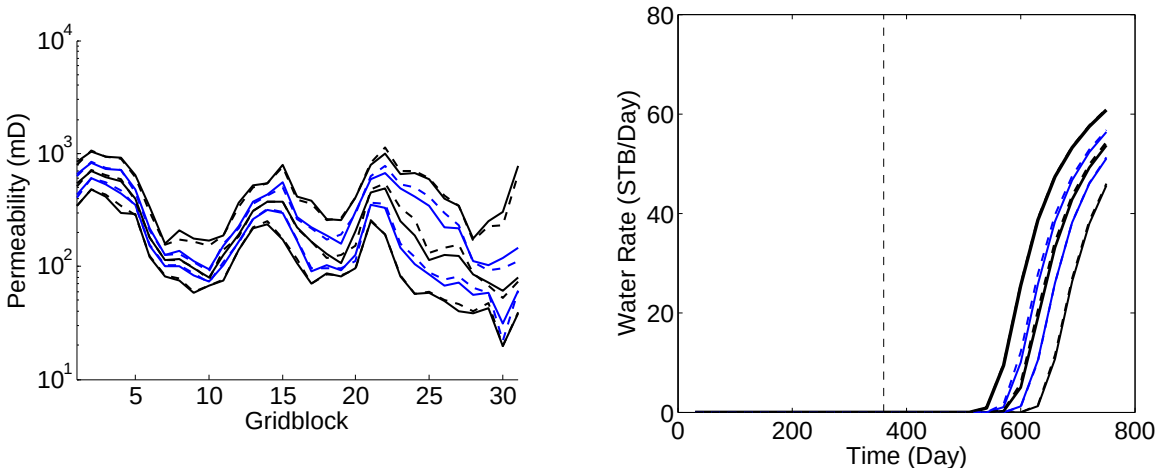


(a) Permeability distribution using the two-level MCMC method.

(b) Water production rate using the two-level MCMC method.

Figure 4.12: The posterior distribution of permeability (a) and water production rate prediction (b) obtained using iterations from 15 to 25 thousand when applying the two-level MCMC method for example 2. In both of the figures, the red curve represents the true case, the solid black curves from bottom to top are P5, median and P95, the blue dashed curves from bottom to top are P25 and P75. In plot (b), the vertical dashed lines separates the historical and prediction periods.

parallel Markov chains but in this case, the initial guesses are generated randomly from the prior Gaussian distribution, the size of each chain is 23 million and we plotted the convergence rate in Fig. 4.14. Compared with the two-level MCMC MPSRF results of Fig. 4.11, Fig. 4.14 shows that, when the random walk proposal distribution is used, very long chains much be generated before the value of MPSRF converges to a value close to one. In fact, the results of Fig. 4.14 suggest that the MPSRF value is still slightly decreasing as the chain index increases even after generating chains of length 23 million whereas the two-level MCMC results of Fig. 4.11 indicate the value of MPSRF has stabilized at a value close to unity after 50 thousand iterations. MCMC with a random walk proposal distribution converges very slowly because each new state proposed is obtained from a relatively small neighborhood around the current state in order to obtain acceptable acceptance rates. This proposal distribution results in a chain where the states are highly correlated as a function of the chain index [78, 6, 59], and, in this case, it requires the generation of very long chains in



(a) Comparison of permeability distribution. **(b) Comparison of water production rate.**

Figure 4.13: (a) the solid curves are distributions obtained using iterations from 60 to 200 thousand, the dashed curves are distributions obtained using iterations from 15 to 25 thousand. In each set of the two distributions, curves in black from bottom to top are the percentiles P5, median and percentiles P95, curves in blue from bottom to top are the percentiles P25 and P75; (b) the solid curves are distributions obtained using iterations from 60,000 to 200,000, the dashed curves are distributions obtained using states from 15,000 to 25,000. In each set of the two distributions. Different colors in the curves have the same meaning as (a). In plot (b), the vertical dashed lines separates the historical and prediction periods.

order to generate a reasonable characterization of the posterior pdf. Fig. 4.14 indicates that the chains converge very slowly, but the value of MPSRF has almost stabilized at a value of roughly 2 from states 19 million to 23 million. Thus, we combine the last 500 thousand samples from each chain to form the posterior distribution. Fig. 4.15(b) and Fig. 4.16(b) present the posterior distribution of the permeability field and the prediction of the water production rate (P5, P25, P50, P75 and P95).

In Fig. 4.15, we compare the posterior distribution of the permeability field obtained using the two-level MCMC method (a) and random walk (b). Note that the marginal pdf's of gridblock permeabilities obtained using the two different proposal pdf's are qualitatively similar, but there are clear difference, for example, the marginal distribution for gridblocks 28, 29 and 30. Fig. 4.16 presents the comparison of the posterior distributions of the predicted water production rate. Note that the uncertainty obtained using the random walk results is lower than the uncertainty obtained using the two-level MCMC method. For both the two-

level MCMC method and the random walk, the water production rate from the true model is slightly higher than P75 curve. Which MCMC implementation gives the best uncertainty quantification is uncertain but based on a comparison of the MPSRF results of Fig. 4.11 and Fig. 4.14, we believe that the two-level MCMC results are superior. It is important to note that with the two-level MCMC scheme, Fig. 4.11 indicates the chains converged after 15,000 MCMC iterations whereas for MCMC with random walk, Fig. 4.14 suggests that we have to generate at least 18 to 19 million states before we start sampling from the target posterior distribution. Fig. 4.17 shows the normalized objective function values that correspond to the states of the Markov chains. As shown the range of the normalized objective function using the two-level MCMC is larger than the range based on random walk results. Also note that even though we plot every 2nd state from the two-level chains as opposed to every 100th state from the random walk chains, the results of Fig. 4.17 indicate that the random walk proposal procedure exhibit far more correlation than is evident in the two-level results. We emphasize that results on correlation between states is as expected. Specifically, when proposing a new state from the GMM of Eq. 4.4, the proposed state is independent of the current state in the chain which tends to minimize correlations between successive states. On the other hand, when using the random walk proposal distribution, (i) a new state is proposed from a Gaussian centered at the current state in the chain and (ii) the variation of this Gaussian is usually fairly small in order to obtain a reasonable acceptance rate; thus, the resulting states in the chain are highly correlated [78, 6], i.e., the random walk proposal distribution tends to yield chains that are not well mixed and thus very long chains are required to obtain a reasonable characterization of the target pdf.

For this example, we also show below results using a different number of clusters, i.e., 10 clusters, 25 clusters and 83 clusters (all the models obtained from optimization that gave an acceptable value of the objective function), the posterior distribution estimated is almost independent of the number of clusters used as shown in Fig. 4.18 (posterior distribution of permeability) and Fig. 4.19 (posterior distribution of water production rate).

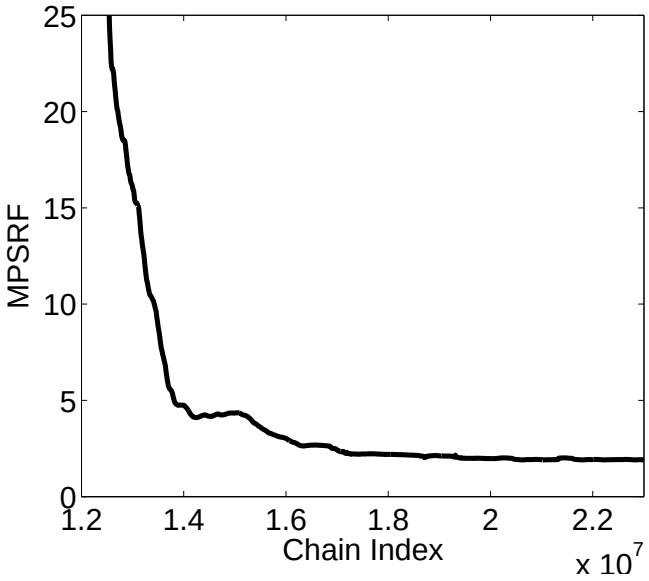
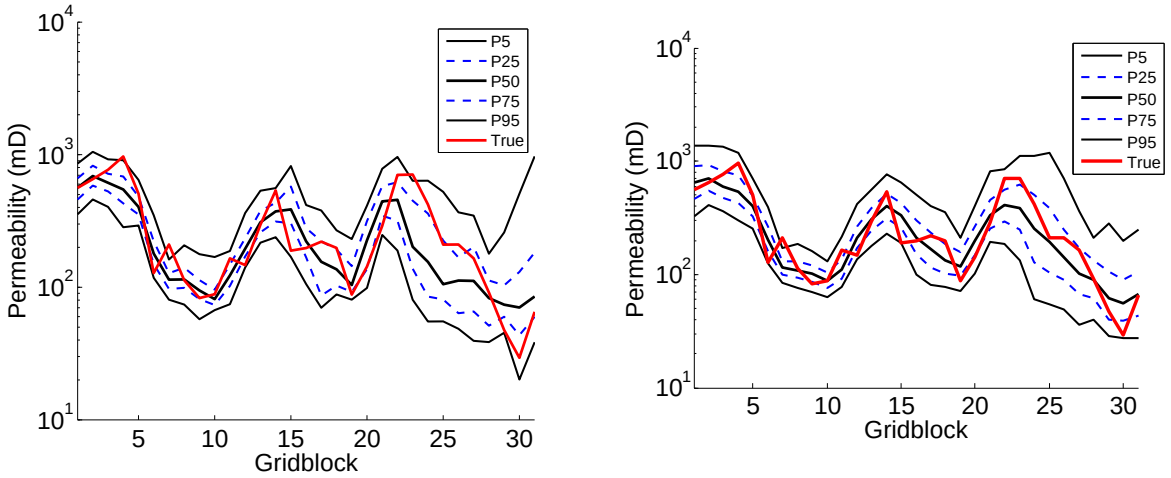


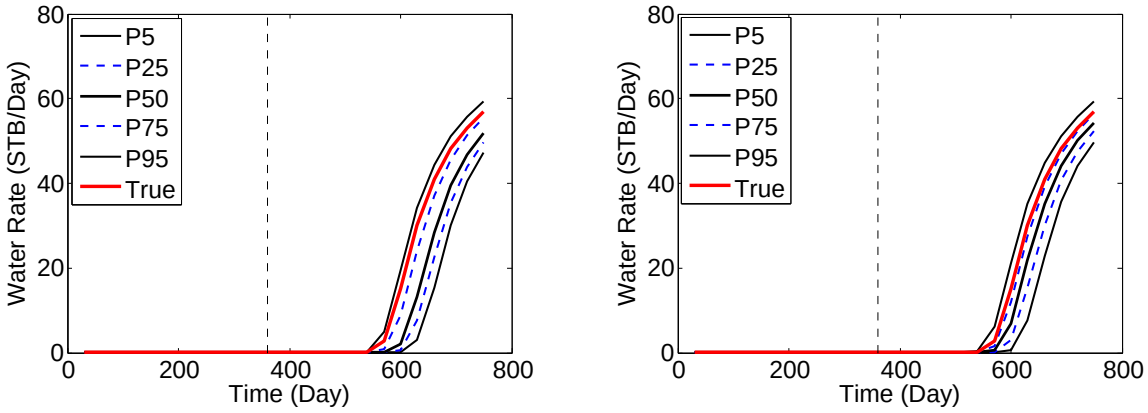
Figure 4.14: Convergence rate calculated using 5 Markov chains of random walk. Only values of MPSRF less than 25 are plotted.



(a) Permeability distribution using the two-level MCMC method.

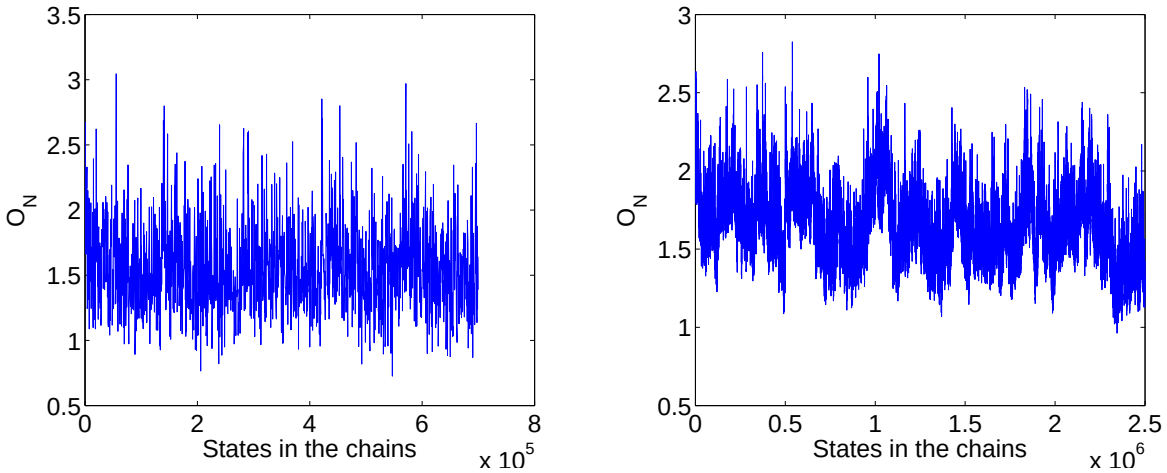
(b) Permeability distribution using the random walk.

Figure 4.15: The posterior distribution of permeability obtained using iterations from 15 to 25 thousand when applying the two-level MCMC method (a) and iterations from 19.5 to 20 million when applying random walk (b) example 2. In both plots, the red curve is the true permeability distribution, the solid black curves from bottom to top are P5, median and P95, the blue dashed curves from bottom to top are P25 and P75.



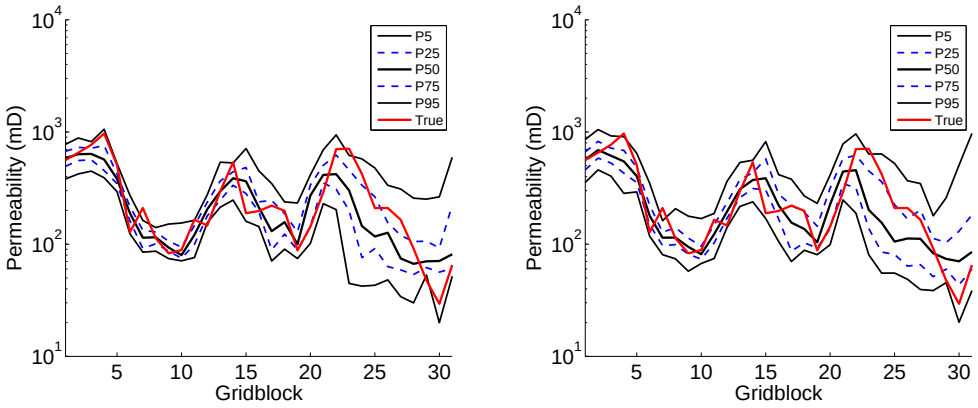
(a) water production rate using the two-level MCMC method. **(b) water production rate using random walk method.**

Figure 4.16: The posterior distribution of water production rate prediction using iterations from 15 to 25 thousand when applying the two-level MCMC method (a) and iterations from 19.5 to 20 million when applying random walk example 2 (b). In both plots, the red curve is the prediction obtained with the true model, the solid black curves from bottom to top are P5, median and P95, the blue dashed curves from bottom to top are P25 and P75.

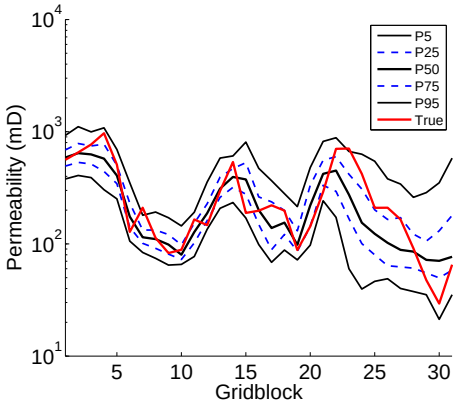


(a) Mixing using two-level MCMC. **(b) Mixing using random walk.**

Figure 4.17: (a) Normalized objective function value of every 2nd state (burn-in discarded) from all the 5 parallel Markov chains using the two-level MCMC. (b) Normalized objective function value of every 100th states (burning discarded) from all the 5 parallel Markov chains using random walk.

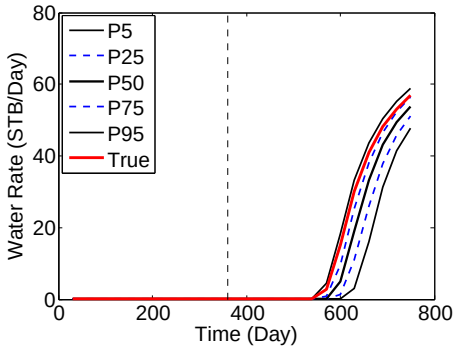


(a) 10 modes. **(b) 25 modes.**

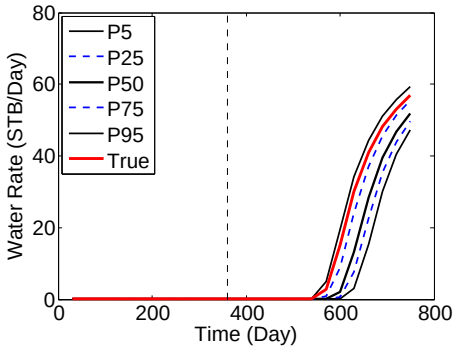


(c) 83 modes.

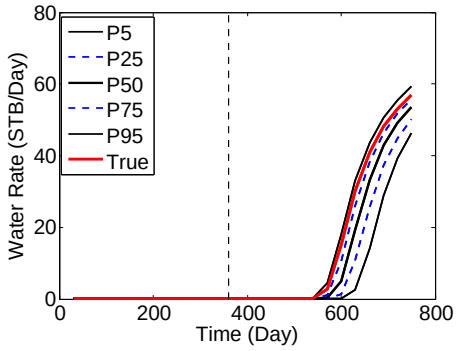
Figure 4.18: The posterior distribution of permeability obtained using iterations from 10 to 15 thousand when applying algorithm 4.2 (a) with 10 modes (b) with 25 modes (c) with 83 modes. In all the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top).



(a) 10 modes.



(b) 25 modes.



(c) 83 modes.

Figure 4.19: Prediction of water production rate obtained using iterations from 10 to 15 thousand when applying algorithm 4.2 (a) with 10 modes (b) with 25 modes (c) with 83 modes. In all the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top). The vertical dashed line separates the history matching and prediction periods.

4.3.4 Example 3

Reservoir model description: We also applied our two-level MCMC algorithm to the IC fault model, which is known to have a complex pdf with multiple local modes. Methods that have been used to estimate the posterior pdf for this model include the genetic algorithms [69], distribution estimation [80] and a population MCMC algorithm [70]. This IC fault model is a 2-D cross-sectional model of a layered reservoir, which is shown in Fig. 4.20. There is a vertical fault in the middle of the reservoir. The grid is 100×12 with each geological layer divided into two simulation layers of equal thickness. Each gridblock is 10 ft wide. The thicknesses from bottom to top are 7.5 ft, 8.5 ft, 9.5 ft, 10.5 ft, 11.5 ft and 12.5 ft. This reservoir consists of six alternating homogenous layers of good and poor quality sands. The porosity of good quality sands and poor quality sands is 0.3 and 0.15, respectively. The model parameters are permeability of good and poor quality sands, represented by k_{high} and k_{low} respectively, and the fault throw thickness denoted by h. The details of the prior information is given in Table 4.1. Water viscosity is 0.37 cp, oil viscosity ranges from 0.51 cp and 0.74 cp over the pressure range from 4041.7 psi to 9014.7 psi. A single water injection well is located on the left edge of the reservoir, and is completed in all layers. It is operated at a constant bottomhole pressure of 8400 psi. The producer at the right edge is also completed in all layers. It is operated at a constant bottomhole pressure of 8335 psi. The historical period corresponds to 3 years, with water injection rate data, oil production rate data and water production rate data assimilated every 30 days. The forecast period ends at 7 years. We added random Gaussian noise with mean equals to zero and standard deviation equal to three percent of the true data to define the measurements data. Because the prior distribution for the model parameter is uniform and we assume that the parameters are independent, the posterior pdf of Eq. 4.1 is simply a constant times the likelihood function given by

$$L(m|d_{obs}) = \exp[-\frac{1}{2}(g(m) - d_{obs})^T C_D^{-1}(g(m) - d_{obs})]. \tag{4.19}$$

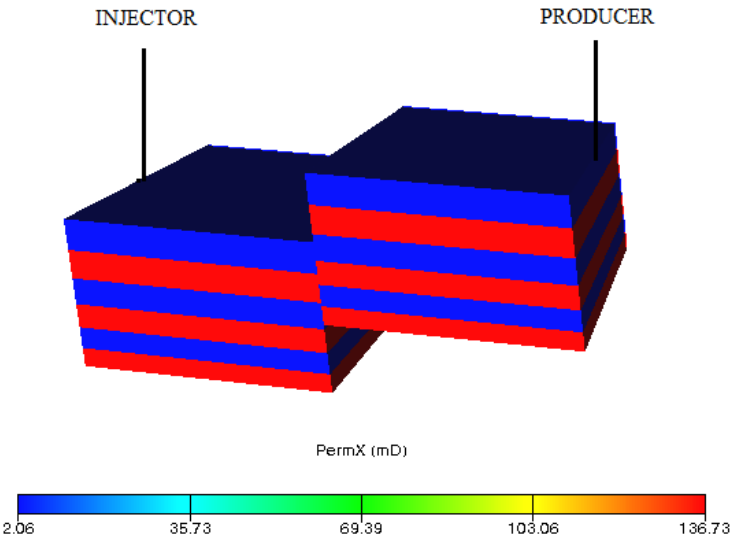


Figure 4.20: Permeability field of IC fault model.

As we wish to compare the results of the two-level scheme with those of generated with population MCMC used by [70], we scale the augment of the likelihood objective function and consequently define the target distribution by Eq. 4.1 with $O(m)$ defined by Eq. 4.3.

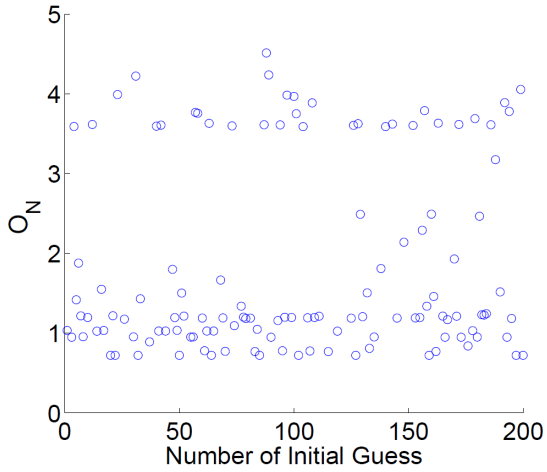
Table 4.1: Parameters for the IC fault model and prior distribution.

Parameter	Units	Prior	True value
Throw thickness	ft	U$[0, 60]$	10.4
k_{low}	md	U$[0, 50]$	1.3
k_{high}	md	U$[100, 200]$	131.6

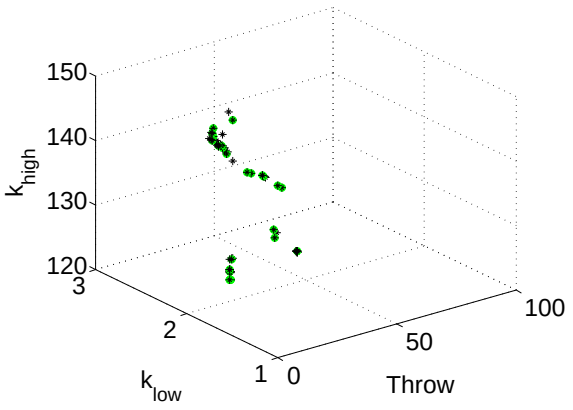
Results: We first tried to use MCMC with the random walk proposal where σ was chosen such that the acceptance rate of the chain is around 0.23. However, we found that using the random walk proposal, we only sample around modes close to the initial guess, i.e., similar to Example 1b, the random walk proposal could not sample the IC-fault model correctly. For two-level MCMC, we generate 200 initial guesses randomly from the prior uniform distributions shown in Table 4.1. Using these initial guesses, we run the quasi-Newton trust region optimization algorithm [13] to minimize $O(m)$ defined by Eq. 4.3. For the IC fault model, which is based on a Eclipse 100 model, we did not have the ability to

compute the gradient of the objective function with the adjoint method. Thus, we were forced to use finite differences to calculate the derivatives involved in the gradient, and the inaccuracy of these finite differences significantly slowed down the convergence of the optimization algorithm. In addition, with three parameters, six simulation runs are required to compute one gradient. Using finite differences to estimate gradients in the optimization algorithm, approximately 32,000 simulation runs were required to generate 200 modes of the posterior pdf. Fig. 4.21(a) shows all the values of $O(m)$ less than 5 that corresponds to a minimizing model obtained by the optimization algorithm. A few values of $O(m)$ greater than 5 were obtained but are not shown. All minimizing model corresponding to a normalized objective function less than 1.5 are used in k-medoids clustering to generate 25 modes which are used to generate a GMM, see Algorithm 4.1. For this case, we did not rescale the covariance matrix, and we use $C = 1$. Fig. 4.21(b) shows the distribution of all the models that give a normalized objective function value less than 5 and all the modes found by k means clustering. Starting from different initial states generated from the initial GMM constructed at the first stage of the algorithm, we run 5 parallel Markov chains with 10 thousand iterations; the change in the value of the MPSRF with iteration number (index of state) is shown in Fig. 4.22. The results of Fig. 4.22 indicate that the chains converge very quickly. Based on the convergence rate (MPSRF), we somewhat aggressively use states from 500 to 2500 iterations of each parallel chain to represent the posterior distribution. Fig. 4.23 presents the distribution of predicted oil production rate and water production rate (percentiles P5, P25, P50, P75 and P95) obtained using the two-level MCMC method. Note that the uncertainties in the water and oil production rates are relatively small, but the spread bounds the true prediction.

The results of Fig. 4.23 are based on 5 parallel Markov chains, with each chain of length 2500, i.e., the total number of simulation runs required is 12,500 $(2,500 \times 5)$. In order to verify that the 10,000 samples (burn-in discarded) used to generate Fig. 4.23 are sufficient to give a good characterization of the posterior pdf, we compare in Fig. 4.24 the results from the 10,000 samples with those obtained by combining all states from all chains



(a) Normalized objective function value after optimization.



(b) Distribution of modes.

Figure 4.21: Example 3: (a) The minimum normalized objective function value from different initial guesses. (b) Distribution of models whose normalized objective function value is less than 1.5 (black) and modes selected by k means clustering (green).

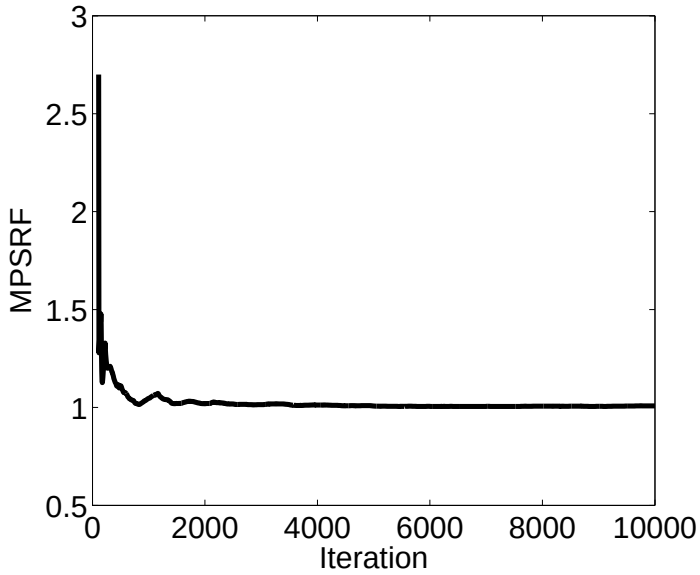
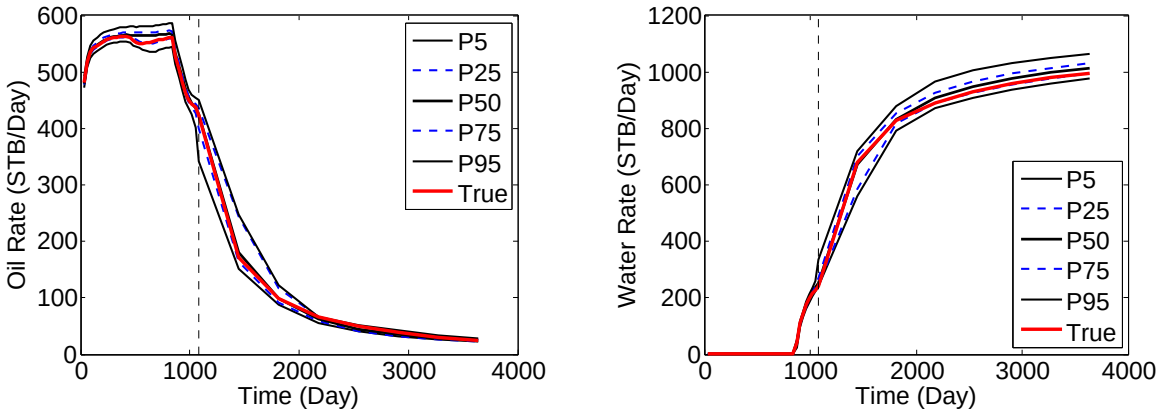


Figure 4.22: Convergence rate calculated using Algorithm 4.2 to generate 5 Markov chains for example 3. Only values of MPSRF less than 10 are plotted.

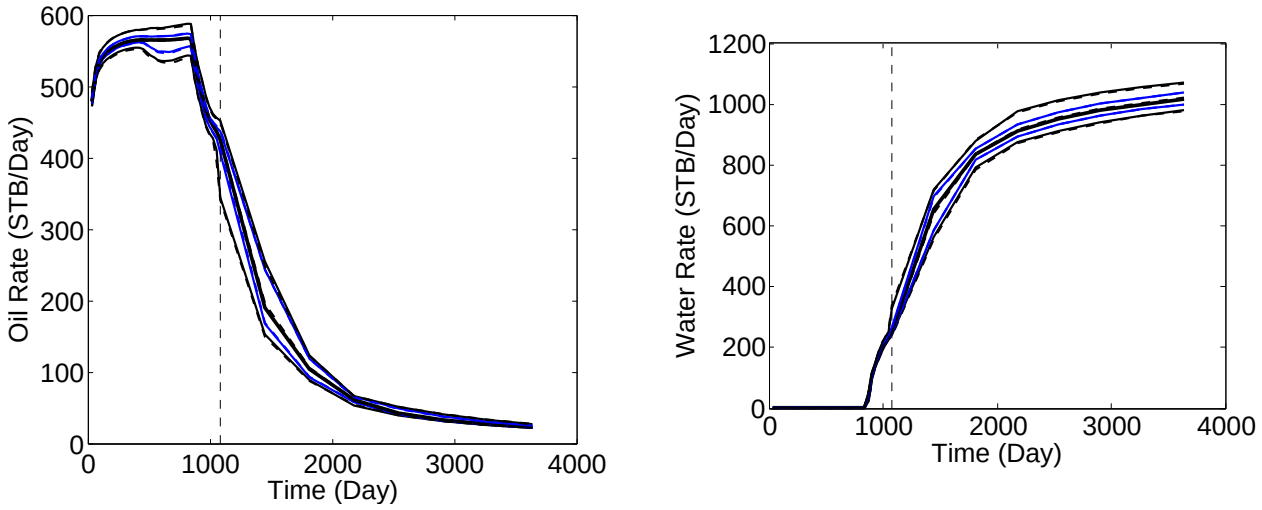


(a) Oil production rate. **(b) Water production rate.**

Figure 4.23: Prediction of oil production rate (a) and water production rate (b) using iterations from 500 to 2500, Algorithm 4.2, example 3. In both of the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top). The vertical dashed line separates the history matching period and the prediction period.

that have a state index from 5,000 to 10,000. As shown in Fig. 4.24, these two samplings give essentially identical estimates of P5, P25, P50, P75 and P95 for the oil and water rates at the producing well for both the historical period, $t < 1100$ days and the future prediction period, $t > 1100$ days. These results suggest that the states from 500 to 2,500 in each chain are sufficient to give a reasonable characterization of the posterior pdf.

For population MCMC algorithm, we follow the same strategy as [70]. To sample the posterior pdf, we run 10 parallel Markov chains each with a different temperature, i.e., each with a different target pdf, and the temperature assigned to each chain is $t_i = (i/10)^5, 1 \leq i \leq 10$. Here, each of the temperature is inversely proportional to the temperature defined in simulated annealing. To characterize the posterior pdf, only the samples with the highest temperature are collected as this is the pdf that corresponds to the target pdf we wish to sample. The other samples corresponding to a lower temperature are used only to prevent being trapped near a local mode. Similar to the two-level MCMC method, to check convergence we also run 5 parallel Markov chains where now each chain has its own 10 parallel chains, one chain for each temperature. For each chain, the initial state is gener-



(a) Oil production rate. **(b) Water production rate.**

Figure 4.24: Two-level MCMC: comparison of oil production rate (a) and water production rate (b) generated by samples from iteration 500 to 2500 (in solid curves) and samples from iteration 5000 to 10 thousand (in dashed curves). In both figures, for both the dashed curves and solid curves, the curves in black from bottom to top are P5, median and P95 and the curves in blue are P25 (bottom) and P75 (top). The vertical dashed line separates the history matching period and the prediction period.

ated randomly from the prior distribution. Fig. 4.25 presents the convergence rate using the population MCMC method. Based on the convergence rate shown in Fig. 4.25, we mixed the samples from states 1000 to states 2000 of each chain to characterize the posterior distribution, with results for the probability percentiles shown in Fig. 4.26. Comparing the posterior distribution shown in Fig. 4.26 with those obtained by our Algorithm 4.2 shown Fig. 4.23, we see that P95 − P5 uncertainty band for the water rate of Fig. 4.26(b) is wider than the corresponding P95−P5 uncertainty band obtained with Algorithm 4.2 (Fig. 4.23(b)), and for times less than 1,000 days, the oil rate uncertainty band of Fig. 4.26(a) is also greater than the one of Fig. 4.23(a). We provide a possible explanation for this difference in the uncertainty band later. The computation cost of the population MCMC is relatively high, because generating a new state using the required Metropolis-within-Gibbs [57] requires evaluating the objective function for the dimension of the model parameters times. For each parallel chain with 10 different temperatures, we need 10 (the number of different temperatures) × 3 (Metropolis-within-Gibbs)= 30 simulation runs to generate a new proposal. To check the

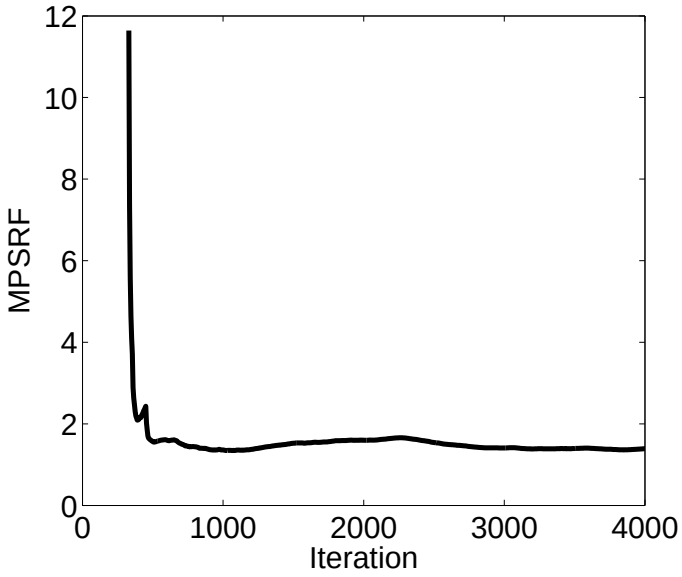
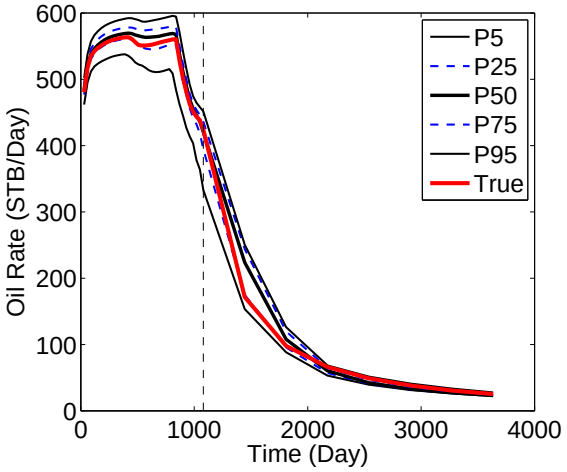


Figure 4.25: Convergence rate of population MCMC for example 3 calculated using 5 Markov chains. Only values of MPSRF less than 12 are plotted.

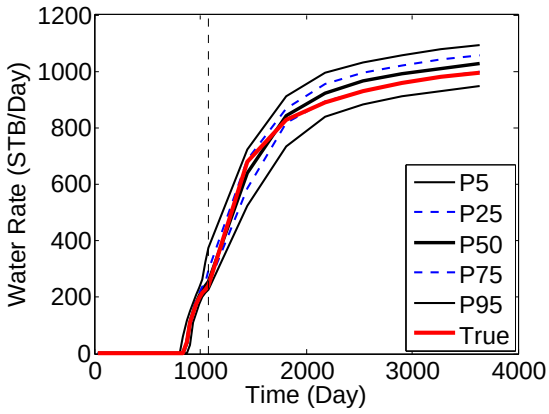
convergence, we run 5 parallel Markov chains, so for each iteration in Fig. 4.25, 150 (30×5) simulation runs are required. Thus, generating the results of Fig. 4.25 requires a total of 600 thousand ($4,000 \times 150$) simulations.

To check whether we used enough samples to characterize the posterior pdf with this population MCMC method, we also compare posterior probability percentiles for the oil and water production rate obtained using states from iterations 1000 to 2000 and to those obtained using states from iterations 2000 to 4000. The comparison is shown in Fig. 4.27. Note that there is a very small difference between the two distributions for the 5th percentile of the oil production rate and for the 75th percentile of the water production rate, but overall the distributions from the two sets of realization are very close.

Fig. 4.28 presents the normalized objective function value O_N, which is the same as the objective function defined in Eq. 4.3, for all the accepted states in all the 5 parallel Markov chains. Comparing Fig. 4.28(a) and Fig. 4.28(b), the mixing of the two-level MCMC chains is better than the mixing using the population MCMC method. In Fig. 4.28(b), the normalized objective function values of the third Markov chain represented by states 4000 to 6000 are greater than the the value of other Markov chains, and this could be the reason

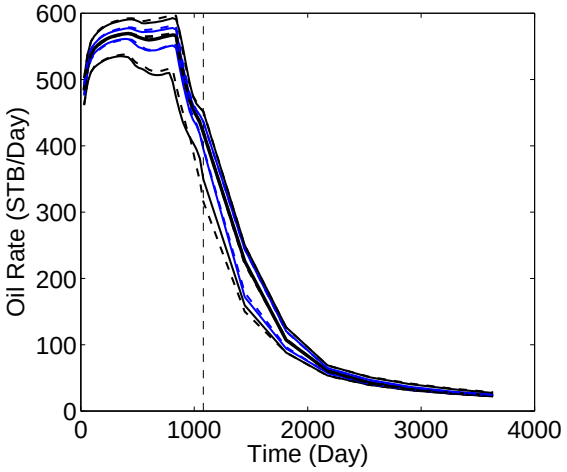


(a) Oil production rate.

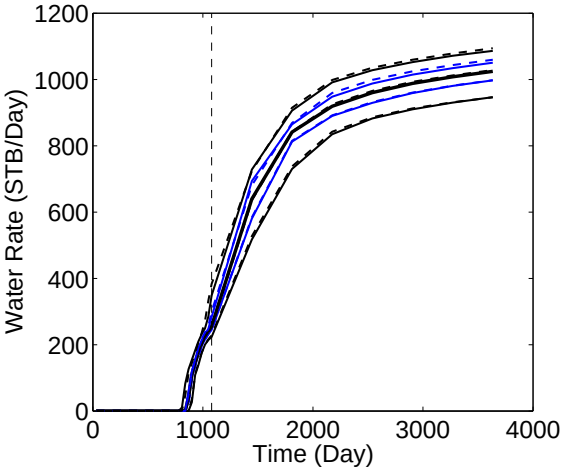


(b) Water production rate.

Figure 4.26: Population MCMC for example 3: prediction of oil production rate (a) and water production rate (b) using iterations from 1000 to 2000. In both of the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top). The vertical dashed line separates the history matching period and the prediction period.

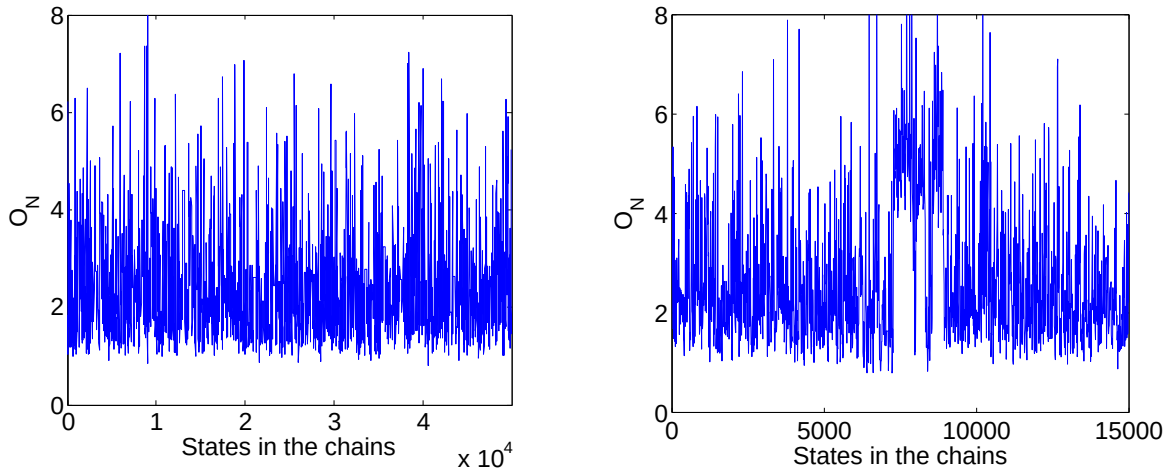


(a) Oil production rate.



(b) Water production rate.

Figure 4.27: Population MCMC: comparison of oil production rate (a) and water production rate (b) generated by samples from iteration 1000 to 2000 (in solid curves) and samples from iteration 2000 to 4000 (in dashed curves). In both of the figures, for both of the dashed curves and solid curves, the curves in black from bottom to top are P5, median and P95, the curves in blue are P25 (bottom) and P75 (top). The vertical dashed line separates the history matching period and the prediction period.



(a) Mixing using two-level MCMC. **(b) Mixing using population MCMC.**

Figure 4.28: (a) Normalized objective function value of every 100th accepted state (burn-in discarded) from all the 5 parallel Markov chains using two-level MCMC. (b) Normalized objective function of every 10th accepted state (burn-in discarded) from all the 5 parallel Markov chains using population MCMC.

why the convergence rate of the 5 parallel chains can only be reduced to a value of about 1.5, also, and as noted earlier the $P95 - P5$ uncertainty range for water rate is larger than the corresponding $P95 - P5$ result obtained with Algorithm 4.2.

4.3.5 Example 4

Although it is anticipated here that the uncertainty quantification methodology proposed perhaps will be most useful when the number of parameters is reduced to 15 to 40 as is often done in field history matching applications, we explore here whether the methodology is computationally feasible for a larger problem that those considered in Examples 1 through 3.

Reservoir model description: This case considered next pertains to a synthetic reservoir defined on a 44×44 uniform reservoir simulation grid. The dimension of each gridblock is $100 \times 100 \times 15\,\text{ft}$. The model parameters are gridblock log permeabilities so the dimension of the vector of model parameters is 1936. The true model was built using an exponential covariance function with major correlation length $2,500$ft and minor correlation length

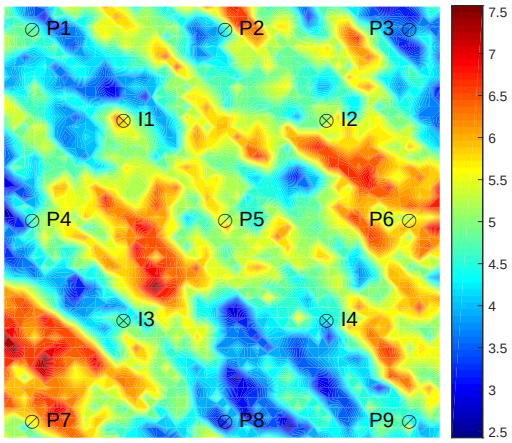


Figure 4.29: True log-permeability field.

1, 100ft. The angle between the positive y-axis and the principle direction of the covariance function is 45°. The prior mean of ln(k) is 5.0, the prior variance of ln(k) is 1.0. Fig. 4.29 shows the true log-permeability field from which the observed data are generated. For this reservoir, there are nine production wells and four water injection wells. Fig. 4.29 also shows the location of wells. The initial pressure of the reservoir is 3000 psi. All the production wells and injection wells operated under bottomhole pressure control. The injection wells operate at a constant bottomhole pressure of 4500 psi while the production wells operate at a bottomhole pressure of 2800 psi for the first six months then operate at a constant bottomhole pressure of 2500 psi for the rest of the simulation. The observed data are oil- and water-production rates and water-injection rate, and they are measured every month during the first 36 months. To generate the observed data, Gaussian random noise with zero mean and standard deviation equal to 5% (with a minimum of 2 STB/d) are added to the rate data predicted by the true model. The history-matching period is 36 months, and the subsequent prediction period is 20 months. We assimilate data every 30 days, and for the whole history match period, the total number of observed data is 792. As the prior pdf is Gaussian and the measurement errors are Gaussian, the posterior pdf is given by Eq. 4.1, where the objective function $O(m)$ is given by Eq. 4.2.

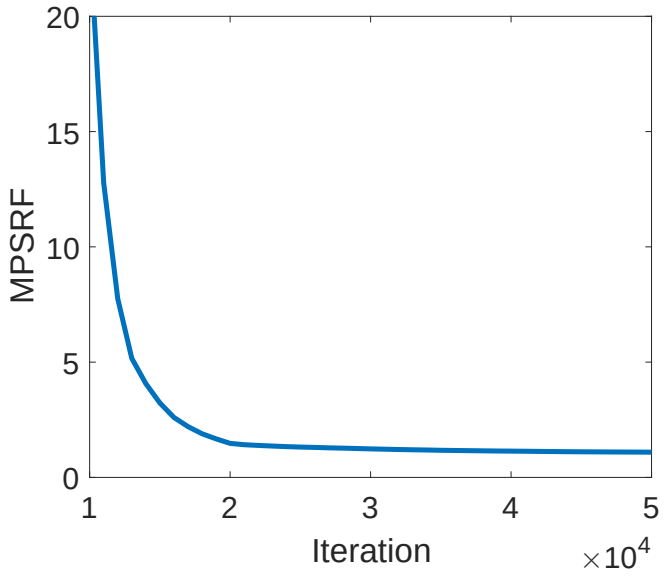


Figure 4.30: Convergence rate (MPSRF) of Algorithm 4.2 calculated using 5 Markov chains. Only values of MPSRF less than 20 are plotted.

Results: To construct the GMM in Algorithm 4.1, we use 250 initial guesses randomly generated from the prior Gaussian distribution of the model parameters, 69,000 reservoir simulation runs were used to estimate the initial Gaussian mixture model. Running such a large number of reservoir simulation runs is not feasible in practise so later, we propose a more computationally efficient method. In this case, we use all minimizing models which result in a normalized objective function value less than 1.5 to generate 35 modes by k-medoids clustering and these modes are used to construct the initial GMM proposal distribution for Algorithm 4.2. In order to improve the computational efficiency, we rescale the covariance matrices to $0.1 \times C_\ell$. To check the convergence of Algorithm 4.2, we also run 5 Markov chains in parallel starting from five different initial states generated randomly from the proposal distribution (GMM). Fig. 4.30 presents the convergence results, i.e., the value of MPSRF versus the chain index, it indicates that the five Markov chains converge after about 20 thousand iterations. Thus, we use the samples from 20 thousand to 30 thousand iterations to generate the posterior distribution. The overall acceptance rate for this case is 0.23.

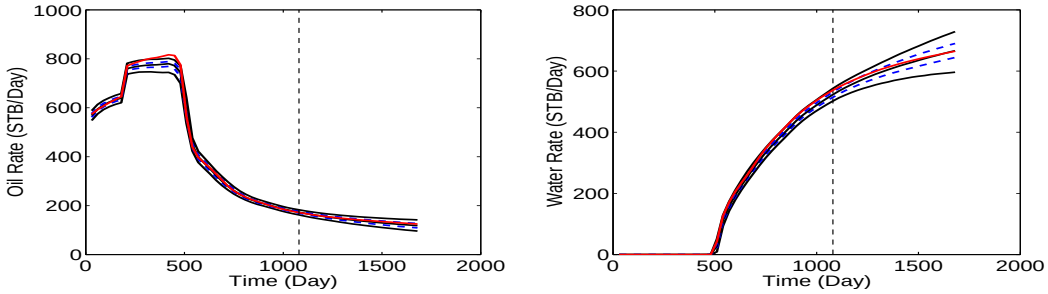
We did not compare our results with random walk for this case, because the computational time required to run random walk for this case was prohibitive. Fig. 4.31 shows the

posterior distribution of oil production rate and water production rate in Producer P3, Producer P5 and Producer P8. The results shown encompass qualitatively results for all wells. For both the water and oil production rate, the prediction from the true reservoir model is between the $P5$ and $P95$ results, except for the case of the water production rate of well 5 where the true prediction is very maginally higher than the $P95$ prediction at the very end of the prediction period. Fig. 4.32 presents the posterior distribution of water injection rate for all the wells obtained using Algorithm 4.2. For well 1, in the history matching period, the true water injection rate is close to $P95$, whereas in the prediction period, the true is higher than $P95$. For all the other wells, the true water injection rate is in the band between by $P5$ and $P95$.

Even ignoring the tens of thousands of reservoir simulation runs necessary to compute 250 minimizing model, the cost of running such example in practice would be prohibitive because it required generating about 20,000 states in each of five chains to obtain convergence, and the generation of each state requires one reservoir simulation run. Thus, in the next chapter, we seek a more efficient way to increase the convergence rate in order to reduce the number of states required in the MCMC algorithm to provide an approximate sampling of the target pdf.

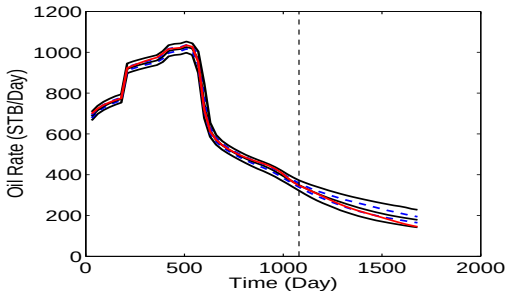
4.4 Possible Modifications

The weights in the proposal pdf (GMM) could affect the efficiency of the two-level MCMC method. If the modes of the posterior pdf are widely separated by very low probability regions, using the weights proportional to the unnormalized target pdf values $\exp(-O(m))$ at convergence is preferable because it could make the GMM closer to the target pdf. However, it appears that this choice is not optimal if modes are not separated by low probability regions [81]. To make the proposal pdf closer to the target pdf, we can adapt the weights in the GMM during the sampling process. To add weight adaptation to Algorithm 4.2, at the end of the current iteration i, the weight of the jth Gaussian ω_j is adapted based on the

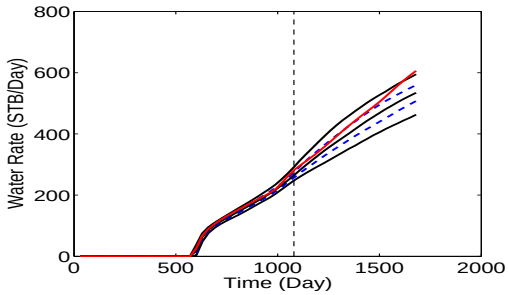


(a) Oil Rate Well 3.

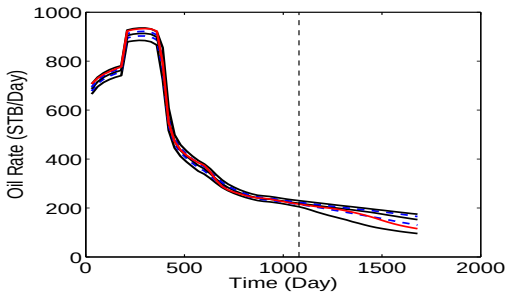
(b) Water Rate Well 3.



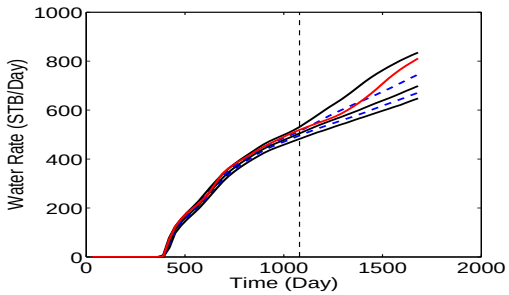
(c) Oil Rate Well 5.



(d) Water Rate Well 5.

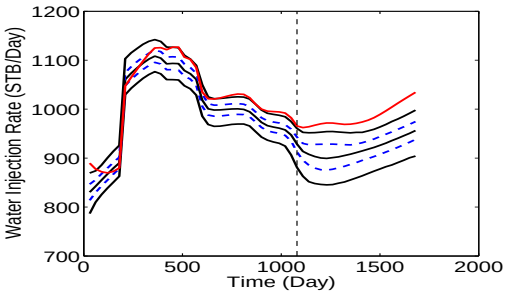


(e) Oil Rate Well 8.

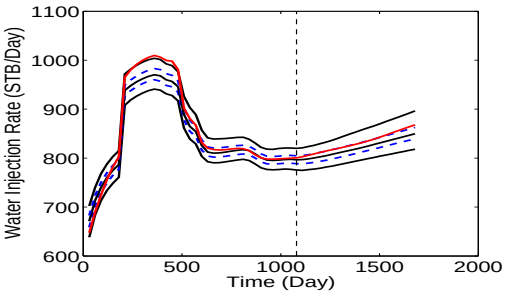


(f) Water Rate Well 8.

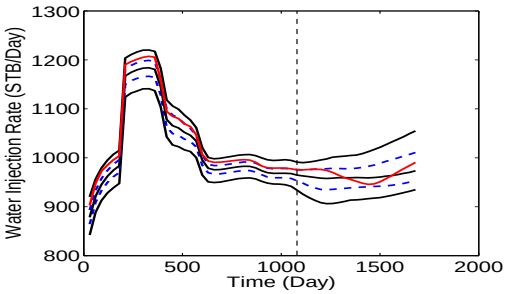
Figure 4.31: Example 4: the prediction of oil production rate and water production rate of algorithms 4.1 using states from 20 thousand to 30 thousand. In all the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top). The vertical dashed line separates the history matching and prediction periods.



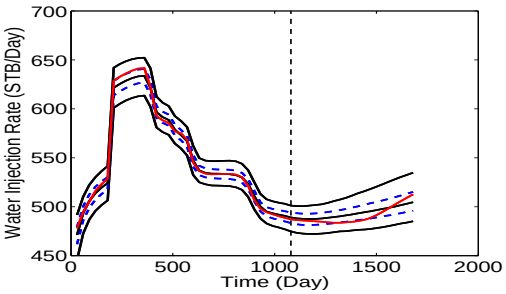
(a) Water Injection Rate Well 1.



(b) Water Injection Rate Well 2.



(c) Water Injection Rate Well 3.



(d) Water Injection Rate Well 4.

Figure 4.32: The prediction of water injection rate using states from 20 thousand to 30 thousand of algorithms 4.1. In all the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top). The vertical dashed line separates the history matching and prediction periods.

following equation:

$$\omega(j) = \frac{i_j}{i}, \tag{4.20}$$

where i_j is the number of states that belong to the jth Gaussian. There are two issues related to the weight updating. The first issue relates to the fact that it does not make sense to start updating the weights when we generate the first accepted state that belongs to Gaussian. If the first accepted state belongs to Gaussian k, then we would set $\omega_k(i) = 1$ and all other weights to zero, which is clearly unreasonable. Although we have not fully explored this issue here, we conjecture that weights should not be changed (adapted) until we have accepted at least $\ell \times n_g$ proposed states in the chain where n_g denotes the number of Gaussians in the GMM proposal distribution and ℓ is a positive integer. We also conjecture that ℓ should be between 10 and 1000, but we have not explored this choice in this work. We also do not start to adapt weights until the chain contains at least one sample from each Gaussian in the GMM proposal distribution to avoid setting the weight of the original Gaussian equal to zero, the second issue is the equation of which Gaussian an accepted state should be assigned to. While the simple answer would be to assign the new state to the GMM Gaussian that it is sampled from, we use here a different procedure. Specifically we assign the state m_i to the jth Gaussian where j is determined by

$$j = \arg\min_{\ell} ||m_i - m_\ell^*||. \tag{4.21}$$

In this example, we only update the weight when there are more than 200 (100×2) states in the chain and there is at least 1 state accepted from each of the Gaussians.

To investigate whether this weight adaptation combined with covariance matrix adaptation improves the computational efficiency, we try the three different sets of initial weights shown in Table 4.2 for the toy problem of example 1a. Here, ω_1 and ω_2 are weights in the proposal GMM; see Eq. 4.22 below. The third set of weights is based on the value of the target pdf without the normalizing constant, i.e., $\exp(-O(m))$, evaluated at each mode. For all weights, the proposal GMM is given by

Table 4.2: Different weight in GMM

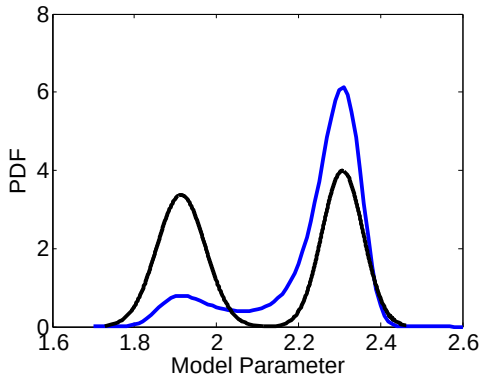
Weights	ω_1	ω_2	Acceptance Rate without Adapting Weights
Set 1	$\frac{2}{3}$	$\frac{1}{3}$	0.56
Set 2	$\frac{1}{2}$	$\frac{1}{2}$	0.69
Set 3	0.88	0.12	0.80

$$p(m) = \omega_1 G(2.3077|m, 0.0025) + \omega_2 G(1.9141|m, 0.0035). \tag{4.22}$$

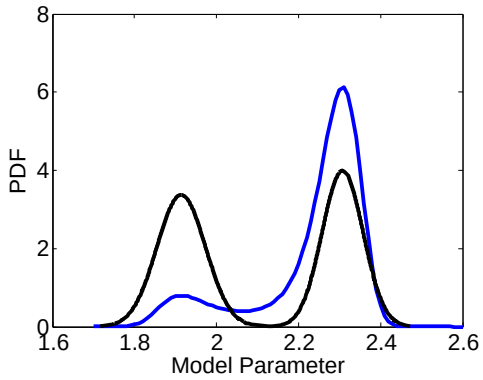
For all three sets of weights, we can use the states from the 500th iteration to the 2000th iteration of all the chains to generate the posterior distribution. In Fig. 4.33, we show the comparison of target pdf and proposal (GMM) pdf using the three different sets of weights of Table 4.2. Note that the proposal distribution using the weights of set 3 gives the proposal distribution which is closest to the target pdf. Fig. 4.34 shows the comparison of target pdf and proposal (GMM) pdf after adaptation (at the end of the Markov chain) using different sets of weights. With weight adaptation, the final GMM proposal distributions are all essentially equal (Table 4.3) and fairly close to the target pdf (Fig. 4.34). Fig. 4.35 presents the convergence rate using the different sets of weights, it indicates that the chain converges faster using the weight in set 3 because these initial weights gives a good initial proposal GMM. Fig. 4.36 indicates that the distribution of the samples in the Markov chain is exactly the same as the target pdf.

Table 4.3: Acceptance rate using different weights in GMM

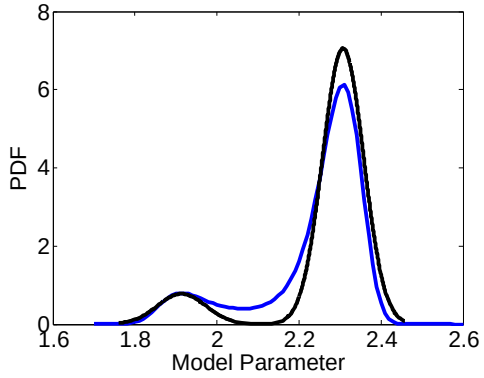
	ω_1	ω_2	Acceptance Rate	ω_1 after Adaptation	ω_2 after Adaptation
Set 1	$\frac{2}{3}$	$\frac{1}{3}$	0.8	0.84	0.16
Set 2	$\frac{1}{2}$	$\frac{1}{2}$	0.8	0.84	0.16
Set 3	0.88	0.12	0.8	0.84	0.16



(a) Set 1.

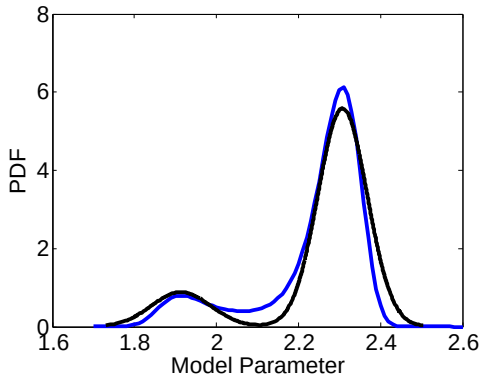


(b) Set 2.

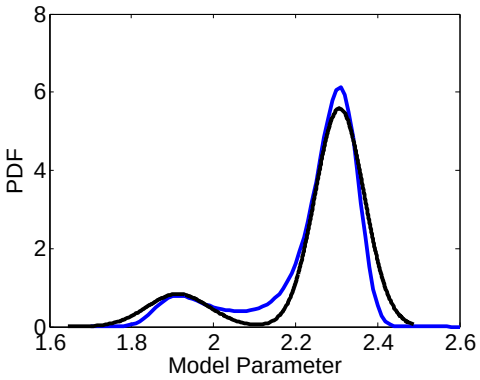


(c) Set 3.

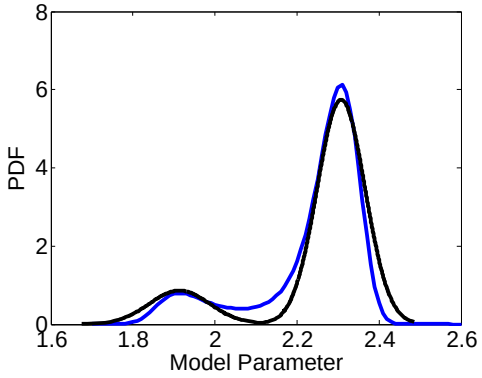
Figure 4.33: Comparison of target pdf in blue and proposal (GMM) pdf in black using the sets of weights of Table 4.3 in the GMM.



(a) Set 1.

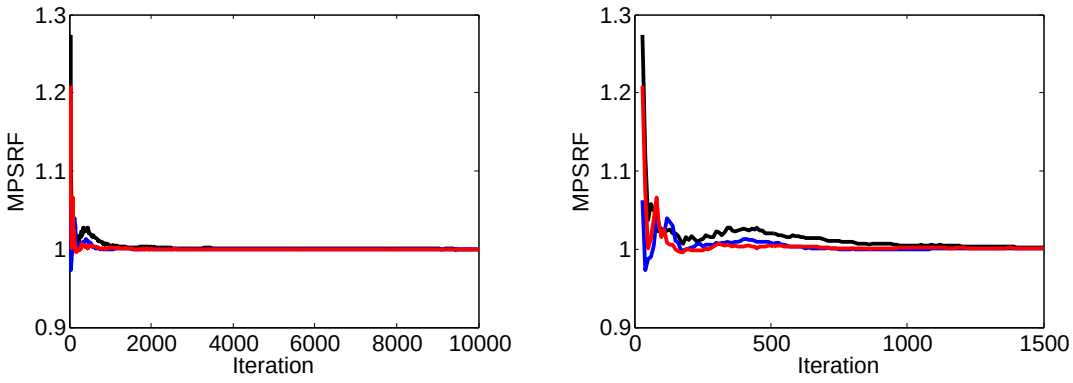


(b) Set 2.



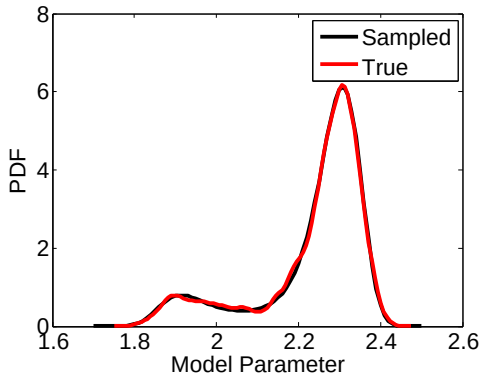
(c) Set 3.

Figure 4.34: Comparison of target pdf in blue and proposal (GMM) pdf after adaptation (of both weight and covariance matrix) in black using different sets of weight in the GMM.

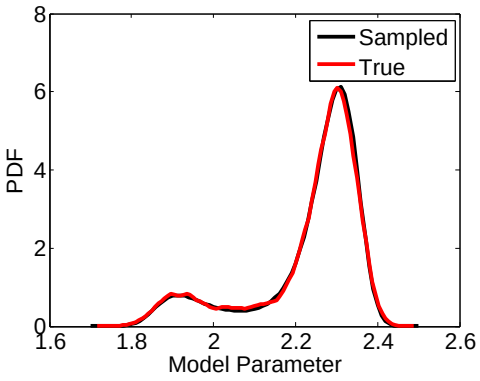


(a) Convergence comparison (1-10,000). **(b) Convergence comparison (1-1,500).**

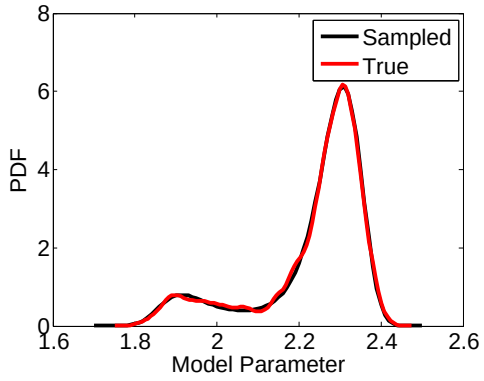
Figure 4.35: Comparison of convergence rate (MPSRF) using different sets of inital weights with both weight and covariance matrix adaptation for example 1a, blue curve obtained from set 1, red curve obtained from set 3, black curve obtained from set 2. (a) shows the convergence rate from iteration 1 to 10,000. (b) shows the convergence rate from iteration 1 to 1,500.



(a) Posterior distribution using weight in set 1.



(b) Posterior distribution using weight in set 2.

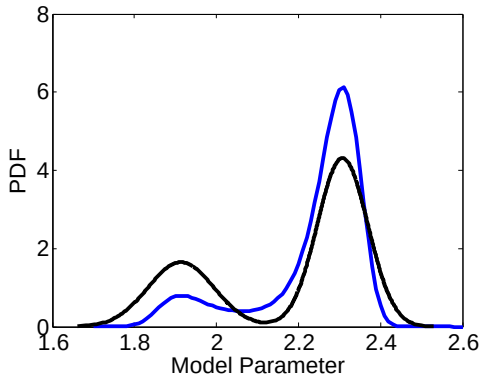


(c) Posterior distribution using weight in set 3.

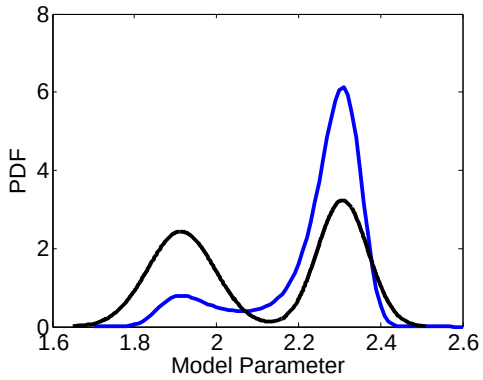
Figure 4.36: Comparison of sampled pdf (weight and covariance matrix adaptation) with target pdf using different sets of weight in the GMM. (a) Distribution of samples from the 500th iteration to 2000th iteration of all the chains using weights in set 1; (b) Distribution of samples from the 500th iteration to 2000th iteration of all the chains using weights in set 2; (c) Distribution of samples from the 500th iteration to 2000th iteration of all the chains using weights in set 3;

Without the weight adaptation but with covariance matrix adaptation, the MCMC acceptance rate for set 1, set 2 and set 3, respectively are 0.56, 0.69 and 0.8 (Table 4.2). After we apply the weight adaptation with covariance adaptation, the acceptance rate increases to 0.8 for all sets of weights as shown in Table 4.3.

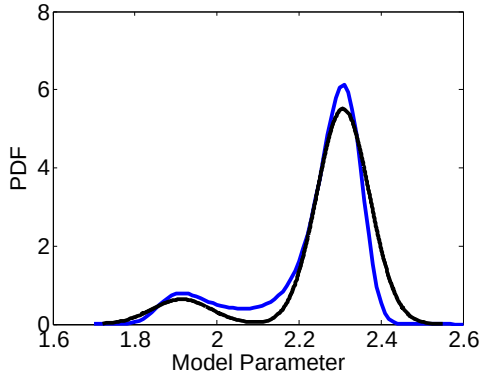
Fig. 4.37 shows the comparison of target pdf and proposal pdf after covariance matrix adaptation using different sets of weights. Using the covariance matrix adaptation only, the final GMM proposal distributions using weights in set 1 (Fig. 4.37(a)) and set 2 (Fig. 4.37(b)) are very different from the target pdf, because the weights in the GMM are not adapted to a value close to the target pdf. However, the proposal distribution using weights in set 3 (Fig. 4.37(c)) is very similar with the target pdf, because the original weights in set 3 are already very close to the weights in the target pdf. Fig. 4.38 presents the convergence rate using different sets of weights, compare with Fig. 4.35 where both weights and covariance matrix are adapted, the chains with only the covariance matrix adaptation converge faster. However, because of the high acceptance rate achieved using both weight and covariance matrix adaptation (weights in set 1 and set 2), the number of states required to form the correct posterior distribution is less than the number of states required to form the correct posterior distribution using covariance matrix adaptation only (weights in set 1 and set 2). Fig. 4.39 indicates that the distribution of the samples in the Markov chain is exactly the same as the target pdf.



(a) Set 1.

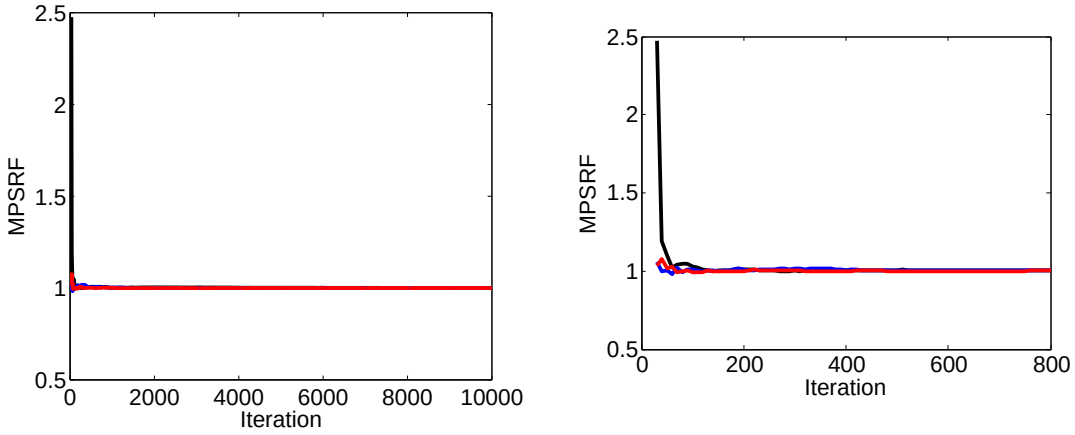


(b) Set 2.



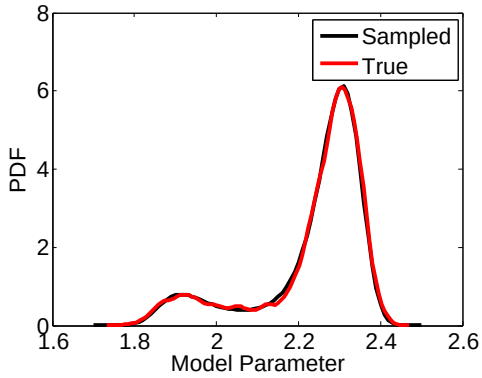
(c) Set 3.

Figure 4.37: Comparison of target pdf in blue and proposal (GMM) pdf after adaptation (of covariance matrix only) in black using different sets of weight in the GMM.

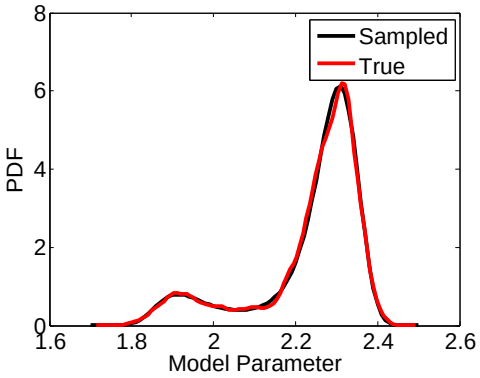


(a) Convergence comparison (1-10,000). **(b) Convergence comparison (1-1,000).**

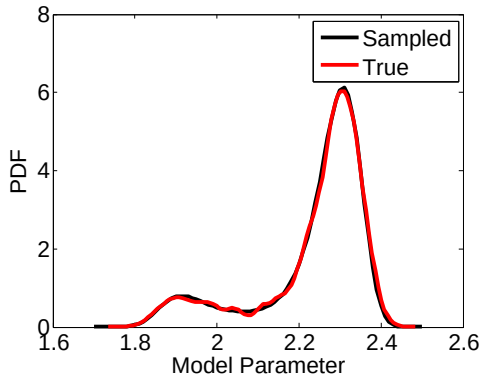
Figure 4.38: Comparison of convergence rate (MPSRF) using different set of weight with only the covariance matrix adaptation. In both (a) and (b), blue curve is obtained from set 1, red curve is obtained from set 2, black curve is obtained from set 3. (a) shows the convergence rate from iteration 1 to 10,000; In order to compare the convergence rate using different sets of weight, (b) shows the convergence rate from iteration 1 to 1,000.



(a) Set 1.



(b) Set 2.



(c) Set 3.

Figure 4.39: Comparison of sampled pdf (covariance matrix adaptation) with target pdf using different sets of weight in the GMM. (a) Distribution of samples from the 500th iteration to 2500th iteration of all the chains; (b) Distribution of samples from the 500th iteration to 2500th iteration of all the chains; (c) Distribution of samples from the 500th iteration to 2000th iteration of all the chains;

CHAPTER 5

AN APPROXIMATE TWO-LEVEL MCMC ALGORITHM

Although the two-level MCMC algorithm (Algorithm 4.2) can sample the target pdf correctly, for high-dimensional problems the computation cost of this algorithm is still high. In Example 4, for the 44×44 case, it required 100,000 simulation runs to obtain convergence of all five Markov chains. To reduce the computational cost, in this chapter, we present a more efficient but approximate way to sample the target pdf.

5.1 An approximate two-level MCMC algorithm

To lower the computation cost of the proposed two-level MCMC method, we propose an approximate two-level MCMC method and apply it to example 1, 2, 3 and 4. For each of the examples we compare the results using Algorithm 4.2 and Algorithm 5.1 presented below. The steps of the second two-level MCMC method are given below.

Algorithm 5.1 - Approximate Two-Level MCMC

1. Apply Algorithm 4.1 to obtain a Gaussian mixture model. This step represents Level 1 and steps 2-7 represent Level 2.

2. Generate an equal number (n_s) of samples from each Gaussian and form a set S which contains all samples from all Gaussians. The number n_s is chosen by the user based on computational resources. For example, at step 1, if the number of Gaussians in the GMM is 25 and at step 2, we generate 200 samples from each of the 25 Gaussians, then there are 5,000 (200×25) samples in the set S which means we need to run the forward model 5000 times to compute the necessary values of the unnormalized target pdf.

3. Set $i = 0$ and sample m_0 randomly from the set S.

4. Propose a new state, $\widetilde{m}_{i+1}$, randomly from the set S, and define the acceptance probability

$$\alpha(m_i, \widetilde{m}_{i+1}) = \min(1, \frac{\pi(\widetilde{m}_{i+1})q(m_i)}{\pi(m_i)q(\widetilde{m}_{i+1})}), \tag{5.1}$$

where the Gaussian mixture proposal pdf is given by

$$q(m) = \sum_{\ell=1}^{k} \frac{1}{k}\mathcal{N}(m_\ell^*, C_\ell) = \sum_{\ell=1}^{k} \frac{1}{k}G(m|m_\ell^*, C_\ell). \tag{5.2}$$

Here, k is the number of modes selected by k means clustering; m_ℓ^* and C_ℓ are the mean and covariance matrix associated with the ℓth selected mode.

5. Generate a random number u from the uniform distribution $U(0, 1)$.

6. If $u \leq \alpha(m_i, \widetilde{m}_{i+1})$, the new proposed state is accepted and set $m_{i+1} = \widetilde{m}_{i+1}$. Otherwise, repeat the current state in the chain, i.e., $m_{i+1} = m_i$. If the next state m_{i+1} is repeated in the chain for more than 20 times, find the closest Gaussian mode to m_{i+1}, i.e., find the Gaussian index in the GMM such that

$$j = \arg\min_k ||m_{i+1} - m_k^*||. \tag{5.3}$$

Update the covariance matrix associated with the jth Gaussian in the GMM using

$$C_j^{(i+1)} = C_j^{(i)} + \frac{1}{i}[(m_{i+1} - m_j^*)(m_{i+1} - m_j^*)^T - C_j^{(i)}]. \tag{5.4}$$

Here, m_j^* is the mean associated with the jth Gaussian in the GMM and $C_j^{(i+1)}$ is the covariance matrix associated with the jth Gaussian in the $i + 1$th iteration. Unlike Algorithm 4.2, we set the gain factor equal to 1 directly. Because the point of this algorithm is to save computational cost and we do not need to run simulation during the sampling process, we skip the experiment process to tune the gain factor to have

a good acceptance rate.

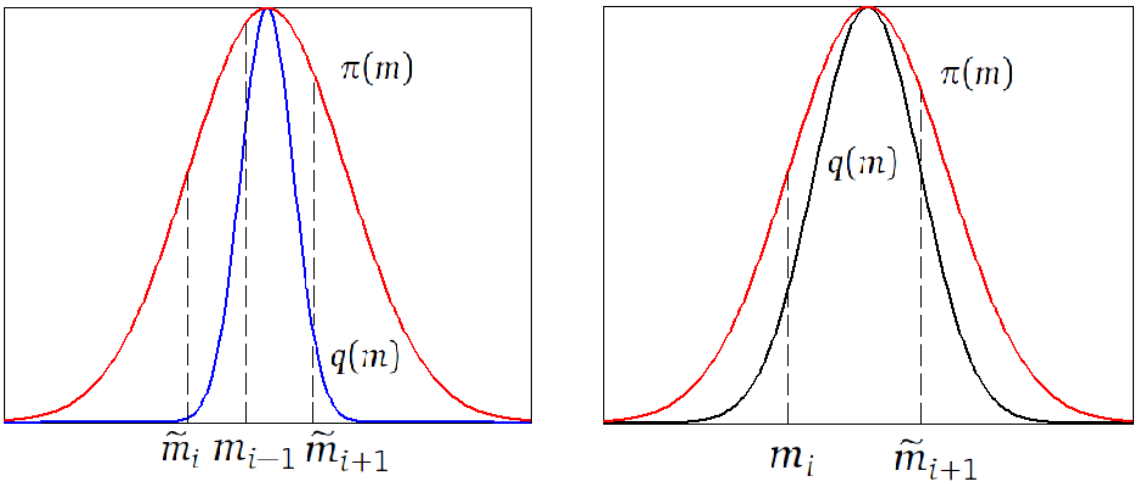
7. Set $i = i + 1$ and go to step 4, until the number of states reaches the desired number.

We need to make some important comments pertaining to the adaptation step in Algorithm 5.1. (i) Note that the initial Gaussian mixture model, all of the Gaussians have equal weights. Because of this, generating 200 from each Gaussian mixture model provides a correct procedure for sampling the initial GMM model. Thus, the application of Algorithm 5.1 with the adaptation step deleted, provides a correct procedure to generate samples from the target pdf ($\pi(m)$). However, one cannot expect to obtain a highly accurate sample of the high-dimensional Gaussian with only 200 samples so in the high-dimensional case, we will obtain a very approximate approximation of the target pdf. (ii) Note that the adaptation procedure used in Algorithm 5.1 is different than the adaptation procedure used in Algorithm 4.2. However, the more important point is that regardless of the adaptation methodology used, whenever we modify the covariance of an individual Gaussian by adaptation, the 200 initial samples generated using the initial covariance of that Gaussian, do not represent a correct sampling of the updated (adapted) GMM so we introduce a theoretical flaw into the algorithm. That is, when we choose a sample from the 200 samples of the original Gaussian the correct probability of proposing that sample is obtained by evaluating the initial GMM rather than at the adapted GMM. It follows that if we use the adapted GMM in Metropolis-Hasting and do not change the initial set of samples, then we introduce a theoretical flaw into the Metropolis-Hastings acceptance probability so we have no assurance that we sample the target distribution correctly. On the other hand, if we use the initial Gaussian mixture model, in Metropolis-Hastings, then we have not used the adaptation for any purpose at all. (iii) The preceding comment applies to any adaptation procedure used in Algorithm 5.1. We could of course eliminate the aforementioned theoretical flaw by discarding all samples from the initial Gaussian and resampling the updated Gaussian but then we are effectively using Algorithm 4.2, and thus forego the computational efficiency that Algorithm 4.2 was designed to achieve by placing a limit on the number of times that we have to evaluate the

target distribution to 200 times the number of Gaussians in the proposal GMM. (iv) Each evaluation of the target pdf requires one run of the forward model. For the quantification of uncertainty in reservoir modeling and prediction, each forward model represents one reservoir simulation run which can take minutes to several hours so for such problems, it is critical that we reduce the number of forward model evaluations used to characterize uncertainty. In fact, in practice, we may need to reduce the number of samples from each Gaussian to a number well below 200.

In light of the preceding comments, we adopt a tradeoff between theoretical rigor and computational feasibility. Namely, we do change the set of samples generated from the initial GMM when we update the covariance matrices of the Gaussians but in an attempt to reduce the aforementioned theoretical error that emanates from not replacing these original samples with samples from the updated GMM, we only update when a Markov chain repeats the same state from the same Gaussian 20 times, i.e., when 20 consecutive states in the chain are from the same Gaussian of the GMM. This particular updating scheme seems heuristic but the updating methodology has an intuitively logical basis as is discussed later. Although the method is still not theoretically rigorous, the computational examples we have done show this updating scheme, gives an uncertainty characterization that is similar to one that is obtained from Algorithm 4.2, which is a theoretically sound algorithm. On the other hand, if we implement the modification of Algorithm 5.1 obtained by updating the covariance matrices at each time a new state is generated and do not change the original set of samples from the initial Gaussian mixture model, then results not included here show that a very poor characterization of the target pdf is obtained. Although the results are not shown here, it is important to note that if we replace the covariance matrix adaptation scheme of Algorithm 4.2 with the adaptation scheme of Algorithm 5.1, we obtain approximations of the target pdf that are extremely close to those using Algorithm 4.2.

When the Gaussian distribution cannot provide a good approximation of a local mode, the situation in Fig. 5.1(a) could happen. Suppose in the Markov chain the current state is m_{i-1}, and a new state $\widetilde{m}_i$ is proposed from the proposal distribution. The acceptance



(a) Before Adaptation. **(b) After Adaptation.**

Figure 5.1: Explanation of how covariance adaptation effectively broadens the proposal distribution. (a) shows the target pdf and proposal pdf before adaptation; (b) shows the target pdf and proposal pdf after adaptation.

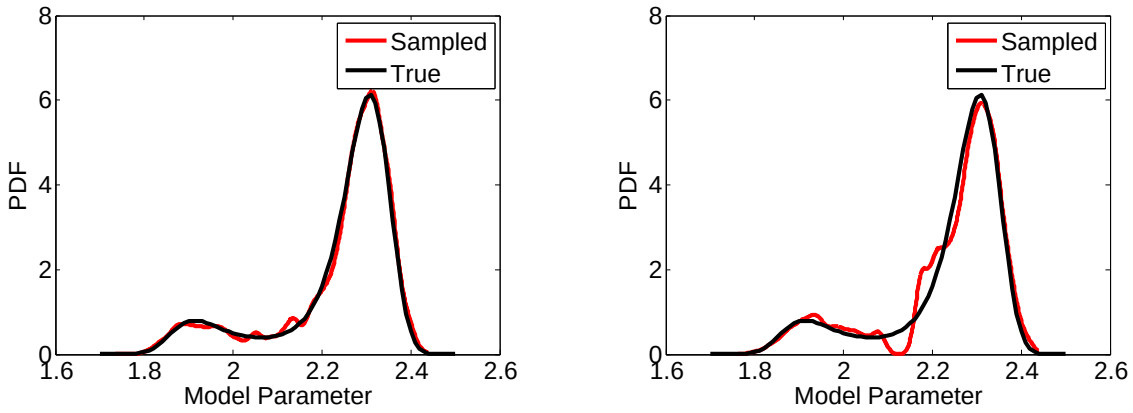
probability for the new proposed state is $\alpha = \min(1, \frac{\pi(\widetilde{m}_i)q(m_{i-1})}{q(\widetilde{m}_i)\pi(m_{i-1})})$. Fig. 5.1(a) indicates that the value of $q(\widetilde{m}_i)$ is very small, thus the value of $\frac{\pi(\widetilde{m}_i)}{q(\widetilde{m}_i)}$ is very large and the new proposed state $\widetilde{m}_i$ will be accepted as the next state in the chain with a probability of 1, i.e., $m_i = \widetilde{m}_i$. Now that the current state in the Markov chain is $m_i = \widetilde{m}_i$, for a new proposed state $\widetilde{m}_{i+1}$, the acceptance probability is $\alpha = \min(1, \frac{\pi(\widetilde{m}_{i+1})q(m_i)}{q(\widetilde{m}_{i+1})\pi(m_i)})$. From Fig. 5.1(a) we can see that the value of $\frac{q(m_i)}{\pi(m_i)}$ is extremely small, because $q(m_i)$ is a very small number. Thus, the acceptance probability for any new proposed state is very small, and because of this, the current state m_i is likely be repeated many times in the chain before we accept a proposed state that is different from m_i. However, with the adaptation step embedded in step 6 of Algorithm 4.2, whenever m_i is repeated more than 20 times, m_i is used to adapt the covariance matrix. This adaptation causes the proposal distribution to become wider and closer to the target pdf as depicted schematically in Fig. 5.1(b). Because of this adaptation, the value of the term $\frac{q(m_i)}{\pi(m_i)}$ in the acceptance probability is larger (see Fig. 5.1(b)) and thus, the acceptance probability for the new proposed state $\tilde{m}_{i+1}$ is larger.

5.1.1 Results for example 1

Using 30 different initial guesses generated randomly from the prior pdf, we run the quasi-Newton trust region optimization algorithm [13] to minimize the objective function $O(m)$ defined by Eq. 4.16. After optimization, we obtained two modes with the Gaussian distributions $\mathcal{N}(2.3077, 0.0025)$ and $\mathcal{N}(1.9141, 0.0035)$. Following step 2 of Algorithm 5.1, 1500 samples are generated from each of the modes to form the set S. Then, we run a Markov chain with 150,000 states, and obtain the posterior pdf shown in Fig. 5.2(a). The results of this plot, indicate that, with 3000 objective function evaluations, we can obtain a good representation of the posterior distribution. For comparison, we also generate 200 samples from each of the modes and form the set S with 400 (200×2) samples, and run a Markov chain with 150,000 states, the posterior pdf obtained, which is shown in Fig. 5.2(b) is not as smooth as the posterior pdf obtained by generating 1500 samples from each mode, and around $m = 2.1$ there are no samples generated from the original proposal pdf. Nevertheless, considering that only 400 samples and 400 runs of the forward model are used to generate the approximate pdf, we are able to obtain an approximation of the target pdf that is roughly correct.

5.1.2 Results for example 2

Using 200 initial guesses in step 1 of Algorithm 5.1, after the optimization process, all minimizing models corresponding to a normalized objective function less than 1.5 are used in k-medoids clustering to generate 10 modes. Here, we rescale the covariance matrix using the same factor as the one used in Algorithm 4.2. We sample each of the Gaussians corresponding to each of the 10 modes until we obtain 200 samples that give a normalized objective function value less than 30. Here we use 30 to save computational cost, we want to discard those models with a huge objective function value, at the same time we do not want to use a number too small to increase the computational cost too much. Since we discard the samples which give a normalized objective function value greater than 30, the total number of simulation runs used to generate the set of 2,000 samples is actually 3600. Fig. 5.3(a)



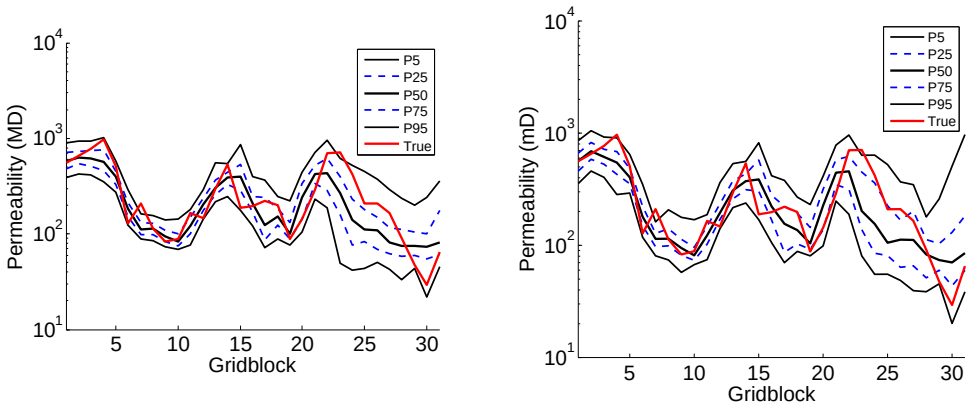
(a) Pdf obtained using 1500 samples from each mode. **(b) Pdf obtained using 200 samples from each mode.**

Figure 5.2: Comparison of target pdf with the distribution obtained using the approximate two-level MCMC, example 1a. (a) Generate 1500 samples from each mode. (b) Generate 200 samples from each mode.

and Fig. 5.4(a) present the posterior distribution of permeability and prediction of water production rate (P5, P25, P50, P75 and P95), respectively, obtained using Algorithm 5.1. In Fig. 5.3 and Fig. 5.4, we compare the results obtained using Algorithm 4.2 and Algorithm 5.1 with 10 modes. The posterior distribution of permeability and prediction of water rate using Algorithm 5.1 is very similar to the posterior distribution obtained using Algorithm 4.2. Thus, for this example, Algorithm 5.1 gives a reasonably accurate characterization of the posterior pdf with a very low computational cost.

5.1.3 Results for example 3

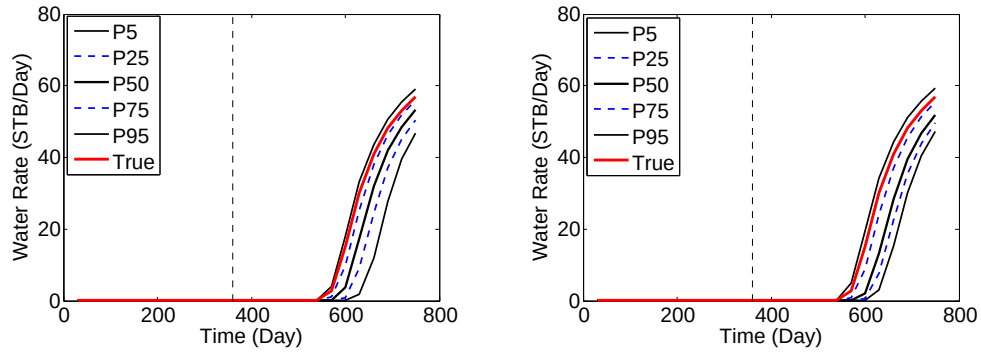
Using 200 initial guesses in step 1 of Algorithm 5.1, we run the quasi-Newton trust region optimization algorithm [13] to minimize $O(m)$ defined by Eq. 4.3. All minimizing models corresponding to a normalized objective function less than 1.5 are used in k-medoids clustering to generate 25 modes which are used to generate all samples in set S in step 2 of Algorithm 5.1. Here, we rescale the covariance matrix using the same factor as the one used in Algorithm 4.2. We sample each individual Gaussian in the GMM until we obtain 100 models, which give a value of the normalized objective function less than 30.



(a) Posterior distribution of permeability using Algorithm 5.1

(b) Posterior distribution of permeability using Algorithm 4.2.

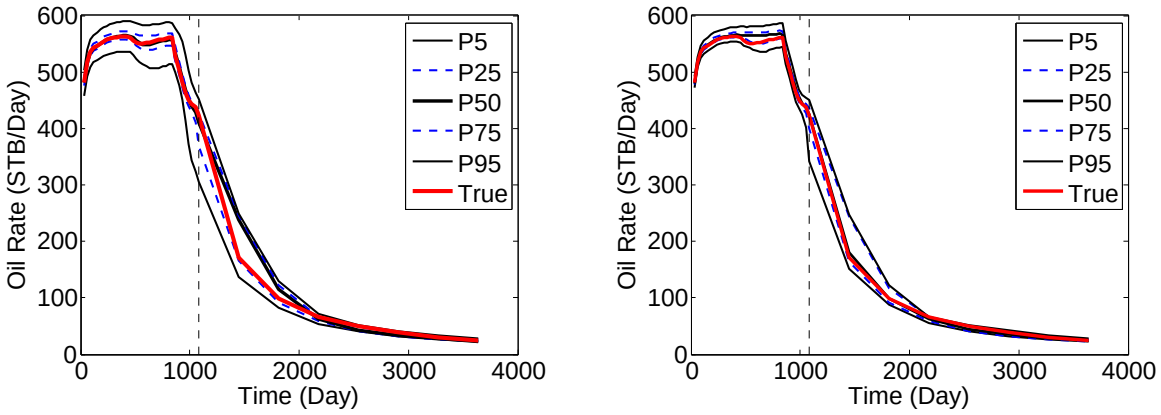
Figure 5.3: Example 2: comparison of posterior permeability distribution using (a) approximate two-level MCMC (algorithm 4.2) (b) two-level MCMC (Algorithm 5.1). In both of the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top).



(a) Posterior distribution of permeability using Algorithm 5.1

(b) Posterior distribution of permeability using Algorithm 4.2.

Figure 5.4: Example 2: comparison of water production rate prediction using (a) approximate two-level MCMC (algorithm 4.2) (b) two-level MCMC (Algorithm 5.1). In both of the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top). The vertical dashed line separates the history matching and prediction periods.



(a) Oil production rate from approximate two-level MCMC.

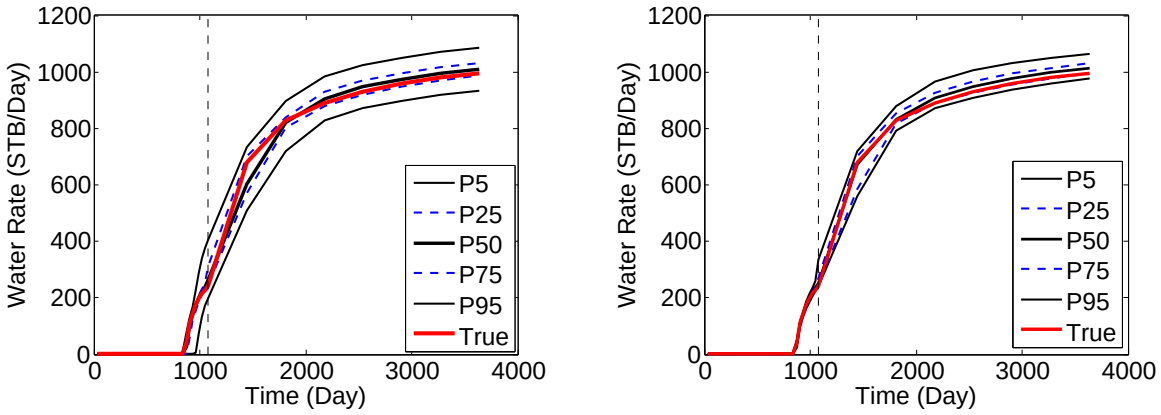
(b) Oil production rate from two-level MCMC.

Figure 5.5: Example 3: prediction of oil production rate obtained using (a) the approximate two-level MCMC method (b) the two-level MCMC method. In both of the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top). The vertical dashed line separates the history matching period and the prediction period.

Thus the set S in step 2 of algorithm 5.1 consists of 2500 models. Note we discard the samples which give a normalized objective function value greater than 30, so the total number of simulation runs used to generate 2,500 modes is 4600. Fig. 5.5 and Fig. 5.6 present the posterior distribution of the oil production rate and water production rate (P5, P25, P50, P75 and P95) obtained using the approximate two-level MCMC (a) and two-level MCMC (b). The posterior distribution obtained using the approximate two-level MCMC is similar to the posterior distribution obtained using the two-level MCMC, but the posterior distribution obtained using the approximate two-level MCMC has a higher uncertainty than the posterior distribution obtained using the two-level MCMC. Note, however, thus with a very low computational cost, the approximate two-level MCMC gives a reasonably accurate characterization of the posterior pdf.

5.1.4 Results for example 4

Using 250 initial guesses in Algorithm 5.1, after the optimization procedure, we use all minimizing models corresponding to a normalized objective function less than 1.5 to



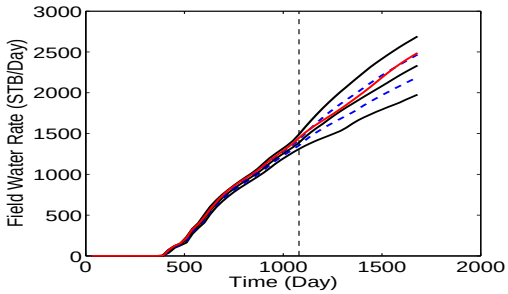
(a) Water production rate from approximate two-level MCMC.

(b) Water production rate from two-level MCMC.

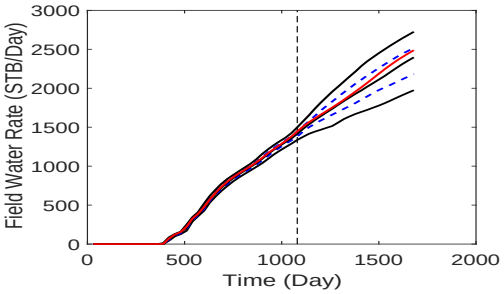
Figure 5.6: Example 3: prediction of water production rate obtained using (a) the approximate two-level MCMC method (b) the two-level MCMC method. In both of the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top). The vertical dashed line separates the history matching period and the prediction period.

generate 35 modes using k-medoids clustering. All the modes are used to generate samples in set S in step 2 of Algorithm 5.1. We sample each of Gaussian until we obtain 200 samples that give a normalized objective function less than 30. For this particular problem, it required the generation of a total of 9000 samples and thus 9000 reservoir simulation runs to obtain 200 samples from each of the 35 Gaussians. Fig. 5.7 compares the uncertainty quantification of the prediction of field water and oil production rate, field water injection rate. Fig. 5.7 and Fig. 5.7 compare the uncertainty quantification of the predictions of the oil production rate and water production rate of all the producers. The results of these figures indicate that the posterior distributions obtained using Algorithm 4.2 and Algorithm 5.1 are very close. Fig. 5.8 presents the comparison of the uncertainty quantification of the water injection rate at all the injectors using Algorithm 4.2 and Algorithm 5.1. Similar to the water and oil production rates, the posterior distributions of water injection rates obtained with Algorithm 4.2 and Algorithm 5.1 are very similar. Again, the results indicate that Algorithm 5.1 provides a characterization of the target pdf that is similar to the one

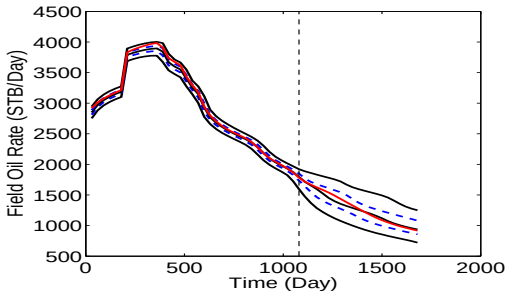
obtained with Algorithm 4.2, but Algorithm 5.1 incurs a much lower computational cost.



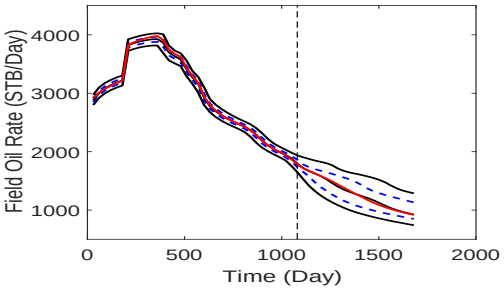
(a) Field Water Production Rate Algorithm 4.2.



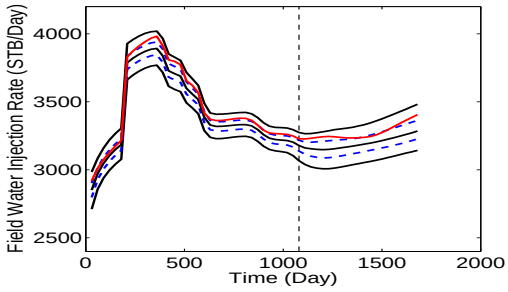
(b) Field Water Production Rate Algorithm 5.1.



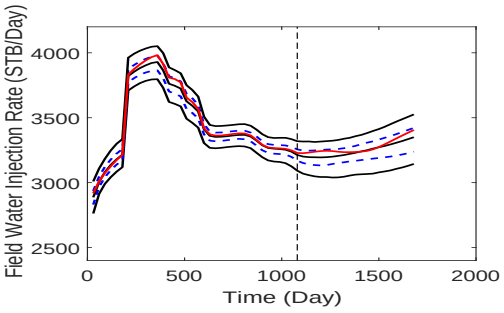
(c) Field Oil Production Rate Algorithm 4.2.



(d) Field Oil Production Rate Algorithm 5.1.

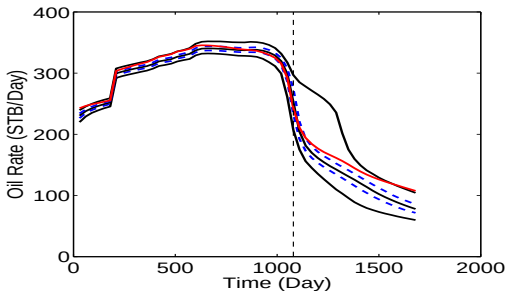


(e) Field Water Injection Rate Algorithm 4.2.

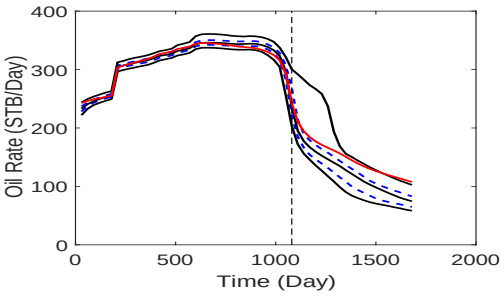


(f) Field Water Injection Rate Algorithm 5.1.

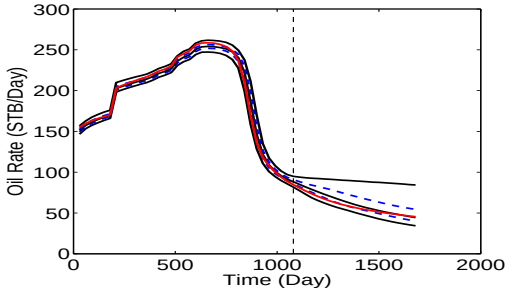
Figure 5.7: The prediction of field water production rate using algorithms 4.2 (a) and 5.1 (b). The prediction of field oil production rate using algorithms 4.2 (c) and 5.1 (d). The prediction of field water injection rate using algorithms 4.2 (e) and 5.1 (f). In all the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top). The vertical dashed line separates the history matching and prediction periods.



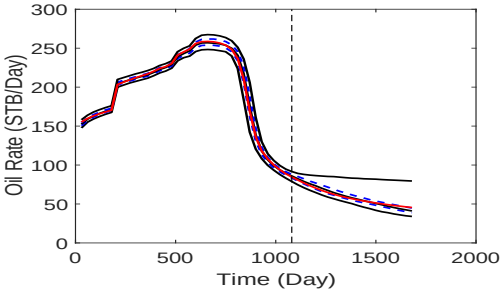
(a) Oil Rate Well 1 Algorithm 4.2.



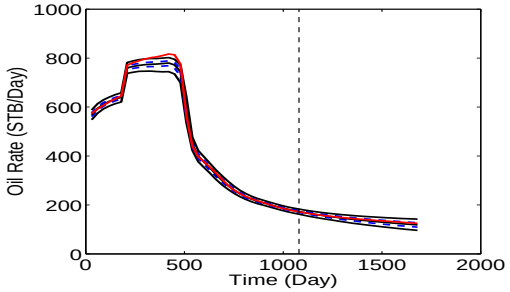
(b) Oil Rate Well 1 Algorithm 5.1.



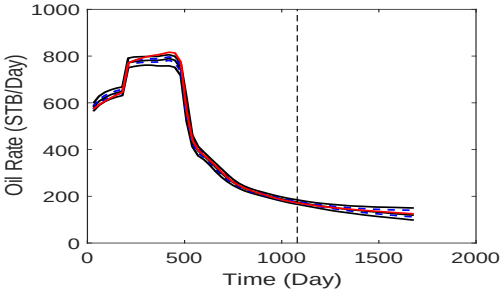
(c) Oil Rate Well 2 Algorithm 4.2.



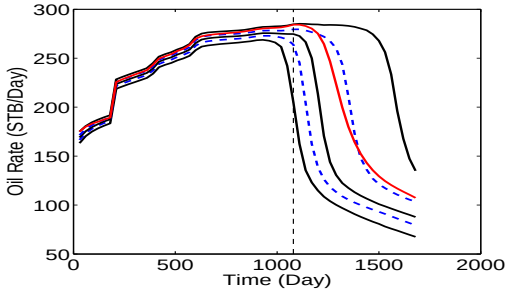
(d) Oil Rate Well 2 Algorithm 5.1.



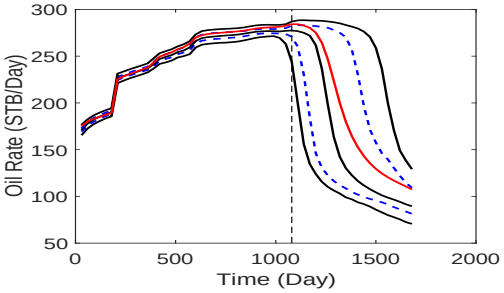
(e) Oil Rate Well 3 Algorithm 4.2.



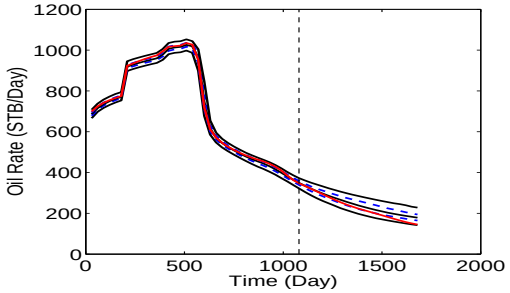
(f) Oil Rate Well 3 Algorithm 5.1.



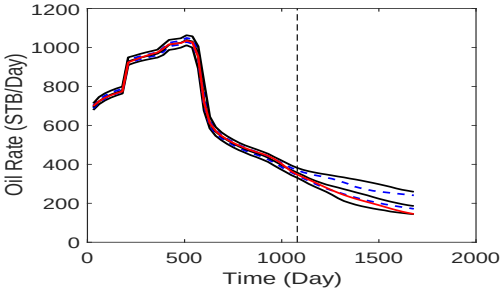
(g) Oil Rate Well 4 Algorithm 4.2.



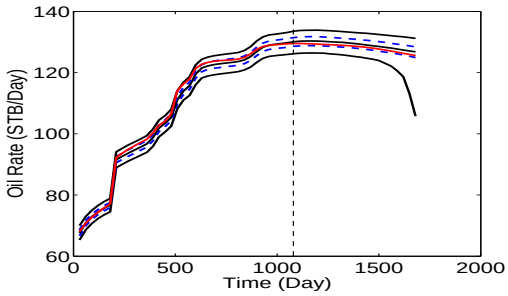
(h) Oil Rate Well 4 Algorithm 5.1.



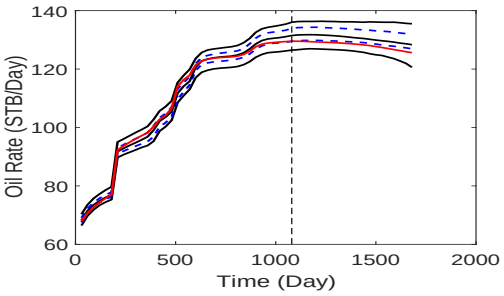
(i) Oil Rate Well 5 Algorithm 4.2.



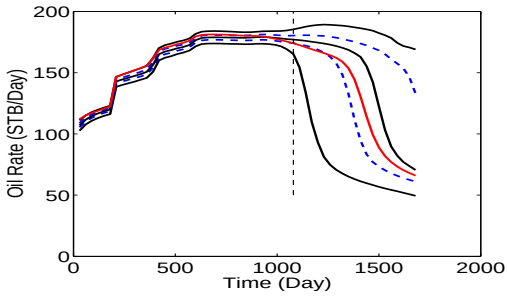
(j) Oil Rate Well 5 Algorithm 5.1.



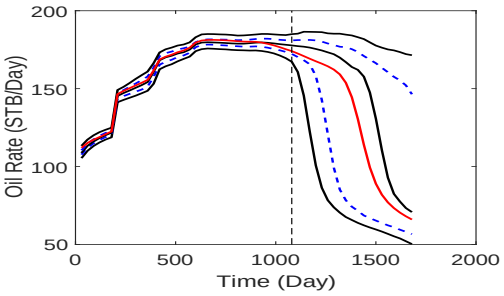
(k) Oil Rate Well 6 Algorithm 4.2.



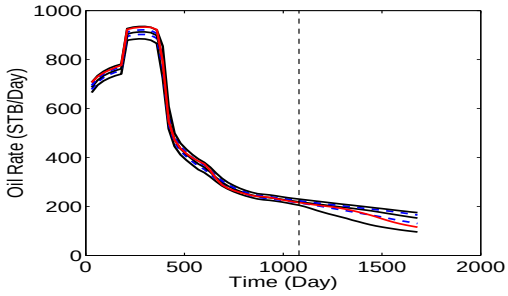
(l) Oil Rate Well 6 Algorithm 5.1.



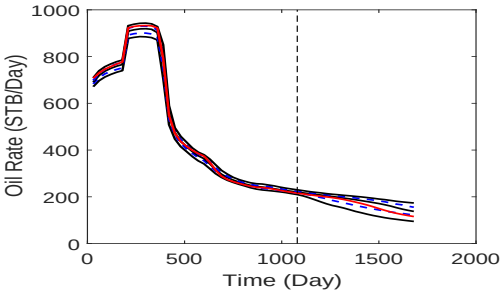
(m) Oil Rate Well 7 Algorithm 4.2.



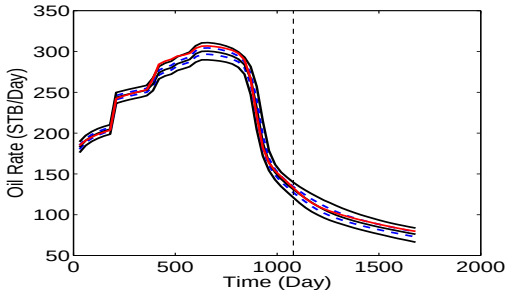
(n) Oil Rate Well 7 Algorithm 5.1.



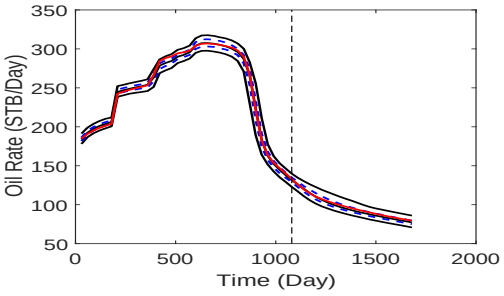
(o) Oil Rate Well 8 Algorithm 4.2.



(p) Oil Rate Well 8 Algorithm 5.1.

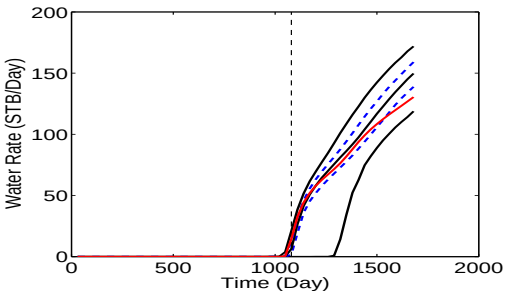


(q) Oil Rate Well 9 Algorithm 4.2.

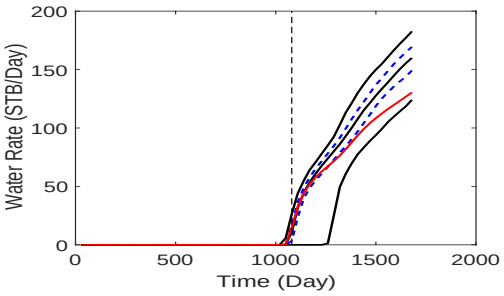


(r) Oil Rate Well 9 Algorithm 5.1.

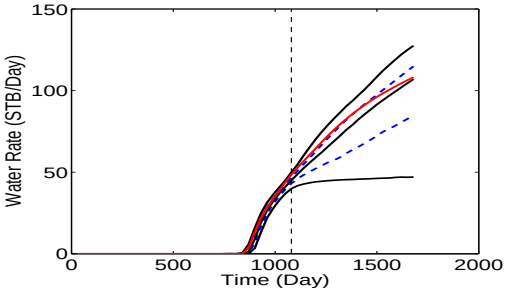
Figure 5.7: The prediction of oil production rate using algorithms 4.2 and 5.1. In all the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top). The vertical dashed line separates the history matching and prediction periods.



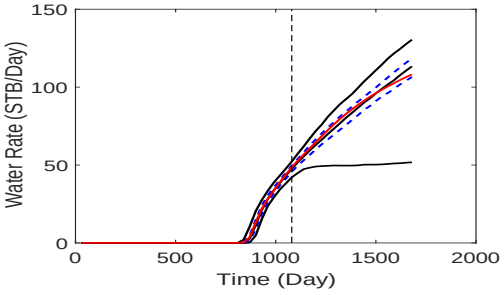
(a) Water Rate Well 1 Algorithm 4.2.



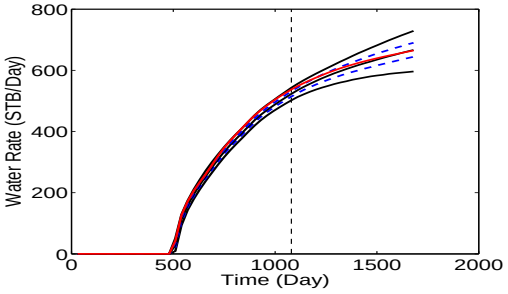
(b) Water Rate Well 1 Algorithm 5.1.



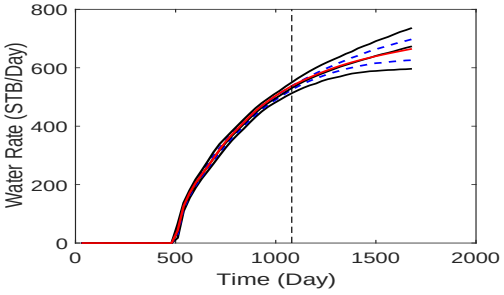
(c) Water Rate Well 2 Algorithm 4.2.



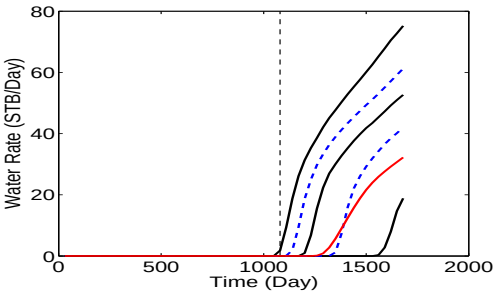
(d) Water Rate Well 2 Algorithm 5.1.



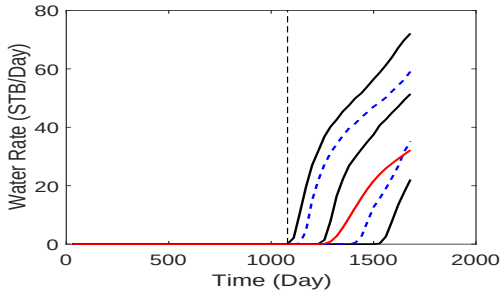
(e) Water Rate Well 3 Algorithm 4.2.



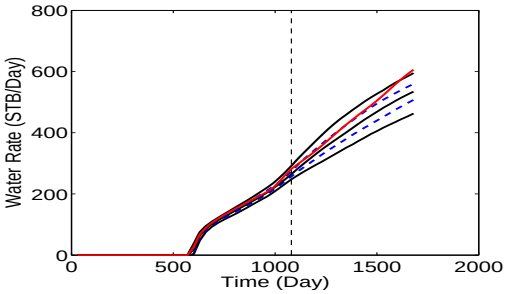
(f) Water Rate Well 3 Algorithm 5.1.



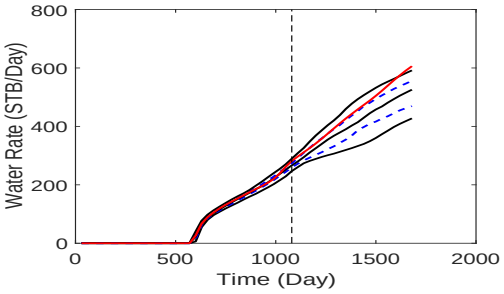
(g) Water Rate Well 4 Algorithm 4.2.



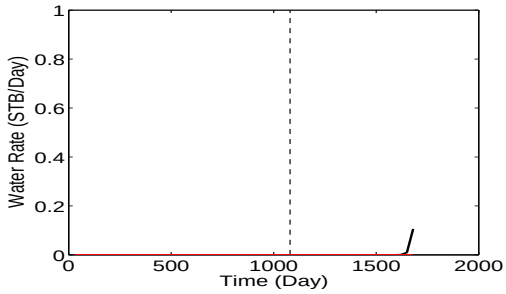
(h) Water Rate Well 4 Algorithm 5.1.



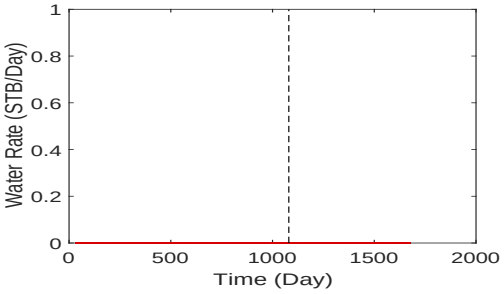
(i) Water Rate Well 5 Algorithm 4.2.



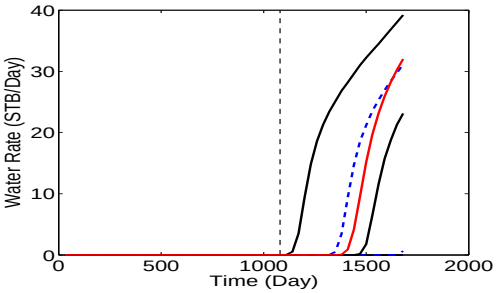
(j) Water Rate Well 5 Algorithm 5.1.



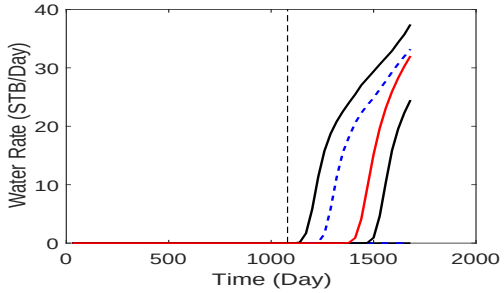
(k) Water Rate Well 6 Algorithm 4.2.



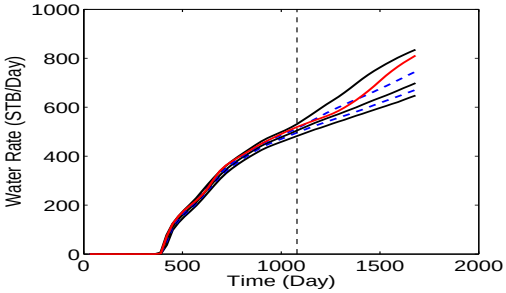
(l) Water Rate Well 6 Algorithm 5.1.



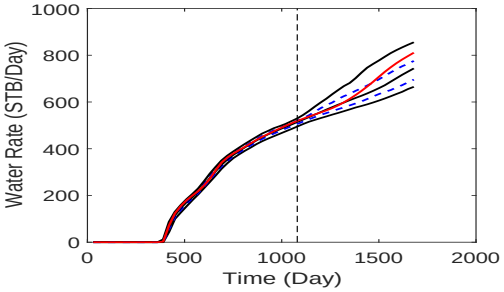
(m) Water Rate Well 7 Algorithm 4.2.



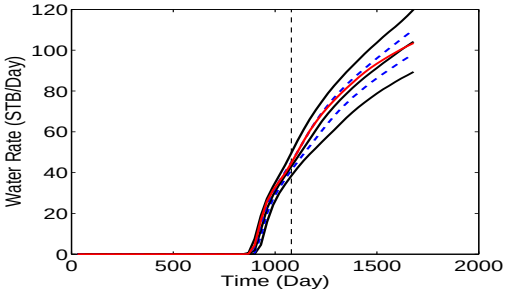
(n) Water Rate Well 7 Algorithm 5.1.



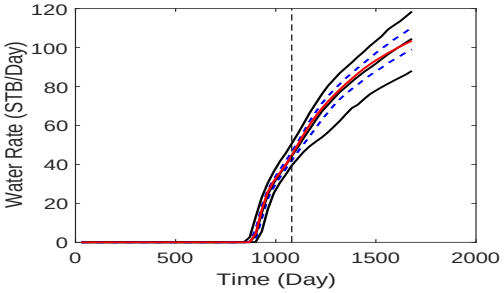
(o) Water Rate Well 8 Algorithm 4.2.



(p) Water Rate Well 8 Algorithm 5.1.

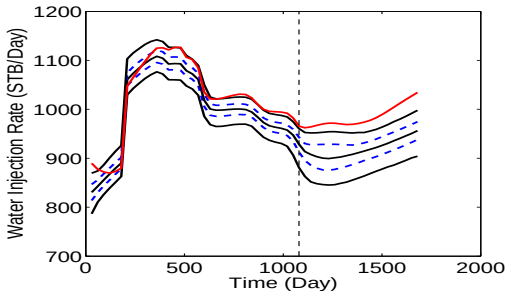


(q) Water Rate Well 9 Algorithm 4.2.

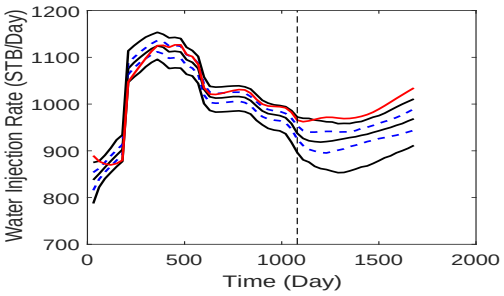


(r) Water Rate Well 9 Algorithm 5.1.

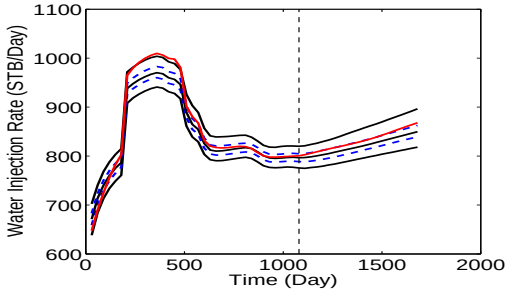
Figure 5.7: The prediction of water production rate using algorithms 4.2 and 5.1. In all the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P5 (bottom), median (middle) and P95 (top). The vertical dashed line separates the history matching and prediction periods.



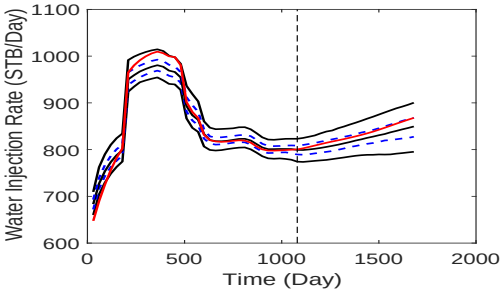
(a) Water Injection Rate Well 1 Algorithm 4.2.



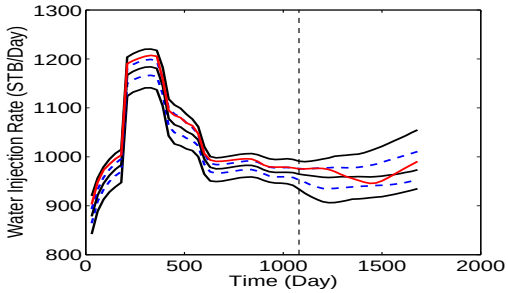
(b) Water Injection Rate Well 1 Algorithm 5.1.



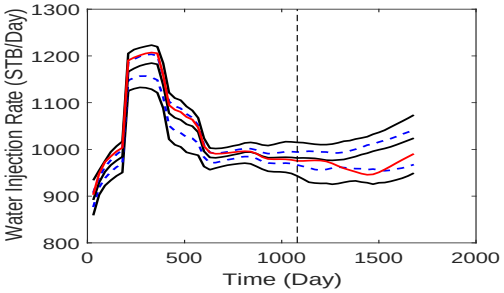
(c) Water Injection Rate Well 2 Algorithm 4.2.



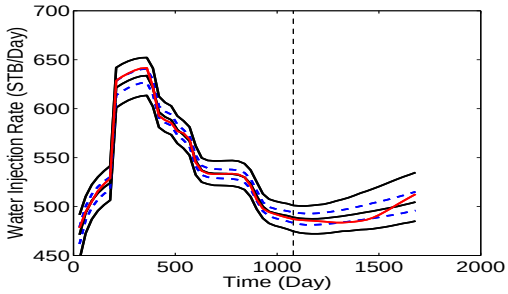
(d) Water Injection Rate Well 2 Algorithm 5.1.



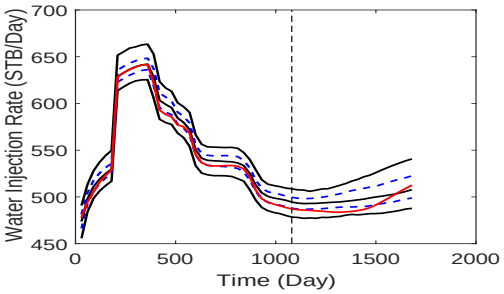
(e) Water Injection Rate Well 3 using Algorithm 4.2.



(f) Water Injection Rate Well 3 using Algorithm 5.1.



(g) Water Injection Rate Well 4 Algorithm 4.2.



(h) Water Injection Rate Well 4 Algorithm 5.1.

Figure 5.8: The prediction of water injection rate using algorithms 4.2 and 5.1. In all the plots, the red curve is the prediction obtained with the true model, the dashed curves in blue are P25 (bottom) and P75 (top). The three black solid curves are P2 (bottom), median (middle) and P98 (top). The vertical dashed line separates the history matching and prediction periods.

CHAPTER 6

CONCLUSIONS

The new two-level MCMC algorithm derived here provides a computationally feasible method to generate realizations (samples) of reservoir model parameters that accurately characterize the posterior pdf conditional to dynamic data. Based on the computational results, this two-level MCMC scheme can achieve a correct sampling of the posterior pdf for the reservoir model parameters with far less computational effort than is required by MCMC methods based on a random walk proposal distribution or population MCMC. This computational efficiency is the expected theoretical result because the two-level scheme employs a Gaussian mixture model (GMM) as a proposal distribution where this GMM is constructed so that it provides a reasonably good approximation to the posterior pdf. The use of a proposal distribution based on GMM proposal distribution constructed by the technique presented here also enable the sampling of a multimodal posterior pdf. Using the samples generated from the two-level scheme enables one to obtain a good approximation of the uncertainty in future reservoir performance predictions. We also present a modified version of the basic two-level MCMC method. The modified MCMC method is far more computationally efficient than the basic two-level method but still provides a reasonable approximation of the target (posterior) pdf of model parameter. However, this modified MCMC does not have a firm theoretical base.

We also compare the performance of several MCMC methods in terms of the quality of history matching, uncertainty characterization and computational cost on a highly nonlinear water flooding case. Among all the methods considered, the two-level MCMC method can sample the posterior pdf correctly with the lowest computational cost. The methods presented here should prove useful for typical history-matching approximations where the

number of model parameters is reduced to one or two dozen.

BIBLIOGRAPHY

[1] Sigurd I. Aanonsen, Geir Nævdal, Dean S. Oliver, Albert C. Reynolds, and Brice Vallés. Review of ensemble Kalman filter in petroleum engineering. 14(3):393–412, 2009.

[2] F. Alabert. The practice of fast conditional simulations through the LU decomposition of the covariance matrix. 19(5):369–386, 1987.

[3] C. Andrieu and J. Thoms. A tutorial on adaptive MCMC. *Statistics and Computing*, 18:343–373, 2008.

[4] James O. Berger. *The Present and Futre of Bayesian Multivariate Analysis*. Purdue University, Department of Statistics, 1992.

[5] A. Bianco, A. Cominelli, L. Dovera, G. Nævdal, and B. Vallès. History matching and production forecast uncertainty by means of the ensemble Kalman filter: A real field application. In *Proceedings of the EAGE/EUROPEC Conference and Exhibition, London, U.K., 11–14 June*, number SPE 107161, 2007.

[6] Luciane Bonet-Cunha, D. S. Oliver, R. A. Rednar, and A. C. Reynolds. A hybrid Markov chain Monte Carlo method for generating permeability fields conditioned to multiwell pressure data and prior information. 11(3):261–271, 1998.

[7] Stephen P. Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7:434–455, 1998.

[8] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of Markov chain Monte Carlo*. CRC press, 2011.

[9] Gerrit Burgers, Peter van Leeuwen, and Geir Evensen. Analysis scheme in the ensemble Kalman filter. 126(6):1719–1724, 1998.

[10] J.N. Carter, P.J. Ballester, Z. Tavassoli, and P.R. King. Our calibrated model has no predictive value: An example from the petroleum industry. *Reliab. Eng. Syst. Saf.*, 91: 1373–1381, 2006.

[11] Jonathan N Carter and Pedro J Ballester. A real parameter genetic algorithm for cluster identification in history matching. *Proceedings of 8th European Conference on the Mathematics of Oil Recovery, Cannes, France, 30 August–02 September*, 2004.

[12] Bailian Chen, Jincong He, Xianhuan Wen, Wen Chen, and Albert C. Reynolds. Pilot design analysis using proxies and Markov chain Monte Carlo method. *Proceedings of the 15th European Conference on the Mathematical of Oil Recovery - Amsterdam, 29 August-1 September*, 2016.

[13] Chaohui Chen, Gaoming Li, and Albert C. Reynolds. Robust constrained optimization of short and long-term NPV for closed-loop reservoir management. In *Proceedings of SPE Reservoir Simulation Symposium*, number SPE 141314, 2011.

[14] Meng-Hui Chen and B. W. Schmeiser. General hit-and-run Monte Carlo sampling for evaluating multidimensional integrals. *Operations Research Letters*, 19(4):161–169, 1996.

[15] Yan Chen and Dean Oliver. Levenberg-Marquardt forms of the iterative ensemble smoother for efficient history matching and uncertainty quantification. 17(4):689–703, 2013.

[16] S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White. MCMC methods for functions: Modifying old algorithms to make them faster. *Statistical Science*, 28:424–446, 2013.

[17] Clayton V. Deutsch and André G. Journel. *GSLIB: Geostatistical Software Library and User's Guide*. Oxford University Press, New York, 1992.

[18] Paul Dostert, Yalchin Efendiev, Thomas Y. Hou, and Wuan Luo. Coarse-gradient Langevin algorithms for dynamic data integration and uncertainty quantification. *Journal of Computational Physics*, 217:123–142, 2006.

[19] Yalchin Efendiev, Akhil Datta-Gupta, Xianlin Ma, and Bani Mallick. Modified Markov chain Monte Carlo method for dynamic data integration using streamline approach. *Mathematical Geology*, 40:213–232, 2008.

[20] Alexandre A. Emerick and Albert C. Reynolds. EnKF-MCMC. In *Proccedings of the SPE EUROPEC/EAGE Annual Conference and Exhibition, Barcelona, Spain, 4–17 June*, number SPE 131375, 2010.

[21] Alexandre A. Emerick and Albert C. Reynolds. History matching a field case using the ensemble Kalman filter with covariance localization. TUPREP research report, The University of Tulsa, 2011.

[22] Alexandre A. Emerick and Albert C. Reynolds. History matching time-lapse seismic data using the ensemble Kalman filter with multiple data assimilations. 16(3):639–659, 2012.

[23] Alexandre A. Emerick and Albert C. Reynolds. Investigation of the sampling performance of ensemble-based methods with a simple reservoir model. 17(2):325–350, 2013.

[24] Alexandre A. Emerick and Albert C. Reynolds. Ensemble smoother with multiple data assimilations. *Computers & Geosciences*, 55:3–15, 2013.

[25] Alexandre A. Emerick and Albert C. Reynolds. History-matching production and seismic data in a real field case using the ensemble smoother with multiple data assim-

ilation. In *Proccedings of the SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA, 18-20 February*, number SPE SPE-163645, 2013.

[26] Geir Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99(C5):10143–10162, 1994.

[27] Geir Evensen and Peter Jan van Leeuwen. An ensemble Kalman smoother for nonlinear dynamics. *Monthly Weather Review*, 128:1852–1867, 2000.

[28] Geir Evensen, J. Hove, H. C. Meisingset, E. Reiso, K. S. Seim, and O. Espelid. Using the EnKF for assisted history matching of a North Sea reservoir model. In *Proceedings of the SPE Reservoir Simulation Symposium, Houston, Texas, 26–28 February*, number SPE 106184, 2007.

[29] William Feller. An introduction to probability theory and its applications.

[30] Kristian Fossum and Trond Mannseth. Parameter sampling capabilities of sequential and simultaneous data assimilation: II. statistical analysis of numerical results. *Inverse Problems*, 30, 2014.

[31] Kristian Fossum and Trond Mannseth. Assessment of ordered sequential data assimilation. *Computational Geosciences*, 19:821–844, 2015.

[32] Guohua Gao and A. C. Reynolds. An improved implementation of the LBFGS algorithm for automatic history matching. 11(1):5–17, 2006.

[33] Guohua Gao, Jeroen C Vink, Chaohui Chen, Mohammadali Tarrahi, Yaakoub El Khamra, et al. Uncertainty quantification for history matching problems with multiple best matches using a distributed Gauss-Newton method. *SPE Annual Technical Conference and Exhibition, 26-28 September, Dubai, UAE*, 2016.

[34] Guohua Gao, Jeroen C. Vink, Chaohui Chen, Yaakoub El Khamra, and Mohammadali Tarrahi. Distributed Gauss-Newton optimization method for history matching problems with multiple best matches. *Computational Geosciences*, pages 1–18, 2017.

[35] Alan E. Gelfand and Adrian F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409, 1990.

[36] A. Gelman, R. O. Roberts, and W. R. Gilks. Efficient Metropolis juming rules. *Bayesian Statistics*, 5:599–607, 1996.

[37] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoraion of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[38] Charles J. Geyer. Markov chain Monte Carlo maximum likelihood. *Computing Science and Statistics: Preceedings of the 13rd Symposium on the Interface*, pages 156–163, 1991.

[39] David E. Goldverg. *Genetic Algorithms in Search, Optimization, and Maching Learning*. Addison Wesley, 1989.

[40] H. Haario, E. Saksmaan, and J. Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7:223–242, 2001.

[41] Nikolaus Hansen. The CMA evolution strategy: A comparing review. In Lozano JA, Larranaga P, Inza I, and Bengoetxea E, editors. *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102, 2006.

[42] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[43] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[44] Vibeke Haugen, Geir Nævdal, Lars-Jrgen Natvik, Geir Evensen, Aina M. Berg, and Kristin M. Flornes. History matching using the ensemble Kalman filter on a North Sea field case. 13(4):382–391, 2008.

[45] Jincong He, Jiang Xie, Pallav Sarma, Xian-Huan Wen, Wen H. Chen, and Jairam Kamath. Model-based a priori evaluation of surveillance programs effectiveness using proxies. In *Proccedings of the SPE Reservoir Simulation Symposium, Houston, Texas, USA, 23-25 February*, number SPE 173229, 2015.

[46] Jincong He, Jiang Xie, Pallav Sarma, Xian-Huan Wen, Wen H. Chen, and Jairam Kamath. Proxy-based work flow for a priori evaluation of data-acquisition programs. Online First, 2016.

[47] J. H. Holland. Adaptation in naural and artificial systems. *University of Michigan Press*, 1975.

[48] P. L. Houtekamer and Herschel L. Mitchell. Data assimilation using an ensemble Kalman filter technique. 126(3):796–811, 1998.

[49] Koji Hukushima and Koji Nemoto. Exchange Monte Carlo method and application to spin glass simulations. *Journal of the Physical society of Japan*, 65:1604–1608, 1996.

[50] Marco A. Iglesias, Kody J. H. Law, and Andrew M. Stuart. Evaluation of Gausian approximations for data assimilation in reservoir models. *Compuational Geoscience*, 17:851–885, 2013.

[51] Robert E. Kass, Bradley P. Carlin, Andrew Gelman, and Radford M. Neal. Mokov chain Monte Carlo in practice: A roundtable discussion. *The American Statistician*, 52:93–100, 1998.

[52] Duc H. Le, Alexandre A. Emerick, and Albert C. Reynolds. An adaptive ensemble smoother with multiple data assimilation for assisted history matching. *SPE Journal*, 21:2195–2207, 2016.

[53] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. Markov chains and mixing times. *American mathematical Society*, 2009.

[54] Ruijian Li. *Conditioning Geostatistical Models to Three-Dimensional Three-Phase Flow Production Data by Automatic History Matching*. Ph.D. thesis, The University of Tulsa, Tulsa, Oklahoma, 2001.

[55] Faming Liang and Wing Hung Wong. Evolutionary Monte Carlo: Applications to c_p model sampling and change point problem. *Math Sciences*, 10:317–342, 2000.

[56] Faming Liang, Chuanhai Liu, and Raymond J Carroll. Stochastic approximation in Monte Carlo computation. *Journal of the American Statistical Association*, 102:305–320, 2007.

[57] Faming Liang, Chuanhai Liu, and Raymond Carroll. *Advanced Markov chain Monte Carlo methods: learning from past samples*, volume 714. John Wiley & Sons, 2011.

[58] J. S. Liu, Faming Liu, and W. H. Wong. The use of multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, 94:121–134, 2000.

[59] Ning Liu and Dean S. Oliver. Evaluation of Monte Carlo methods for assessing uncertainty. 8(2):188–195, 2003.

[60] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trasactions on Information Theory*, 28:129–137, 1982.

[61] Rolf J. Lorentzen, Kjell Kåre Fjelde, Johnny Frøyen, Antonio C.V.M. Lage, Geir Nævdal, and Erlend H. Vefring. Underbalanced drilling: Real time data interpretation and decision support. In *Proceedings of the SPE/IADC Drilling Conference, Amsterdam, Netherlands, 27 February–1 March*, number SPE 67693, 2001.

[62] Rolf J. Lorentzen, Kjell Kåre Fjelde, Johnny Frøyen, Antonio C.V.M. Lage, Geir Nævdal, and Erlend H. Vefring. Underbalanced drilling: Real time data interpretation and

decision support. In *Proceedings of the SPE/IADC Drilling Conference, Amsterdam, Netherlands, 27 February–1 March*, number SPE 67693, 2001.

[63] David Luengo and Luca Martino. Fully adaptive Gaussian mixture Metropolis-Hastings algorithm. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE*, pages 6148–6152, 2013.

[64] X. Ma, M. Al-Harbi, A. Datta-Gupta, and Y. Efendiev. An efficent two-stage sampling method for uncertainty quantification in history matching geological models. 13(1): 77–87, 2008.

[65] Xianlin Ma, Akhil Datta-Gupta, and Yalching Efendiev. An multistage MCMC with nonparametric error model for efficient uncertainty quantification in history matching. In *Proceedings of SPE Annual Technical Conference and Exhibition, Denver, Colorado, 21–24 September*, number SPE 115911, 2008.

[66] James Martin, Lucas C. Wilcox, Carsten Burstedde, and Omar Ghattas. A stochastic Newton MCMC method for large-scale statistical inverse problems with application to seismic inversion. *SIAM Journal on Scientific Computing*, 34:1460–1487, 2012.

[67] K. L. Mengersen and R. L. Tweedie. Rates of convergence of the hastings and Metropolis algorithms. *The Annals of Statistics*, 24:101–121, 1996.

[68] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[69] Linah Mohamed, Michael A. Christie, and Vasily Demyanov. Comparison of stochastic sampling algotithms for uncertainty quantification. *SPE Journal*, 15:31–38, 2010.

[70] Linah Mohamed, Ben Calderhead, Maurizio Filippone, Mike Christie, and Mark Girolami. Population MCMC methods for history matching and uncertainty quantification. *Computational Geosiciences*, 16:423–436, 2012.

[71] Peter Muller. A generic approach to posterior integration and Gibbs sampling. *Technical Report, Purdue University, West Lafayette IN*, 1991.

[72] Peter Muller. Alternatives to the Gibbs sampling scheme. *Technical Report, Institute of Statistics and Decision Sciences, Duke University*, 1993.

[73] Geir Nævdal, Trond Mannseth, and Erland H. Vefring. Near-well reservoir monitoring through ensemble Kalman filter. In *Proceedings of the SPE/DOE Improved Oil Recovery Symposium, 13–17 April*, number SPE 75235, 2002.

[74] Geir Nævdal, L. M. Johnsen, S. I. Aanonsen, and E. H. Vefring. Reservoir monitoring and continuous model updating using ensemble Kalman filter. Number SPE 84372, 2003.

[75] Radford M. Neal. *Bayesian Learning for Neural Networks*. Ph.D. thesis, University of Toronto, Toronto, Ontario, Canada, 1994.

[76] Dean S. Oliver. On conditional simulation to inaccurate data. 28(6):811–817, 1996.

[77] Dean S. Oliver, Nanqun He, and Albert C. Reynolds. Conditioning permeability fields to pressure data. In *Proccedings of the European Conference for the Mathematics of Oil Recovery*, 1996.

[78] Dean S. Oliver, Luciane B. Cunha, and Albert C. Reynolds. Markov chain Monte Carlo methods for conditioning a permeability field to pressure data. 29(1):61–91, 1997.

[79] Dean S. Oliver, Albert C. Reynolds, and Ning Liu. *Inverse Theory for Petroleum Reservoir Characterization and History Matching*. Cambridge University Press, Cambridge, UK, 2008.

[80] I. Petrovska and J.N. Carter. Estimation of distribution algorithms for history matching. *In Proceeding of the 10th European Conference on the Mathematics of Oil Recovery*, 2006.

[81] Javad Rafiee. *Data Assimilation and Uncertainty Quantification with Ensemble methods and Markov Chain Monte Carlo.* Ph.D. dissertation, The University of Tulsa, Tulsa, OK, 2017.

[82] Javad Rafiee and Albert C Reynolds. Theoretical and efficient practical procedures for the generation of inflation factors for ES-MDA. *Inverse Problems*, 33(11):115003, 2017. doi: 10.1088/1361-6420/aa8cb2.

[83] Mickaele Le Ravalec, Benoit Noetinger, and Lin Y. Hu. The FFT moving average (FFT-MA) generator: An efficient numerical method for generating and conditioning Gaussian simulations. *Mathematical Geology*, 32:701–723, 2000.

[84] Albert C. Reynolds, Nanqun He, and Dean S. Oliver. Reducing uncertainty in geostatistical description with well testing pressure data. In Richard A. Schatzinger and John F. Jordan, editors, *Reservoir Characterization–Recent Advances*, pages 149–162. American Association of Petroleum Geologists, 1999.

[85] G. O. Roberts and J. S. Rosenthal. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16:351–367, 2001.

[86] G. O. Roberts and J. S. Rosenthal. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of Applied Probability*, 44:458–475, 2007.

[87] G. O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithm. *The Annals of Applied Probability*, 7(1):110–120, 1997.

[88] Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2:341–363, 1996.

[89] Jeffrey S. Rosenthal. Optimal proposal distributions and adaptive MCMC. *Chapter for MCMC Handbook (Eds. S. Brooks, A. Gelman, G. Jones, and X.-L. Meng)*, 2011.

[90] A. Seiler, J. C. Reivenæs, S. I. Aanonsen, and G. Evensen. Structural uncertainty modeling and updating by production data integration. In *SPE/EAGE Reservoir Characterization and Simulation Conference, Abu Dhabi, UAE 19–21 October*, number SPE 125352, 2009.

[91] R. L. Smith. A Monte Carlo procedure for the random generation of feasible solutions to mathematical programming problems. *Bulletin of the TIMS/ORSA Joint National Meeting, Washington, DC*, 1980.

[92] R. L. Smith. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32:1297–1308, 1984.

[93] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, Philadelphia, USA, 2005.

[94] Kristian Thulin, Gaoming Li, Sigurd Ivar Aanonsen, and Albert C. Reynolds. Estimation of initial fluid contacts by assimilation of production data with EnKF. In *Proceedings of the SPE Annual Technical Conference and Exhibition, Anaheim, California, 11–14 November*, number SPE 109975, 2007.

[95] Luke Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22:1701–1762, 1994.

[96] Håkon Tjelmeland, Henning Omre, and Bjorn Kåre Hegstad. Sampling from Bayesian models in reservoir characterization. Technical Report Statistics No. 2/1994, Norwegian Institute of Technology, Trondheim, Norway, 1994.

[97] Peter Jan van Leeuwen and Geir Evensen. Data assimilation and inverse methods in terms of a probabilistic formulation. *Monthly Weather Review*, 124:2898–2913, 1996.

[98] Yudou Wang, Gaoming Li, and Albert C. Reynolds. Estimation of depths of fluid contacts by history matching using iterative ensemble-Kalman smoothers. 15(2), 2010.

[99] Mohammad Zafari and Albert C. Reynolds. Assessing the uncertainty in reservoir description and performance predictions with the ensemble Kalman filter. 12(3):382–391, 2007.

[100] F. Zhang and A. C. Reynolds. Optimization algorithms for automatic history matching of production data. In *Proceedings of 8th European Conference on the Mathematics of Oil Recovery, Freiberg, Germany, 3–6 September*, 2002.

[101] Fengjun Zhang. *Automatic History Matching of Production Data for Large Scale Problems*. Ph.D. thesis, The University of Tulsa, Tulsa, Oklahoma, 2002.